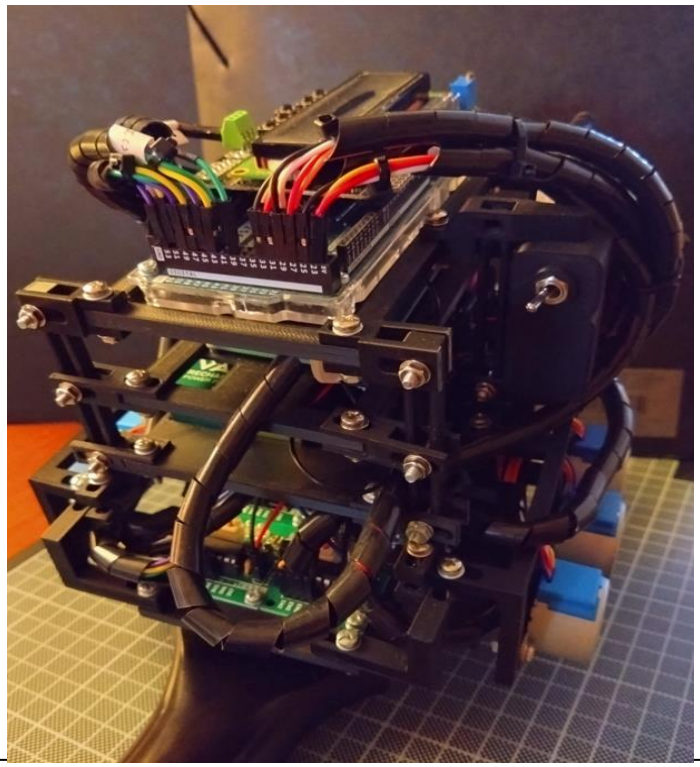
ΠΑΝΕΠΙΣΤΗΜΙΟ ΔΥΤΙΚΗΣ ΑΤΤΙΚΗΣ
ΣΧΟΛΗ ΜΗΧΑΝΙΚΩΝ
ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ & ΗΛΕΚΤΡΟΝΙΚΩΝ ΜΗΧΑΝΙΚΩΝ

Διπλωματική Εργασία

Σχεδίαση και ανάπτυξη συστήματος αυτομάτου ελέγχου του συντονισμού των χορδών κιθάρας



Φοιτητής: Μιχαήλ Συκαράς
ΑΜ: 18387213

Επιβλέπων Καθηγητής

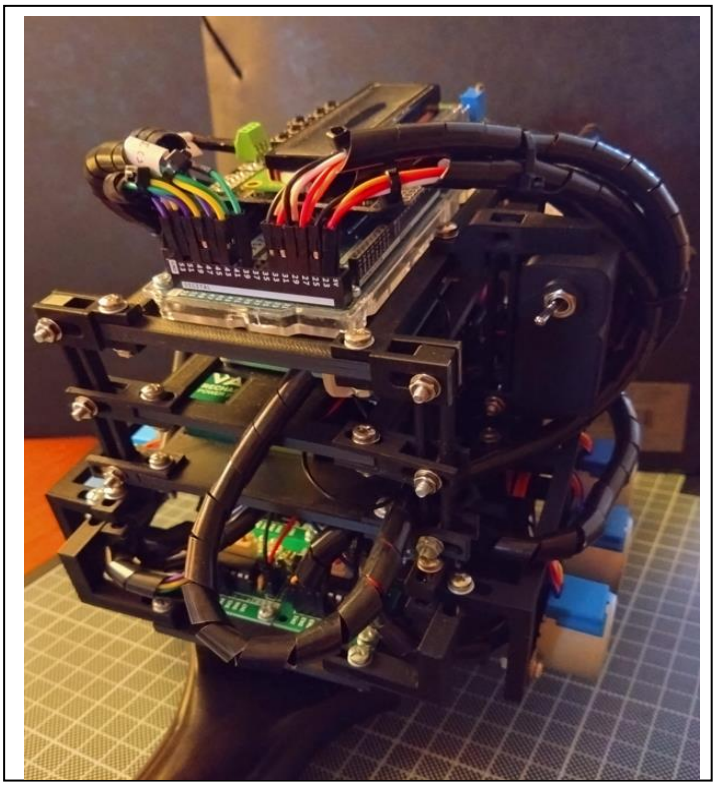
Ξενοφών Διονύσιος Κανδρής
Καθηγητής

ΑΘΗΝΑ-ΑΙΓΑΛΕΩ, Σεπτέμβριος 2024

UNIVERSITY OF WEST ATTICA
FACULTY OF ENGINEERING
DEPARTMENT OF ELECTRICAL & ELECTRONICS ENGINEERING

Diploma Thesis

Design and development of an automatic control system for the tuning of guitar strings



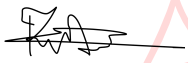
Student: Michail Sikaras
Registration Number: 18387213

Supervisor

Xenofon-Dionysios Kandris
Professor

ATHENS-EGALEO, September 2024

Η Διπλωματική Εργασία έγινε αποδεκτή και βαθμολογήθηκε από την εξής τριμελή επιτροπή:

Ξενοφών-Διονύσιος Κανδρής, Καθηγητής	Ευάγγελος Βαλαμόντες, Καθηγητής	Γρηγόριος Κουλούρας, Αναπληρωτής Καθηγητής
XENOFON DIONYSIOS KANDRIS Digitally signed by XENOFON DIONYSIOS KANDRIS Date: 2024.10.08 22:12:38 +03'00' (Υπογραφή)	EVANGELOS VALAMONTES Digitally signed by EVANGELOS VALAMONTES Date: 2024.10.08 23:14:51 +03'00' (Υπογραφή)	 Digitally signed by GRIGORIOS KOULOURAS DN: c=GR, l=Αθήνα, o=Πανεπιστήμιο Δυτικής Αττικής, sn=KOULOURAS, givenName=GRIGORIOS, serialNumber=6688201272, cn=GRIGORIOS KOULOURAS Date: 2024.10.08 23:26:29 +03'00' (Υπογραφή)

ΔΗΛΩΣΗ ΣΥΓΓΡΑΦΕΑ ΔΙΠΛΩΜΑΤΙΚΗΣ ΕΡΓΑΣΙΑΣ

Ο κάτωθι υπογεγραμμένος Μιχαήλ Συκαράς του Γεωργίου, με αριθμό μητρώου 18387213 φοιτητής του Πανεπιστημίου Δυτικής Αττικής της Σχολής ΜΗΧΑΝΙΚΩΝ του Τμήματος ΗΛΕΚΤΡΟΛΟΓΩΝ ΚΑΙ ΗΛΕΚΤΡΟΝΙΚΩΝ ΜΗΧΑΝΙΚΩΝ,

δηλώνω υπεύθυνα ότι:

«Είμαι συγγραφέας αυτής της διπλωματικής εργασίας και ότι κάθε βοήθεια την οποία είχα για την προετοιμασία της είναι πλήρως αναγνωρισμένη και αναφέρεται στην εργασία. Επίσης, οι όποιες πηγές από τις οποίες έκανα χρήση δεδομένων, ιδεών ή λέξεων, είτε ακριβώς είτε παραφρασμένες, αναφέρονται στο σύνολό τους, με πλήρη αναφορά στους συγγραφείς, τον εκδοτικό οίκο ή το περιοδικό, συμπεριλαμβανομένων και των πηγών που ενδεχομένως χρησιμοποιήθηκαν από το διαδίκτυο. Επίσης, βεβαιώνω ότι αυτή η εργασία έχει συγγραφεί από μένα αποκλειστικά και αποτελεί προϊόν πνευματικής ιδιοκτησίας τόσο δικής μου, όσο και του Ιδρύματος.
Παράβαση της ανωτέρω ακαδημαϊκής μου ευθύνης αποτελεί ουσιώδη λόγο για την ανάκληση του διπλώματός μου.»

Ο Δηλών
Μιχαήλ Συκαράς

(Υπογραφή φοιτητή)

Αφιερώσεις

Στην οικογένειά μου που με στήριξε όλα αυτά τα χρόνια!

Ευχαριστίες

Ένα μεγάλο ευχαριστώ στον κ. Κανδρή Διονύση-Ξενοφών για την υπομονή του και την ευκαιρία να υλοποιήσω την συγκεκριμένη εφαρμογή!

Περίληψη

Η συγκεκριμένη διπλωματική εργασία πραγματεύεται τη σχεδίαση και την ανάπτυξη ενός συστήματος αυτομάτου ελέγχου το οποίο έχει σκοπό τον συντονισμό των χορδών της κιθάρας. Αρχικά, στο πρώτο κεφάλαιο αναλύεται το αναγκαίο θεωρητικό υπόβαθρο το οποίο περιέχει τις απαραίτητες πληροφορίες γύρω από την κιθάρα και την μουσική. Στην συνέχεια, γίνεται αναφορά σε σχετιές ερευνητικές εργασίες. Επίσης, γίνεται αναφορά στην τρισδιάστατη εκτύπωση, τα υλικά και τα ηλεκτρονικά εξαρτήματα των κυκλωμάτων που χρησιμοποιήθηκαν για την εκπόνηση της συγκεκριμένης διπλωματικής εργασίας. Έπειτα, στο δεύτερο κεφάλαιο της διπλωματικής εργασίας αναλύεται το εργαστηριακό μέρος το οποίο περιλαμβάνει τον σχεδιασμό και την κατασκευή των τμημάτων της βάσης, με την χρήση τρισδιάστατης εκτύπωσης, πάνω στην οποία τοποθετείται το σύστημα αυτομάτου ελέγχου. Επίσης, γίνεται ανάλυση των κυκλωμάτων και του τρόπου συνδεσμολογίας μεταξύ τους για να προκύψει το τελικό σύστημα αυτομάτου ελέγχου για τον συντονισμό των χορδών της κιθάρας. Στο τρίτο κεφάλαιο, γίνεται επεξήγηση της λειτουργίας του συστήματος αυτομάτου ελέγχου καθώς και του προγράμματος που γράφτηκε και στο οποίο βασίζεται η λειτουργία του συστήματος. Τέλος, αναφέρονται τα συμπεράσματα για το κατά πόσο το σύστημα αυτομάτου ελέγχου κουρδίσματος χορδών ανταποκρίνεται στο σκοπό και στο στόχο της συγκεκριμένης διπλωματικής εργασίας. Επίσης, γίνεται αναφορά στις μελλοντικές ιδέες για εξέλιξη του συστήματος αυτομάτου ελέγχου τόσο για την κιθάρα όσο και για άλλα έγχορδα μουσικά όργανα.

Λέξεις – κλειδιά

Αντάπτορες, αυτόματο κούρδισμα, βηματικός κινητήρας, επαναφορτιζόμενες μπαταρίες, κιθάρα, κλειδιά, κουρδιστήρι, κύκλωμα ενίσχυσης σήματος, μικρόφωνο επαφής Korg CM-300, νότα, Arduino Mega 2560, πλακέτα οδήγησης βηματικού κινητήρα, συντονισμός χορδών, συχνότητα χορδών, ταλάντωση χορδών, τρισδιάστατη εκτύπωση, χορδές

Abstract

This diploma thesis studies the design and development of an automatic control system for the tuning of guitar strings. Firstly, in Chapter 1, the necessary theoretical background is analyzed which contains the essential information for guitar and music. Subsequently, relevant research works are referred. Also, an analysis related with 3D printing, and the materials and the electronic components for the circuits that have been used to prepare this diploma thesis is provided. Next, in Chapter 2 there is a reference about the laboratory part of this diploma thesis, which contains the designs and development of the plastic base parts with 3D printing in order to place the automatic system on the guitar. Also, there is an analysis about the electronic circuits and how these connect to each other in order to create the control system for tuning the guitar strings. In Chapter 3, both the operation of the automatic control system and the program that has been developed for the operation of the system are analytically described. At the end, concluding remarks are presented and proposals for future development of the system in order to support not only guitars but also other stringed instruments are made.

Keywords

Adapter, Arduino Mega 2560, automatic tuning, contact microphone Korg CM-300, guitar, keys, music note, rechargeable batteries, signal amplifier circuit, stepper motor, stepper motor driver shield, strings, string frequency, string swing, string tuning, tuner, 3D printing

Περιεχόμενα

Κατάλογος Πινάκων **11**

Κατάλογος Εικόνων **13**

Αλφαβητικό Ευρετήριο **29**

ΕΙΣΑΓΩΓΗ **30**

Αντικείμενο της διπλωματικής εργασίας **30**
Σκοπός και στόχοι **30**
Μεθοδολογία **30**
Καινοτομία **31**
Δομή **31**

1 ΚΕΦΑΛΑΙΟ 1ο : Θεωρητικό Μέρος **32**
1.1 Η ιστορία της κιθάρας και η ανατομία της **32**
1.1.1 Ιστορική αναδρομή 32
1.1.2 Η ανατομία της κιθάρας 56
1.1.3 Τα κουρδίσματα της κιθάρας 62
1.2 Αναφορά σε σχετικές εργασίες **71**
1.2.1 Fully Automated Guitar Tuner 72
1.2.2 Automatic Acoustic Guitar Tuner 75
1.2.3 A Digital Guitar Tuner 77
1.2.4 Robotic Electric Guitar Tuner 81
1.2.5 A review on guitars tuners 85
1.2.6 Automatic guitar tuner 90
1.2.7 Final Project ECE 470 Microcontroller Based Guitar Tuner 110
1.2.8 Design of an Electric Guitar Tuner 111
1.2.9 Design and development of a low cost automatic stringed instrument tuner 114
1.2.10 Automatic Guitar Tuner 117
1.3 Η πλατφόρμα ανάπτυξης και το προγραμματιστικό περιβάλλον του Arduino **126**
1.3.1 Γενικά 126
1.3.2 Το προγραμματιστικό περιβάλλον του Arduino 128
1.4 Τρισδιάστατη εκτύπωση (3D Printing) **131**
1.4.1 Χαρακτηριστικά & κατηγορίες των τρισδιάστατων εκτυπωτών (3D Printers) 132
1.4.2 Εφαρμογές σχεδίασης με γραφικό περιβάλλον για τρισδιάστατη εκτύπωση 140

2 ΚΕΦΑΛΑΙΟ 2ο : Εργαστηριακό Μέρος **146**
2.1 Η σχεδίαση και η ανάπτυξη του συστήματος αυτομάτου ελέγχου **146**
2.2 Ανάλυση των υποσυστημάτων **151**
2.2.1 Το Arduino Mega 2560 152
2.2.2 Το μικρόφωνο επαφής Korg CM300 153
2.2.3 Η πλακέτα ενίσχυσης και μετατόπισης του σήματος 154
2.2.4 Ο βηματικός κινητήρας 28BYJ-48 και ο οδηγός ULN2003 167
2.2.5 LCD 16 χαρακτήρων και 2 σειρών με ενσωματωμένο πληκτρολόγιο 183
2.2.6 Τροφοδοσία του συστήματος και τριπολικός διακόπτης (ON – OFF – ON) 187
2.3 Τρισδιάστατοι εκτυπωτές και τρισδιάστατη εκτύπωση **189**
2.3.1 Τρισδιάστατοι εκτυπωτές (3D Printers) 189
2.3.2 Τρισδιάστατη σχεδίαση 192

3 ΚΕΦΑΛΑΙΟ 3ο : Επεξήγηση της λειτουργίας του συστήματος **221**
3.1 Ανάλυση κώδικα προγραμματισμού **221**
3.1.1 Ανάλυση των μεταβλητών, των σταθερών και της βιβλιοθήκης (Κώδικας προγραμματισμού Μέρος 1ο) 221
3.1.2 Ανάλυση της συνάρτησης setup (Κώδικας προγραμματισμού Μέρος 2ο) 224

3.1.3 Ανάλυση της συνάρτησης loop και της συνάρτησης Menu (Κώδικας προγραμματισμού Μέρος 3ο) 228
3.1.4 Ανάλυση της συνάρτησης ButtonRead (Κώδικας προγραμματισμού μέρος 4ο, 5ο, 6ο, 7ο & 8ο) 231
3.1.5 Ανάλυση των συναρτήσεων reset (), checkClipping(), FREQ_DETECTION() & FREQ_DETECTION_STOP() (Κώδικας προγραμματισμού Μέρος 9ο) 337
3.1.6 Ανάλυση της ρουτίνας εξυπηρέτησης διακοπής (ISR) (Κώδικας προγραμματισμού Μέρος 10ο) 340
3.1.7 Ανάλυση της συνάρτησης FREQ_DETECTION1() (Κώδικας προγραμματισμού Μέρος 11ο) 345
3.1.8 Ανάλυση των συναρτήσεων Clockwise_eStep_Motor () & Counterclockwise_eStep_Motor () (Κώδικας προγραμματισμού Μέρος 12ο) 360
3.1.9 Ανάλυση των συναρτήσεων Clockwise_BStep_Motor () & Counterclockwise_BStep_Motor () (Κώδικας προγραμματισμού Μέρος 13ο) 363
3.1.10 Ανάλυση των συναρτήσεων Clockwise_GStep_Motor () & Counterclockwise_GStep_Motor () (Κώδικας προγραμματισμού Μέρος 14ο) 366
3.1.11 Ανάλυση των συναρτήσεων Clockwise_DStep_Motor () & Counterclockwise_DStep_Motor () (Κώδικας προγραμματισμού Μέρος 15ο) 369
3.1.12 Ανάλυση των συναρτήσεων Clockwise_AStep_Motor () & Counterclockwise_AStep_Motor () (Κώδικας προγραμματισμού Μέρος 16ο) 372
3.1.13 Ανάλυση των συναρτήσεων Clockwise_EStep_Motor () & Counterclockwise_EStep_Motor () (Κώδικας προγραμματισμού Μέρος 17ο) 375
3.1.14 Ανάλυση των συναρτήσεων eStep_Motor_Stop (), BStep_Motor_Stop (), GStep_Motor_Stop (), DStep_Motor_Stop (), AStep_Motor_Stop () & EStep_Motor_Stop () (Κώδικας προγραμματισμού Μέρος 18ο) 378
3.1.15 Ανάλυση της συνάρτησης Manual_Button_Read() (Κώδικας προγραμματισμού μέρος 19ο) 381
3.1.16 Ανάλυση της συνάρτησης Manual_Step_Motor_Control() (Κώδικας προγραμματισμού μέρος 20ο) 385
3.2 Οδηγίες χρήσης του συστήματος αυτομάτου ελέγχου για τον συντονισμό των χορδών της κιθάρας 389
3.2.1 Οδηγίες για την επίτευξη του κλασσικού κουρδίσματος της κιθάρας (Standard Tune) 390
3.2.2 Οδηγίες για την επίτευξη του κουρδίσματος των χορδών ένα ημίτονο κάτω (Half-Step Down Tune) 395
3.2.3 Οδηγίες για την επίτευξη του κουρδίσματος της κιθάρας σε «πεσμένη» Ρε (Drop D Tune) 401
3.2.4 Οδηγίες για τον χειροκίνητο έλεγχο του συστήματος 406
4 ΣΥΜΠΕΡΑΣΜΑΤΑ 411
Βιβλιογραφία – Αναφορές - Διαδικτυακές Πηγές 415
Παράρτημα Α 426
Παράρτημα Β 427

Κατάλογος Πινάκων

Πίνακας 1.1 Το κλασσικό-Standard κούρδισμα της κιθάρας https://en.wikipedia.org/wiki/Guitar_tunings [1]

Πίνακας 1.2 Το κούρδισμα της ανοιχτής Ρε (open D) στην κιθάρα (Vestapol Tuning) https://soundadventurer.com/what-are-the-guitar-string-frequencies/ [2]

Πίνακας 1.3 Το κούρδισμα της ανοιχτής Σολ (open G) στην κιθάρα. https://soundadventurer.com/what-are-the-guitar-string-frequencies/ [3]

Πίνακας 1.4 Το κούρδισμα της ανοιχτής Ντο (open C) στην κιθάρα. https://soundadventurer.com/what-are-the-guitar-string-frequencies/ [4]

Πίνακας 1.5 Το κούρδισμα της Ρε αυξημένης 4ης (Dsus4) στην κιθάρα https://soundadventurer.com/what-are-the-guitar-string-frequencies/ [5]

Πίνακας 1.6 Το κούρδισμα της ανοιχτής Μι (open E) στην κιθάρα https://soundadventurer.com/what-are-the-guitar-string-frequencies/ [6]

Πίνακας 1.7 Το κούρδισμα της ανοιχτής Φα9^{ης} (open F9) στην κιθάρα https://producelikeapro.com/blog/note-frequency-chart/ [7]

Πίνακας 1.8 Το κούρδισμα της κιθάρας σε «πεσμένη» Ρε (Drop D tuning) https://soundadventurer.com/what-are-the-guitar-string-frequencies/ [8]

Πίνακας 1.9 Το κούρδισμα της κιθάρας σε διπλή «πεσμένη» Ρε (Double Drop D tuning) https://producelikeapro.com/blog/note-frequency-chart/ [9]

Πίνακας 1.10 Το κούρδισμα της κιθάρας σε «πεσμένη» Ντο (Drop C tuning) https://producelikeapro.com/blog/note-frequency-chart/ [10]

Πίνακας 1.11 Το κούρδισμα της κιθάρας σε διπλή «πεσμένη» Ντο (Double Drop C tuning) https://producelikeapro.com/blog/note-frequency-chart/ [11]

Πίνακας 1.12 Το κούρδισμα της κιθάρας σε κλασσική Ντο (Standard C tuning) https://producelikeapro.com/blog/note-frequency-chart/ [12]

Πίνακας 1.13 Το κούρδισμα της κιθάρας σε «πεσμένη» Σι (Drop B tuning) https://producelikeapro.com/blog/note-frequency-chart/ [13]

Πίνακας 1.14 κούρδισμα της κιθάρας σε διπλή «πεσμένη» Σι (Double Drop B tuning) https://producelikeapro.com/blog/note-frequency-chart/ [14]

Πίνακας 1.15 Το κούρδισμα της κιθάρας σε κλασσική Σι (Standard B tuning) https://producelikeapro.com/blog/note-frequency-chart/ [15]

Πίνακας 1.16 Το κούρδισμα της κιθάρας σε μισό ημίτονο κάτω ή αλλιώς σε κούρδισμα Μιb (Half-Step down tuning or Eb tuning) https://producelikeapro.com/blog/note-frequency-chart/ [16]

Πίνακας 2.1 Απλός κώδικας για την ανάγνωση του ενισχυμένου σήματος από το μικρόφωνο επαφής [17]

Πίνακας 2.2 Κώδικας μετατροπής της ανάλυσης δειγματοληψίας σε 8bit καθώς, σύλληψης του σήματος από το μικρόφωνο επαφής και υπολογισμός της θεμελιώδης συχνότητας https://www.instructables.com/Arduino-Frequency-Detection/ [18]

Πίνακας 2.3 Η σειρά ενεργοποίησης των πηνίων για την επίτευξη της κανονικής κίνησης του κινητήρα https://36projectsblog.wordpress.com/10-arduino-stepper-motor-control/ [19]

Πίνακας 2.4 Η σειρά ενεργοποίησης των πηνίων για την επίτευξη της κίνησης κύματος ή αλλιώς του ολόκληρου βήματος https://upcommons.upc.edu/bitstream/handle/2117/178047/pfg-antonio-hinojosa.pdf [20]

Πίνακας 2.5 Η σειρά ενεργοποίησης των πηνίων για την επίτευξη της κίνησης μισού βήματος https://36projectsblog.wordpress.com/10-arduino-stepper-motor-control/ [21]

Πίνακας 2.6 Ο κώδικας προγραμματισμού για την περιστροφή του άξονα του βηματικού κινητήρα με την χρήση βιβλιοθήκης https://www.instructables.com/Arduino-Stepper-Motor-Example-Sketch-Fixed/ [22]

Πίνακας 2.7 Ο κώδικας προγραμματισμού για την επίτευξη της κανονικής κίνησης του άξονα του βηματικού κινητήρα αριστερόστροφα και δεξιόστροφα [23]

Πίνακας 2.8 Ο κώδικας προγραμματισμού για την επίτευξη της κίνησης κύματος ή αλλιώς του ολόκληρου βήματος του άξονα του βηματικού κινητήρα αριστερόστροφα και δεξιόστροφα [24]

Πίνακας 2.9 Ο κώδικας προγραμματισμού για την επίτευξη της κίνησης μισού βήματος του άξονα του βηματικού κινητήρα αριστερόστροφα και δεξιόστροφα [25]

Πίνακας 2.10 Ο κώδικας προγραμματισμού για τον έλεγχο της LCD οθόνης 16x02 και του ενσωματωμένου πληκτρολογίου [26]

Πίνακας 3.1 Το κομμάτι του κώδικα προγραμματισμού με την κλήση της βιβλιοθήκης, τον καθορισμό των σταθερών, την δήλωση και την αρχικοποίηση των μεταβλητών [27]

Πίνακας 3.2 Ο κώδικας προγραμματισμού της συνάρτησης setup [28]

Πίνακας 3.3 Ο κώδικας προγραμματισμού της συνάρτησης setup [29]

Πίνακας 3.4 Η ανάλυση του κώδικα προγραμματισμού της συνάρτησης ButtonRead() για την ανάγνωση των πλήκτρων [30]

Πίνακας 3.5 Ο κώδικας προγραμματισμού για την αναγνώριση του πλήκτρου SELECT από το σύστημα και του κλασσικού κουρδίσματος (Standard Tune) της κιθάρας [31]

Πίνακας 3.6 Ο κώδικας για το κούρδισμα της κιθάρας ενός ημιτόνου κάτω (Half-Step Down Tune) [32]

Πίνακας 3.7 Ο κώδικας για το κούρδισμα της κιθάρας σε «πεσμένη» Ρε (Drop D Tune) [33]

Πίνακας 3.8 Ο κώδικας για τον χειροκίνητο έλεγχο του συστήματος από τον χρήστη (Manual Control) [34]

Πίνακας 3.9 Ο κώδικας προγραμματισμού των συναρτήσεων reset(), checkClipping(), FREQ_DETECTION() & FREQ_DETECTION_STOP() [35]

Πίνακας 3.10 Ο κώδικας προγραμματισμού της ρουτίνας εξυπηρέτησης διακοπής (ISR) [36]

Πίνακας 3.11 Ο κώδικας προγραμματισμού συνάρτησης FREQ_DETECTION1() [37]

Πίνακας 3.12 Ο κώδικας προγραμματισμού των συναρτήσεων Clockwise_eStep_Motor () & Counterclockwise_eStep_Motor () [38]

Πίνακας 3.13 Ο κώδικας προγραμματισμού των συναρτήσεων Clockwise_BStep_Motor () & Counterclockwise_BStep_Motor () [39]

Πίνακας 3.14 Ο κώδικας προγραμματισμού των συναρτήσεων Clockwise_GStep_Motor () & Counterclockwise_GStep_Motor () [40]

Πίνακας 3.15 Ο κώδικας προγραμματισμού των συναρτήσεων Clockwise_DStep_Motor () & Counterclockwise_DStep_Motor () [41]

Πίνακας 3.16 Ο κώδικας προγραμματισμού των συναρτήσεων Clockwise_AStep_Motor () & Counterclockwise_AStep_Motor () [42]

Πίνακας 3.17 Ο κώδικας προγραμματισμού των συναρτήσεων Clockwise_EStep_Motor () & Counterclockwise_EStep_Motor () [43]

Πίνακας 3.18 Ο κώδικας προγραμματισμού των συναρτήσεων eStep_Motor_Stop (), BStep_Motor_Stop (), GStep_Motor_Stop (), DStep_Motor_Stop (), AStep_Motor_Stop () & EStep_Motor_Stop () [44]

Πίνακας 3.19 Ο κώδικας προγραμματισμού της συνάρτησης Manual_Button_Read() [45]

Πίνακας 3.20 Ο κώδικας προγραμματισμού της συνάρτησης Manual_Step_Motor_Control() [46]

Πίνακας Α Αναλυτικό κοστολόγιο των υλικών για την ανάπτυξη του συστήματος αυτομάτου ελέγχου για τον συντονισμό των χορδών [47]

Κατάλογος Εικόνων

Εικόνα 1.1 Η λύρα στην αρχαία Ελλάδα https://www.tar.gr/alla_moysikologika_moysiki_stin_arxaia_ellada_br_tis_amalias_iliadi-article-729.html?category_id=13 [48]

Εικόνα 1.2 Η κιθάρα στην αρχαία Ελλάδα https://www.tar.gr/alla_moysikologika_moysiki_stin_arxaia_ellada_br_tis_amalias_iliadi-article-729.html?category_id=13 [49]

Εικόνα 1.3 Η μορφή του φόρμιγξ (φόρμιγγα) στην αρχαία Ελλάδα https://www.tar.gr/alla_moysikologika_moysiki_stin_arxaia_ellada_br_tis_amalias_iliadi-article-729.html?category_id=13 [50]

Εικόνα 1.4 Η Vihuela τον 15ο και 16ο αιώνα https://el.wiktionary.org/wiki/%CE%B2%CE%B9%CF%87%CE%BF%CF%85%CE%AD%CE%BB%CE%B1 [51]

Εικόνα 1.5 Η σύγχρονη μορφή της κλασσικής κιθάρας https://el.wikipedia.org/wiki/%CE%9A%CE%B9%CE%B8%CE%AC%CF%81%CE%B1 [52]

Εικόνα 1.6 Σχέδια της Rickenbacker A-22 lap-steel ή αλλιώς τηγάνι (frying pan) https://el.wikipedia.org/wiki/%CE%97%CE%BB%CE%B5%CE%BA%CF%84%CF%81%CE%B9%CE%BA%CE%AE_%CE%BA%CE%B9%CE%B8%CE%AC%CF%81%CE%B1 [53]

Εικόνα 1.7 Η Rickenbacker A-22 ή αλλιώς τηγάνι (frying pan) https://carnegiemuseums.org/carnegie-magazine/summer-2022/objects-of-our-affection-going-electric/ [54]

Εικόνα 1.8 Σχέδια της μεταλλικής ακουστικής κιθάρας με τρικονικό σύστημα αντήχησης της National String Instruments https://chasingguitars.com/nationaldobro-history/ [55]

Εικόνα 1.9 National Tri-Cone Guitar https://chasingguitars.com/nationaldobro-history/ [56]

Εικόνα 1.10 National single resonator guitar ή αλλιώς National Biscuit Guitar https://en.wikipedia.org/wiki/National_String_Instrument_Corporation#Dobro [57]

Εικόνα 1.11 Το εσωτερικό της National single resonator guitar ή αλλιώς National Biscuit Guitar το οποίο μοιάζει με μπισκότο https://chasingguitars.com/nationaldobro-history/ [58]

Εικόνα 1.12 Τα σχέδια της National Single resonator guitar (National Biscuit Guitar) https://chasingguitars.com/nationaldobro-history/ [59]

Εικόνα 1.13 Το αντηχείο της κιθάρας Dobro με την «αραχνοειδές» (Spider-shape) βάση της γέφυρας https://chasingguitars.com/nationaldobro-history/ [60]

Εικόνα 1.14 Η κιθάρα Dobro https://www.vintageguitar.com/30152/a-guide-to-vintage-dobros/ [61]

Εικόνα 1.15 Τα σχέδια της κιθάρας Dobro https://chasingguitars.com/nationaldobro-history/ [62]

Εικόνα 1.16 Ο Paul Tutmac σε φωτογραφία με το μπάσο Audiovox 736 και τις ηλεκτρικές κιθάρες Audiovox https://jivetimerecords.com/northwest/paula-tutmarc/ [63]

Εικόνα 1.17 Το μπάσο Audiovox 736 https://jivetimerecords.com/northwest/paula-tutmarc/ [64]

Εικόνα 1.18 Η ακουστική κιθάρα τύπου Arch Top της εταιρίας Vega Company το 1938 Vegaphone C-75 https://retrofret.com/product.asp?ProductID=7911 [65]

Εικόνα 1.19 Η Epiphone Recording του 1928 https://www.theguitardatabase.com/main_database/epiphone-recording-d-archtop-with-cutaway-u-s-a-1928/ [66]

Εικόνα 1.20 Η Epiphone Masterbilt του 1931 https://www.archtop.com/ac_31bwy.html [67]

Εικόνα 1.21 Η Epiphone Electar του 1935 https://reverb.com/item/2532627-1935-epiphone-electar-electrophone-lap-steel-guitar-w-ohsc [68]

Εικόνα 1.22 Η κιθάρα-άρπα του Gibson το 1907 https://chasingguitars.com/gibson-history/ [69]

Εικόνα 1.23 Η Gibson Style O Acoustic Archtop Guitar του 1918 https://chasingguitars.com/gibson-history/ [70]

Εικόνα 1.24 Η ναυαρχίδα της Gibson, η L-5 Archtop Guitar του 1922-1933 https://reverb.com/p/gibson-l-5-archtop-1922-1933 [71]

Εικόνα 1.25 Η Gibson ES-150 (Electric Spanish) 1936-1957 https://chasingguitars.com/gibson-history/ [72]

Εικόνα 1.26 Το σχέδιο της ηλεκτρικής κιθάρας lap steel του Leo Fender το 1944 https://en.wikipedia.org/wiki/Fender_(company) [73]

Εικόνα 1.27 Η Fender Esquier του 1946 https://reverb.com/item/2192489-uber-rare-fender-1946-esquire-prototype-collectible-snake-head-50th-anniversary-custom-shop-26-50v [74]

Εικόνα 1.28 Η Fender Broadcaster και μετέπειτα Telecaster του 1950 https://fineartamerica.com/featured/fender-telecaster-since-1950-wb-johnston.html [75]

Εικόνα 1.29 Η Fender Stratocaster του 1954 https://reverb.com/p/fender-stratocaster-1954 [76]

Εικόνα 1.30 Η Fender Jazzmaster του 1959 https://www.retrofret.com/product.asp?ProductID=9299 [77]

Εικόνα 1.31 Η Fender Jaguar του 1962 https://www.retrofret.com/product.asp?ProductID=11799&name=Fender-Jaguar-Solid-Body-Electric-Guitar-1962 [78]

Εικόνα 1.32 Το Power Tune Systems της Tronical Company στις Gibson Robot Guitars https://www.premierguitar.com/gear/gibsons-self-tuning-guitar [79]

Εικόνα 1.33 Η γέφυρα τύπου Tune-o-matic με τις πιεζοσέλες οι οποίες μεταδίδουν το σήμα στον μικροεπεξεργαστή https://www.alegree.co.uk/products/piezo-acoustic-tone-tune-o-matic-bridges-crystal-saddles [80]

Εικόνα 1.34 Το σύστημα Transperformance self-tuning guitar https://newatlas.com/the-transperformance-self-tuning-guitar/4951/#gallery:1 [81]

Εικόνα 1.35 Το G-Force tuning system της Tronical company
https://sixstringacoustic.com/g-force-tuning-system-gibsons-automatic-tuning-system [82]

Εικόνα 1.36 Η ανατομία της κλασσικής κιθάρας
https://el.wikipedia.org/wiki/%CE%9A%CE%B9%CE%B8%CE%AC%CF%81%CE%B1 [83]

Εικόνα 1.37 Η ανατομία της ακουστικής κιθάρας
https://www.musicheaven.gr/html/modules.php?name=News&file=article&id=4424 [84]

Εικόνα 1.38 Η ανατομία της ηλεκτρικής κιθάρας
https://www.musicheaven.gr/html/modules.php?name=News&file=article&id=4424 [85]

Εικόνα 1.39 Επτάχορδη κλασσική κιθάρα https://en.wikipedia.org/wiki/Seven-string_guitar [86]

Εικόνα 1.40 Επτάχορδη ηλεκτρική κιθάρα (Gibson Les Paul Standard)
https://www.gearnews.com/gibson-les-paul-standard-7-string-magnificent-seven/ [87]

Εικόνα 1.41 Οκτάχορδη ηλεκτρική κιθάρα (Ibanez RG5328)
https://www.ibanez.com/na/products/detail/rg5328_00_02.html [88]

Εικόνα 1.42 Δεκάχορδη κλασσική κιθάρα https://en.wikipedia.org/wiki/Ten-string_guitar [89]

Εικόνα 1.43 Δεκάχορδη ηλεκτρική κιθάρα (Halo Guitars Octavia 10-string)
https://www.haloguitars.com/store/new-halo-10-string-guitar-limited-edition-octavia-10-fanned-fret [90]

Εικόνα 1.44 Δωδεκάχορδη ακουστική κιθάρα (Gibson J-45 Standard)
https://www.gibson.com/en-US/Acoustic-Guitar/ACCDMY224/Vintage-Sunburst [91]

Εικόνα 1.45 Δεκαοχτάχορδη ηλεκτρική κιθάρα (Gibson EDS-1275 Double Neck)
https://www.long-mcquade.com/181760/Guitars/Electric-Guitars/Gibson/EDS-1275-Double-Neck---Cherry.htm [92]

Εικόνα 1.46 Το πλήρως αυτοματοποιημένο σύστημα κουρδίσματος της κιθάρας (Fully Automated Guitar Tuner) https://courses.engr.illinois.edu/ece445/getfile.asp?id=18170 [93]

Εικόνα 1.47 Η μορφή των δύο καρδιοειδής μικροφώνων
https://courses.engr.illinois.edu/ece445/getfile.asp?id=18170 [94]

Εικόνα 1.48 Το κουρδιστήρι GA-30 της εταιρίας Korg
https://musiclandlive.com/products/korg-ga-30-chromatic-tuner [95]

Εικόνα 1.49 Το ηλεκτρονικό κύκλωμα που συνδέει το κουρδιστήρι με τον κινητήρα
https://dspace.mit.edu/bitstream/handle/1721.1/32878/62588821-MIT.pdf;sequence=2 [96]

Εικόνα 1.50 Το τελικό σύστημα για το αυτόματο κουρδιστήρι ακουστικής κιθάρας https://dspace.mit.edu/bitstream/handle/1721.1/32878/62588821-MIT.pdf;sequence=2 [97]

Εικόνα 1.51 Οι μεταβολές των συχνοτήτων των χορδών κατά την περιστροφή των αντίστοιχων κλειδιών κατά 180^{o} αριστερόστροφα και δεξιόστροφα. https://www.researchgate.net/publication/45888076_A_Digital_Guitar_Tuner [98]

Εικόνα 1.52 Το τελικό σύστημα του Robotic Electric Guitar Tuner https://www.diva-portal.org/smash/get/diva2:1200615/FULLTEXT01.pdf [99]

Εικόνα 1.53 Το σχέδιο του συστήματος αυτομάτου ελέγχου για το Robotic Electric Tuner https://www.diva-portal.org/smash/get/diva2:1200615/FULLTEXT01.pdf [100]

Εικόνα 1.54 Ο αντάπτορας που σχεδιάστηκε για να εφαρμόζει στον άξονα του κινητήρα και στα κλειδιά της κιθάρας για το Robotic Electric Guitar Tuner https://www.diva-portal.org/smash/get/diva2:1200615/FULLTEXT01.pdf [101]

Εικόνα 1.55 Οι τρεις τύποι περιέλιξης των χορδών Roundwound, Flatwound & Groundwound
https://www.ece.ucf.edu/seniordesign/fa2014sp2015/g01/Final_SD1_Proj.pdf [102]

Εικόνα 1.56 Μαγνήτης μονού πηνίου
https://www.ece.ucf.edu/seniordesign/fa2014sp2015/g01/Final_SD1_Proj.pdf [103]

Εικόνα 1.57 Πιεζοηλεκτρικός μαγνήτης
https://www.ece.ucf.edu/seniordesign/fa2014sp2015/g01/Final_SD1_Proj.pdf [104]

Εικόνα 1.58 Μαγνήτης διπλού πηνίου (Humbucker)
https://www.ece.ucf.edu/seniordesign/fa2014sp2015/g01/Final_SD1_Proj.pdf [105]

Εικόνα 1.59 Παράδειγμα κωδικοποιητή περιστροφικού άξονα
https://www.ece.ucf.edu/seniordesign/fa2014sp2015/g01/Final_SD1_Proj.pdf [106]

Εικόνα 1.60 Η βάση από λαμαρίνα με τους κινητήρες συνδεδεμένους πάνω στα κλειδιά του μπάσου https://www.ece.ucf.edu/seniordesign/fa2014sp2015/g01/Final_SD1_Proj.pdf [107]

Εικόνα 1.61 Το τελικό σχέδιο του αυτόματου κουρδιστηριού κιθάρας (Automatic Guitar Tuner) https://www.ece.ucf.edu/seniordesign/fa2014sp2015/g01/Final_SD1_Proj.pdf [108]

Εικόνα 1.62 Το τελικό κύκλωμα για την ανίχνευση της συχνότητας των χορδών μιας κιθάρας
https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.710.11&rep=rep1&type=pdf [109]

Εικόνα 1.63 Η λειτουργία ενός μαγνήτη κιθάρας
https://www.theseus.fi/bitstream/handle/10024/153013/Bhimsen_Budhathoki.pdf;jsessionid=FE1EE4018461CE1927615AA0185C0C22?sequence=1 [110]

Εικόνα 1.64 Το μπλοκ διάγραμμα του συστήματος https://www.theseus.fi/bitstream/handle/10024/153013/Bhimsen_Budhathoki.pdf;jsessionid=FE1EE4018461CE1927615AA0185C0C22?sequence=1 [111]

Εικόνα 1.65 Το κύκλωμα ενίσχυσης του σήματος της κιθάρας https://www.theseus.fi/bitstream/handle/10024/153013/Bhimsen_Budhathoki.pdf;jsessionid=FE1EE4018461CE1927615AA0185C0C22?sequence=1 [112]

Εικόνα 1.66 Η ακουστική κιθάρα Fender T-Bucket 300CE TBK https://ruc.udc.es/dspace/bitstream/handle/2183/23780/2019_Sevilla-Salcedo_Design-developmet-low-cost-instrument-tuner.pdf;jsessionid=B4AE4F89A05C6DE021C7E95CD89E8C4C?sequence=3 [113]

Εικόνα 1.67 Χρήση του πρωτότυπου συστήματος για το κούρδισμα της PE (D3) χορδής https://ruc.udc.es/dspace/bitstream/handle/2183/23780/2019_Sevilla-Salcedo_Design-developmet-low-cost-instrument-tuner.pdf;jsessionid=B4AE4F89A05C6DE021C7E95CD89E8C4C?sequence=3 [114]

Εικόνα 1.68 Τα διαγράμματα μεταβολής της συχνότητας συναρτήσει της μείωσης της τάσης (αριστερά) και της αύξησης της τάσης (δεξιά) των χορδών https://ruc.udc.es/dspace/bitstream/handle/2183/23780/2019_Sevilla-Salcedo_Design-developmet-low-cost-instrument-tuner.pdf;jsessionid=B4AE4F89A05C6DE021C7E95CD89E8C4C?sequence=3 [115]

Εικόνα 1.69 Το διάγραμμα της μεταβολής της συχνότητας της χορδής PE (D3) συναρτήσει των επαναλήψεων της διαδικασίας https://ruc.udc.es/dspace/bitstream/handle/2183/23780/2019_Sevilla-Salcedo_Design-developmet-low-cost-instrument-tuner.pdf;jsessionid=B4AE4F89A05C6DE021C7E95CD89E8C4C?sequence=3 [116]

Εικόνα 1.70 Διαπασών https://upcommons.upc.edu/bitstream/handle/2117/178047/pfg-antonio-hinojosa.pdf [117]

Εικόνα 1.71 Κουρδιστήρι με βελόνα https://upcommons.upc.edu/bitstream/handle/2117/178047/pfg-antonio-hinojosa.pdf [118]

Εικόνα 1.72 Κουρδιστήρι με σφιγκτήρα https://upcommons.upc.edu/bitstream/handle/2117/178047/pfg-antonio-hinojosa.pdf [119]

Εικόνα 1.73 Το αυτόματο κουρδιστήρι Power Tune από την Tronical που χρησιμοποιεί η Gibson https://upcommons.upc.edu/bitstream/handle/2117/178047/pfg-antonio-hinojosa.pdf [120]

Εικόνα 1.74 Το κουρδιστήρι Automatic crank tuner από την εταιρία Jowoom https://upcommons.upc.edu/bitstream/handle/2117/178047/pfg-antonio-hinojosa.pdf [121]

Εικόνα 1.75 Η σειρά εκτέλεσης των διεργασιών https://upcommons.upc.edu/bitstream/handle/2117/178047/pfg-antonio-hinojosa.pdf [122]

Εικόνα 1.76 Το μικρόφωνο επαφής Korg cm200 https://upcommons.upc.edu/bitstream/handle/2117/178047/pfg-antonio-hinojosa.pdf [123]

Εικόνα 1.77 Ο βηματικός κινητήρας 28BYJ-48
https://upcommons.upc.edu/bitstream/handle/2117/178047/pfg-antonio-hinojosa.pdf [124]

Εικόνα 1.78 Ο οδηγός του βηματικού κινητήρα ULN2003
https://upcommons.upc.edu/bitstream/handle/2117/178047/pfg-antonio-hinojosa.pdf [125]

Εικόνα 1.79 Ο βηματικός κινητήρας με τον αντάπτορα για την περιστροφή του κλειδιού
https://upcommons.upc.edu/bitstream/handle/2117/178047/pfg-antonio-hinojosa.pdf [126]

Εικόνα 1.80 Η βάση για την εφαρμογή του κινητήρα με τον αντάπτορα επάνω στην κεφαλή της κιθάρας https://upcommons.upc.edu/bitstream/handle/2117/178047/pfg-antonio-hinojosa.pdf [127]

Εικόνα 1.81 Το σύστημα Automatic Guitar Tuner
https://upcommons.upc.edu/bitstream/handle/2117/178047/pfg-antonio-hinojosa.pdf [128]

Εικόνα 1.82 Arduino UNO https://store.arduino.cc/products/arduino-uno-rev3 [129]

Εικόνα 1.83 Arduino Mega 2560 https://store.arduino.cc/products/arduino-mega-2560-rev3 [130]

Εικόνα 1.84 Arduino Lilypad https://docs.arduino.cc/retired/boards/lilypad-arduino-main-board/ [131]

Εικόνα 1.85 Arduino Nano https://store.arduino.cc/products/arduino-nano [132]

Εικόνα 1.86 Arduino Nano 33BLE SENSE https://store-usa.arduino.cc/products/arduino-nano-33-ble-sense [133]

Εικόνα 1.87 Arduino Nano 33IoT https://store.arduino.cc/products/arduino-nano-33-iot [134]

Εικόνα 1.88 Arduino Due https://store.arduino.cc/products/arduino-due [135]

Εικόνα 1.89 Arduino UNO WiFi REV2 https://store.arduino.cc/products/arduino-uno-wifi-rev2 [136]

Εικόνα 1.90 Η επίσημη ιστοσελίδα του Arduino για την εγκατάσταση του Arduino IDE https://www.arduino.cc/en/software [137]

Εικόνα 1.91 Η διαδικτυακή εφαρμογή ανάπτυξης κώδικα Arduino Web Editor
https://app.arduino.cc/sketches/2e59fca7-4c44-4c07-8c0e-b29f5f87d2c0 [138]

Εικόνα 1.92 Τα τέσσερα βασικά τμήματα του περιβάλλοντος ανάπτυξης Arduino IDE [139]

Εικόνα 1.93 Τρισδιάστατη εκτύπωση με στερεολιθογραφία (stereolithography)
https://www.custompartnet.com/wu/stereolithography [140].

Εικόνα 1.94 Τρισδιάστατη εκτύπωση με επιλεκτική συσσώρευση λέιζερ (Selective Laser Sintering) https://en.wikipedia.org/wiki/Selective_laser_sintering [141]

Εικόνα 1.95 Τρισδιάστατοι εκτυπωτές οικιακής χρήσης (αριστερά) και βιομηχανικής χρήσης (δεξιά) με μοντελοποίηση συντηγμένης εναπόθεσης (Fused Deposition Modeling) ή αλλιώς κατασκευή με συντηγμένη ίνα (Fused Filament Fabrication) https://www.creality.com/products/creality-ender-3-s1-3d-printer & https://3dprintingindustry.com/news/3d-printing-industry-awards-enterprise-3d-printer-year-ffffdm-111574/ [142]

Εικόνα 1.96 Τρισδιάστατοι εκτυπωτές επεξεργασίας ψηφιακού φωτός (Digital Light Process) https://www.openpr.com/news/2855514/digital-light-processing-dlp-3d-printer-market-2022-2029-top [143]

Εικόνα 1.97 Τρισδιάστατος εκτυπωτής (HP Jet Fusion 3D Printer 4200) με πολλαπλό τζετ σύντηξης (Multi Jet Fusion) https://www.researchgate.net/figure/Schematic-of-Multi-Jet-Fusion-MJF-process-a-HP-3D-4200-printer-showing-fusing_fig2_331599125 [144]

Εικόνα 1.98 Τρισδιάστατος εκτυπωτής με PolyJet https://resources.cadimensions.com/cadimensions-resources/sla-vs-polyjet-need-know [145]

Εικόνα 1.99 Τρισδιάστατος εκτυπωτής με άμεση μεταλλική πυροσυσσωμάτωση με λέιζερ (Direct Metal Laser Sintering) https://www.3m3drobotics.com/direct-metal-laser-sinteringdmls/ [146]

Εικόνα 1.100 Τρισδιάστατος εκτυπωτής ο οποίος χρησιμοποιεί την τεχνική τήξης με δέσμη ηλεκτρονίων (Electron Beam Melting) https://www.sciencedirect.com/topics/chemistry/electron-beam-melting [147]

Εικόνα 1.101 Τα έξι βασικά μέρη του γραφικού περιβάλλοντος Autodesk Fusion [148]

Εικόνα 1.102 Η επιλογή του δισδιάστατου επιπέδου κατά την επιλογή της δημιουργίας σχεδίου (Create Sketch) [149]

Εικόνα 1.103 Η μετάβαση σε δισδιάστατο επίπεδο σχεδιασμού και η αλλαγή της βιβλιοθήκης σε Sketch [150]

Εικόνα 1.104 Τα πέντε βασικά μέρη του γραφικού περιβάλλοντος της εφαρμογής UltiMaker Cura [151]

Εικόνα 1.105 Οι ρυθμίσεις που συνίστανται με την τρισδιάστατη εκτύπωση του μοντέλου [152]

Εικόνα 1.106 Επιπρόσθετες ρυθμίσεις για την τρισδιάστατη εκτύπωση του μοντέλου [153]

Εικόνα 2.1 Το μικρόφωνο επαφής Korg CM300 https://www.korg.com/us/products/tuners/cm_300/specifications.php [154]

Εικόνα 2.2 Το Arduino Mega 2560 https://store.arduino.cc/products/arduino-mega-2560-rev3 [155]

Εικόνα 2.3 DC κινητήρες διαφόρων μεγεθών https://grobotronics.com/mechanics-and-wheels/servo/dc-motors/?sl=en&features_hash= [156]

Εικόνα 2.4 DC κινητήρες με μειωτήρα https://grobotronics.com/mechanics-and-wheels/servo/dc-motors/page-4/?sl=en&features_hash= [157]

Εικόνα 2.5 Διάφοροι σερβοκινητήρες https://grobotronics.com/mechanics-and-wheels/servo/servo-el/page-2/ [158]

Εικόνα 2.6 Διάφοροι βηματικοί κινητήρες https://grobotronics.com/mechanics-and-wheels/servo/stepper-motors/ [159]

Εικόνα 2.7 Ο βηματικός κινητήρας 28BYJ-48 με τον αντίστοιχο οδηγό (drive shield) ULN2003 https://www.devobox.com/el/motors/109-step-motor-28byj-48-5vdc.html & https://www.devobox.com/el/motor-control/221-uln2003-stepper-motor-driver-board.html?search_query=step+motor+shield&results=130 [160]

Εικόνα 2.8 Απλή οθόνη LCD 16 χαρακτήρων και 2 σειρών https://www.devobox.com/el/lcd/170-lcd-display-16x2-blue-blacklight-for-arduino.html [161]

Εικόνα 2.9 Οθόνη LCD 16 χαρακτήρων και 2 σειρών με ενσωματωμένο πληκτρολόγιο και δυνατότητα ενσωμάτωσης πάνω στην πλακέτα Arduino Mega2560 https://www.devobox.com/el/lcd/630-lcd-1602-keypad-shield-for-arduino.html?search_query=LCD&results=38 [162]

Εικόνα 2.10 Το Arduino Mega2560 με τον μικροελεγκτή ATmega2560 της Atmel https://store.arduino.cc/products/arduino-mega-2560-rev3 [163]

Εικόνα 2.11 Διάγραμμα των ακροδεκτών του Arduino Mega2560 https://store.arduino.cc/products/arduino-mega-2560-rev3 [164]

Εικόνα 2.12 Μικρόφωνο επαφής CM300 της εταιρίας Korg https://www.korg.com/us/products/tuners/cm_300/index.php [165]

Εικόνα 2.13 Ο τελεστικός ενισχυτής TL082 της Texas Instruments https://www.rapidonline.com/texas-instruments-tl082cp-bi-fet-dual-operational-amplifier-82-0064 [166]

Εικόνα 2.14 Το διάγραμμα συνδέσεως των ακροδεκτών του ολοκληρωμένου κυκλώματος τελεστικών ενισχυτών TL082 https://www.ti.com/lit/ds/symlink/tl082-n.pdf [167].

Εικόνα 2.15 Κύκλωμα ενίσχυσης του σήματος από το μικρόφωνο επαφής https://www.instructables.com/Arduino-Audio-Input/ [168]

Εικόνα 2.16 Το κύκλωμα μετατόπισης του ενισχυμένου σήματος μεταξύ 0Volt και 5Volt https://www.instructables.com/Arduino-Audio-Input/ [169]

Εικόνα 2.17 Το ολοκληρωμένο κύκλωμα ενίσχυσης και μετατόπισης του σήματος από το μικρόφωνο επαφής https://www.instructables.com/Arduino-Audio-Input/ [170]

Εικόνα 2.18 Τα εργαλεία και τα υλικά για την υλοποίηση της πλακέτας ενίσχυσης και μετατόπισης του σήματος [171]

Εικόνα 2.19 Η πλακέτα ενίσχυσης και μετατόπισης του σήματος από το μικρόφωνο επαφής [172]

Εικόνα 2.20 Οι κυματομορφές των χορδών της κιθάρας, α) Μι μπάσο (E), β) Λα (A), γ) Ρε (D), δ) Σολ (G), ε) Σι (B) & στ) Μι καντίνι (e) [173]

Εικόνα 2.21 Ο βηματικός κινητήρας 28BYJ-48 https://www.devobox.com/el/motors/109-step-motor-28byj-48-5vdc.html?search_query=step+motor+shield&results=130 [174]

Εικόνα 2.22 Η διαμόρφωση των καλωδίων και ο τρόπος σύνδεσης των πηνίων του βηματικού κινητήρα 28BYJ-48 https://components101.com/motors/28byj-48-stepper-motor [175]

Εικόνα 2.23 Το σύστημα γραναζιών του βηματικού κινητήρα με αναλογία 64:1 https://lastminuteengineers.com/28byj48-stepper-motor-arduino-tutorial/ [176]

Εικόνα 2.24 πλακέτα οδήγησης ULN2003 του βηματικού κινητήρα https://www.hadex.cz/spec/m513.pdf [177]

Εικόνα 2.25 Ζευγάρια τρανζίστορ Darlington NPN & PNP https://www.electronics-tutorials.ws/transistor/darlington-transistor.html [178]

Εικόνα 2.26 Το ηλεκτρονικό κύκλωμα της πλακέτας οδήγησης ULN2003 για τον βηματικό κινητήρα https://www.electronicoscaldas.com/datasheet/ULN2003A-PCB.pdf [179]

Εικόνα 2.27 Η LCD οθόνη 16x02 με ενσωματωμένο πληκτρολόγιο https://www.devobox.com/el/lcd/630-lcd-1602-keypad-shield-for-arduino.html?search_query=LCD&results=38 [180]

Εικόνα 2.28 Το ηλεκτρονικό σχέδιο της πλακέτας με την LCD οθόνη 16x02 και το ενσωματωμένο πληκτρολόγιο https://www.openimpulse.com/blog/wp-content/uploads/wpsc/downloadables/LCD-Keypad-Shield-V1.0-Schematic.pdf [181]

Εικόνα 2.29 α) Η μπαταρία των 9Volt της εταιρίας Varta, β) Η μπαταρία τύπου 18650 της Samsung των 3.7Volt https://www.batteries.gr/gr/audio-and-video/rechargeable-batteries/mpataria-varta-recharge-accu-power-9v-epanafortizomeni-200mah.html & https://www.devobox.com/el/18650/580-samsung-inr-18650-35e-3500mah-36v-37v.html [182]

Εικόνα 2.30 Ο τριπολικός διακόπτης τριών θέσεων https://grobotronics.com/toggle-switch-on-off-on-3pdt-3a-250v.html [183]

Εικόνα 2.31 Το ηλεκτρολογικό σχέδιο ενός τριπολικού διακόπτη με δύο εξόδους και μία «νεκρή» θέση [184]

Εικόνα 2.32 Ο τρισδιάστατος εκτυπωτής Creality Ender-3 Pro https://www.creality.com/products/ender-3-pro-3d-printer [185]

Εικόνα 2.33 Ο τρισδιάστατος εκτυπωτής Creality Ender-3 S1 https://www.creality.com/products/creality-ender-3-s1-3d-printer [186]

Εικόνα 2.34 Το αναλογικό παχύμετρο με το οποίο μετρήθηκαν διαστάσεις https://www.intool.gr/index.php?id_product=247&controller=product [187]

Εικόνα 2.35 Το σκίτσο του κατασκευαστή των βηματικών κινητήρων 28BYJ-48 με τις διαστάσεις του άξονα https://components101.com/sites/default/files/component_datasheet/28byj48-step-motor-datasheet.pdf [188].

Εικόνα 2.36 Το πρόχειρο σχέδιο με τις διαστάσεις του βηματικού κινητήρα και του άξονά του [189].

Εικόνα 2.37 Το πρόχειρο σχέδιο με τις διαστάσεις των κλειδιών της κιθάρας [190].

Εικόνα 2.38 Το πρόχειρο σχέδιο του αντάπτορα το οποίο πραγματοποιήθηκε σύμφωνα με τις μετρούμενες διαστάσεις [191].

Εικόνα 2.39 Ο τρισδιάστατος σχεδιασμός του αντάπτορα στο Autodesk Fusion [192].

Εικόνα 2.40 Τα διάφορα στρώματα και ο τρόπος γεμίσματος μεταξύ των τοιχωμάτων του αντάπτορα στο λογισμικό τεμαχισμού UltiMaker Cura [193].

Εικόνα 2.41 Ο αντάπτορας μετά την ολοκλήρωση της τρισδιάστατης εκτύπωσης [194].

Εικόνα 2.42 Εφαρμογή του βηματικού κινητήρα επάνω στο κλειδί της κιθάρας με την χρήση του αντάπτορα [195].

Εικόνα 2.43 Το πρόχειρο σχέδιο της βάσης σταθεροποίησης των βηματικών κινητήρων και στήριξης των υπολοίπων εξαρτημάτων [196].

Εικόνα 2.44 Το τρισδιάστατο σχέδιο της βάσης σταθεροποίησης των βηματικών κινητήρων και στήριξης των υπόλοιπων βάσεων στο Autodesk Fusion [197].

Εικόνα 2.45 Τα διάφορα στρώματα και ο τρόπος γεμίσματος μεταξύ των στρωμάτων της βάσης σταθεροποίησης των βηματικών κινητήρων και στήριξης των υπόλοιπων βάσεων στο λογισμικό τεμαχισμού UltiMaker Cura [198].

Εικόνα 2.46 Η εκτυπωμένη βάση σταθεροποίησης των βηματικών κινητήρων και στήριξης των υπολοίπων βάσεων [199].

Εικόνα 2.47 Η βάση σταθεροποίησης των βηματικών κινητήρων και στήριξης των υπολοίπων βάσεων τοποθετημένη επάνω στην κεφαλή της κιθάρας [200].

Εικόνα 2.48 Το πρόχειρο σχέδιο με τις διαστάσεις της πλακέτας οδήγησης των βηματικών κινητήρων [201].

Εικόνα 2.49 Το τρισδιάστατο σχέδιο της βάσης για τις πλακέτες οδήγησης των βηματικών κινητήρων ULN2003 στο Autodesk Fusion [202].

Εικόνα 2.50 Το τρισδιάστατο σχέδιο των συνδέσμων στήριξης της βάσης με τις πλακέτες οδήγησης των βηματικών κινητήρων ULN2003 στο Autodesk Fusion [203].

Εικόνα 2.51 Τα διάφορα στρώματα και ο τρόπος γεμίσματος μεταξύ των στρωμάτων της βάσης στήριξης των πλακετών οδήγησης των βηματικών κινητήρων ULN2003 στο λογισμικό τεμαχισμού UltiMaker Cura [204].

Εικόνα 2.52 Τα διάφορα στρώματα και ο τρόπος γεμίσματος μεταξύ των στρωμάτων των συνδέσμων στήριξης της βάσης με τις πλακέτες οδήγησης ULN2003 στο λογισμικό τεμαχισμού UltiMaker Cura [205].

Εικόνα 2.53 Η βάση στήριξης των πλακετών οδήγησης των βηματικών κινητήρων ULN2003 [206].

Εικόνα 2.54 Η βάση στήριξης των πλακετών ULN2003 μαζί με τους συνδέσμους [207].

Εικόνα 2.55 Η βάση στήριξης των πλακετών οδήγησης των βηματικών κινητήρων ULN2003 στην κεφαλή της κιθάρας [208].

Εικόνα 2.56 Το τρισδιάστατο σχέδιο του αντάπτορα στήριξης των υπολοίπων βάσεων επάνω στην κεφαλή της κιθάρας στο Autodesk Fusion [209].

Εικόνα 2.57 Τα διάφορα στρώματα και ο τρόπος γεμίσματος μεταξύ των στρωμάτων για τους αντάπτορες στήριξης των υπολοίπων βάσεων στην κεφαλή της κιθάρας μέσω του λογισμικού τεμαχισμού UltiMaker Cura [210].

Εικόνα 2.58 Οι αντάπτορες στήριξης των υπολοίπων βάσεων επάνω στις βάσεις σταθεροποίησης των βηματικών κινητήρων [211].

Εικόνα 2.59 Το πρόχειρο σχέδιο με τις διστάσεις των μπαταριών των 9 Volt [212].

Εικόνα 2.60 Το πρόχειρο σχέδιο με τις διαστάσεις της μπαταριοθήκης για την υποδοχή των μπαταριών τύπου 18650 [213].

Εικόνα 2.61 Το τρισδιάστατο σχέδιο της βάσης για την στήριξη των μπαταριών στο Autodesk Fusion [214].

Εικόνα 2.62 Τα στρώματα, το γέμισμα ενδιάμεσα στα στρώματα και ο δεντροειδής τρόπος υποστήριξης των υποδοχών για την βάση στήριξης των μπαταριών στο λογισμικό τεμαχισμού UltiMaker Cura [215].

Εικόνα 2.63 Το τρισδιάστατο σχέδιο των συνδέσμων σταθεροποίησης της βάσης στήριξης των μπαταριών στο Autodesk Fusion [216].

Εικόνα 2.64 Τα στρώματα, ο τρόπος γεμίσματος μεταξύ των στρωμάτων και η υποστήριξη των ευάλωτων από την βαρύτητα σημείων του συνδέσμου σταθεροποίησης της βάσης στήριξης των μπαταριών στο λογισμικό τεμαχισμού UltiMaker Cura [217].

Εικόνα 2.65 Η βάση στήριξης των μπαταριών και οι σύνδεσμοι σταθεροποίησης της επάνω στους αντάπτορες [218].

Εικόνα 2.66 Η βάση στήριξης των μπαταριών βιδωμένη επάνω στους αντάπτορες στήριξης στην κεφαλή της κιθάρας [219].

Εικόνα 2.67 Το πρόχειρο σχέδιο της πλακέτας με τις μετρούμενες διαστάσεις [220].

Εικόνα 2.68 Το τρισδιάστατο σχέδιο της βάσης για την στήριξη της πλακέτας ενίσχυσης και μετατόπισης του σήματος από το μικρόφωνο επαφής στο Autodesk Fusion [221].

Εικόνα 2.69 Τα στρώματα και ο τρόπος γεμίσματος μεταξύ των στρωμάτων της βάσης στήριξης της πλακέτας ενίσχυσης και μετατόπισης του σήματος από το μικρόφωνο επαφής στο λογισμικό τεμαχισμού UltiMaker Cura [222].

Εικόνα 2.70 Το τρισδιάστατο σχέδιο των συνδέσμων σταθεροποίησης της βάσης στήριξης της πλακέτας στο Autodesk Fusion [223].

Εικόνα 2.71 Το γέμισμα ενδιάμεσα στα στρώματα και ο τύπος υποστήριξης των ευάλωτων από την βαρύτητα σημείων του συνδέσμου σταθεροποίησης για την βάση στήριξης της πλακέτας στο λογισμικό τεμαχισμού UltiMaker Cura [224].

Εικόνα 2.72 Η βάση στήριξης της πλακέτας ενίσχυσης και μετατόπισης του σήματος από το μικρόφωνο επαφής και οι σύνδεσμοι σταθεροποίησης της βάσης [225].

Εικόνα 2.73 Η βάση στήριξης της πλακέτας ενίσχυσης και μετατόπισης του σήματος από το μικρόφωνο επαφής μαζί με τους συνδέσμους τοποθετημένη στην κεφαλή της κιθάρας [226].

Εικόνα 2.74 Το πρόχειρο σχέδιο με τις διαστάσεις του Arduino Mega 2560 [227].

Εικόνα 2.75 Το πρόχειρο σχέδιο με τις διαστάσεις της πλακέτας της LCD οθόνης με το πληκτρολόγιο [228].

Εικόνα 2.76 Το τρισδιάστατο σχέδιο της βάσης στήριξης του Arduino Mega 2560 στο Autodesk Fusion [229].

Εικόνα 2.77 Τα στρώματα και το γέμισμα ενδιάμεσα στα στρώματα της βάσης στήριξης του Arduino Mega 2560 στο λογισμικό τεμαχισμού UltiMaker Cura [230].

Εικόνα 2.78 Το τρισδιάστατο σχέδιο του συνδέσμου σταθεροποίησης της βάσης στήριξης του Arduino Mega 2560 στο Autodesk Fusion [231].

Εικόνα 2.79 Τα στρώματα, το γέμισμα μεταξύ των στρωμάτων και η δενδροειδής υποστήριξης κάποιων σημείων του συνδέσμου στο λογισμικό τεμαχισμού UltiMaker Cura [232].

Εικόνα 2.80 Η βάση στήριξης του Arduino Mega 2560 μετά την ολοκλήρωση της τρισδιάστατης εκτύπωσης [233].

Εικόνα 2.81 Οι σύνδεσμοι για την σταθεροποίηση της βάσης στήριξης του Arduino επάνω στους αντάπτορες στήριξης των βάσεων [234].

Εικόνα 2.82 Η βάση στήριξης του Arduino Mega 2560 σταθεροποιημένη επάνω στους αντάπτορες στήριξης των βάσεων [235].

Εικόνα 2.83 Το πρόχειρο σχέδιο με τις διαστάσεις του τριπολικού διακόπτη [236].

Εικόνα 2.84 Το πρόχειρο σχέδιο με τις διαστάσεις της επαναφορτιζόμενης μπαταρίας των 9Volt [237].

Εικόνα 2.85 Το τρισδιάστατο σχέδιο της βάσης στήριξης του τριπολικού διακόπτη και της επαναφορτιζόμενης μπαταρίας των 9 Volt [238].

Εικόνα 2.86 Τα στρώματα, το γέμισμα μεταξύ των στρωμάτων και η δενδροειδής υποστήριξη της βάσης στήριξης του τριπολικού διακόπτη και της επαναφορτιζόμενης μπαταρίας στο λογισμικό τεμαχισμού UltiMaker Cura [239].

Εικόνα 2.87 Η βάση στήριξης του τριπολικού διακόπτη και της επαναφορτιζόμενης μπαταρίας των 9 Volt μετά την ολοκλήρωση της τρισδιάστατης εκτύπωσης [240].

Εικόνα 2.88 Η σταθεροποίηση της βάσης στήριξης του τριπολικού διακόπτη επάνω στην κεφαλή της κιθάρας [241].

Εικόνα 3.1 Το όνομα του συστήματος στην LCD οθόνη [242]

Εικόνα 3.2 Το μήνυμα που προτρέπει τον χρήστη να επιλέξει κούρδισμα [243]

Εικόνα 3.3 Το μενού για την επιλογή κουρδίσματος ή του χειροκίνητου ελέγχου του συστήματος [244]

Εικόνα 3.4 Επιλογή του κλασσικού κουρδίσματος (Standard Tune) και οριστικοποίηση της επιλογής [245]

Εικόνα 3.5 Το μήνυμα για την διέγερση της χορδής Μι μπάσο (E) [246]

Εικόνα 3.6 Το μήνυμα που εμφανίζει την επιθυμητή συχνότητα ταλάντωσης της χορδής Μι μπάσο (E) [247]

Εικόνα 3.7 Το μήνυμα για την διέγερση της χορδής Λα (A) [248]

Εικόνα 3.8 Το μήνυμα που εμφανίζει την επιθυμητή συχνότητα ταλάντωσης της χορδής Λα (A) [249]

Εικόνα 3.9 Το μήνυμα για την διέγερση της χορδής Ρε (D) [250]

Εικόνα 3.10 Το μήνυμα που εμφανίζει την επιθυμητή συχνότητα ταλάντωσης της χορδής Ρε (D) [251]

Εικόνα 3.11 Το μήνυμα για την διέγερση της χορδής Σολ (G) [252]

Εικόνα 3.12 Το μήνυμα που εμφανίζει την επιθυμητή συχνότητα ταλάντωσης της χορδής Σολ (G) [253]

Εικόνα 3.13 Το μήνυμα για την διέγερση της χορδής Σι (B) [254]

Εικόνα 3.14 Το μήνυμα που εμφανίζει την επιθυμητή συχνότητα ταλάντωσης της χορδής Σι (B) [255]

Εικόνα 3.15 Το μήνυμα για την διέγερση της χορδής Μι (e) [256]

Εικόνα 3.16 Το μήνυμα που εμφανίζει την επιθυμητή συχνότητα ταλάντωσης της χορδής Μι (e) [257]

Εικόνα 3.17 Το μήνυμα ολοκλήρωσης του κλασσικού κουρδίσματος (Standard Tune) [258]

Εικόνα 3.18 Επανεμφάνιση του μενού επιλογών μετά την ολοκλήρωση του κουρδίσματος [259]

Εικόνα 3.19 Το όνομα του συστήματος στην LCD οθόνη [260]

Εικόνα 3.20 Το μήνυμα που προτρέπει τον χρήστη να επιλέξει κούρδισμα [261]

Εικόνα 3.21 Το μενού για την επιλογή κουρδίσματος ή του χειροκίνητου ελέγχου του συστήματος [262]

Εικόνα 3.22 Επιλογή του κουρδίσματος όλων των χορδών ένα ημίτονο κάτω (Half Step Down Tune) και οριστικοποίηση της επιλογής [263]

Εικόνα 3.23 Το μήνυμα για την διέγερση της χορδής Μιb μπάσο (Eb) [264]

Εικόνα 3.24 Το μήνυμα που εμφανίζει την επιθυμητή συχνότητα ταλάντωσης της χορδής Μιb μπάσο (Eb) [265]

Εικόνα 3.25 Το μήνυμα για την διέγερση της χορδής Λαb (Ab) [266]

Εικόνα 3.26 Το μήνυμα που εμφανίζει την επιθυμητή συχνότητα ταλάντωσης της χορδής Λαb (Ab) [267]

Εικόνα 3.27 Το μήνυμα για την διέγερση της χορδής Ρεb (Db) [268]

Εικόνα 3.28 Το μήνυμα που εμφανίζει την επιθυμητή συχνότητα ταλάντωσης της χορδής Ρεb (Db) [269]

Εικόνα 3.29 Το μήνυμα για την διέγερση της χορδής Σολb (Gb) [270]

Εικόνα 3.30 Το μήνυμα που εμφανίζει την επιθυμητή συχνότητα ταλάντωσης της χορδής Σολb (Gb) [271]

Εικόνα 3.31 Το μήνυμα για την διέγερση της χορδής Σιb (Bb) [272]

Εικόνα 3.32 Το μήνυμα που εμφανίζει την επιθυμητή συχνότητα ταλάντωσης της χορδής Σιb (Bb) [273]

Εικόνα 3.33 Το μήνυμα για την διέγερση της χορδής Μιb (eb) [274]

Εικόνα 3.34 Το μήνυμα που εμφανίζει την επιθυμητή συχνότητα ταλάντωσης της χορδής Μιb (eb) [275]

Εικόνα 3.35 Το μήνυμα ολοκλήρωσης του κλασσικού κουρδίσματος (Standard Tune) [276]

Εικόνα 3.36 Επανεμφάνιση του μενού επιλογών μετά την ολοκλήρωση του κουρδίσματος [277]

Εικόνα 3.37 Το όνομα του συστήματος στην LCD οθόνη [278]

Εικόνα 3.38 Το μήνυμα που προτρέπει τον χρήστη να επιλέξει κούρδισμα [279]

Εικόνα 3.39 Το μενού για την επιλογή κουρδίσματος ή του χειροκίνητου ελέγχου του συστήματος [280]

Εικόνα 3.40 Επιλογή του κουρδίσματος σε «πεσμένη» Ρε (Drop D Tune) και οριστικοποίηση της επιλογής [281]

Εικόνα 3.41 Το μήνυμα για την διέγερση της χορδής Ρε μπάσο (D) [282]

Εικόνα 3.42 Το μήνυμα που εμφανίζει την επιθυμητή συχνότητα ταλάντωσης της χορδής Ρε μπάσο (D) [283]

Εικόνα 3.43 Το μήνυμα για την διέγερση της χορδής Λα (A) [284]

Εικόνα 3.44 Το μήνυμα που εμφανίζει την επιθυμητή συχνότητα ταλάντωσης της χορδής Λα (A) [285]

Εικόνα 3.45 Το μήνυμα για την διέγερση της χορδής Ρε (D) [286]

Εικόνα 3.46 Το μήνυμα που εμφανίζει την επιθυμητή συχνότητα ταλάντωσης της χορδής Ρε (D) [287]

Εικόνα 3.47 Το μήνυμα για την διέγερση της χορδής Σολ (G) [288]

Εικόνα 3.48 Το μήνυμα που εμφανίζει την επιθυμητή συχνότητα ταλάντωσης της χορδής Σολ (G) [289]

Εικόνα 3.49 Το μήνυμα για την διέγερση της χορδής Σι (B) [290]

Εικόνα 3.50 Το μήνυμα που εμφανίζει την επιθυμητή συχνότητα ταλάντωσης της χορδής Σι (B) [291]

Εικόνα 3.51 Το μήνυμα για την διέγερση της χορδής Μι (e) [292]

Εικόνα 3.52 Το μήνυμα που εμφανίζει την επιθυμητή συχνότητα ταλάντωσης της χορδής Μι (e) [293]

Εικόνα 3.53 Το μήνυμα ολοκλήρωσης του κλασσικού κουρδίσματος (Standard Tune) [294]

Εικόνα 3.54 Επανεμφάνιση του μενού επιλογών μετά την ολοκλήρωση του κουρδίσματος [295]

Εικόνα 3.55 Το όνομα του συστήματος στην LCD οθόνη [296]

Εικόνα 3.56 Το μήνυμα που προτρέπει τον χρήστη να επιλέξει κούρδισμα [297]

Εικόνα 3.57 Το μενού για την επιλογή κουρδίσματος ή του χειροκίνητου ελέγχου του συστήματος [298]

Εικόνα 3.58 Επιλογή του χειροκίνητου ελέγχου του συστήματος (Manual Control) και οριστικοποίηση της επιλογής [299]

Εικόνα 3.59 Το μήνυμα που προτρέπει τον χρήστη να επιλέξει την χορδή που επιθυμεί να ελέγξει [300]

Εικόνα 3.60 Επιλογή χορδών και οριστικοποίηση επιλογής στο μενού χειροκίνητου ελέγχου [301]

Εικόνα 3.61 Το μήνυμα που ενημερώνει τον χρήστη για τα πλήκτρα με τα οποία μπορεί να ελέγξει την περιστροφή του βηματικού κινητήρα της χορδής που έχει επιλέξει [302]

Εικόνα 3.62 Το μήνυμα που ενημερώνει τον χρήστη για το πως μπορεί να ολοκληρώσει τον χειροκίνητο έλεγχο της χορδής που έχει επιλέξει [303]

Εικόνα 3.63 Επιλογή και οριστικοποίηση της επιλογής (Back) στο μενού χειροκίνητου ελέγχου [304]

Εικόνα 3.64 Το αρχικό μενού για την επιλογή κουρδίσματος ή του χειροκίνητου ελέγχου του συστήματος [305]

Αλφαβητικό Ευρετήριο

CM: Contact Microphone

DLP: Digital Light Proccess

DMLS: Direct Metal Laser Sinistering

DoBro: Dopyera Brothers

EBM: Electron Beam Melting

FDM:Fused Deposition Modelin

FFF: Fused Filament Fabrication

IDE: Integrated Development Environment

SLS: Selective Laser Sinistering

MJF: Multi Jet Fusion

ΕΙΣΑΓΩΓΗ

Η κιθάρα είναι ένα από τα πιο γνωστά μουσικά όργανα σε όλο τον κόσμο και εμπεριέχεται σε πάρα πολλά είδη μουσικής. Το κούρδισμα των χορδών είναι πολύ σημαντικό στην κιθάρα όπως και σε όλα τα έγχορδα μουσικά όργανα προκειμένου να υπάρχει αρμονία στον ήχο. Οι επαγγελματίες κιθαρίστες μπορούν να κουρδίσουν ακουστικά μία κιθάρα, ακούγοντας δηλαδή τον ήχο-νότα που παράγεται και κουρδίζουν την αντίστοιχη χορδή. Στους αρχάριους κιθαρίστες αυτό είναι δύσκολο, απαιτητικό και αρκετά χρονοβόρο διότι δεν κατέχουν την απαιτούμενη εμπειρία. Με την χρήση όμως ενός συστήματος αυτομάτου ελέγχου συντονισμού χορδών, το κούρδισμα της κιθάρας μπορεί να γίνει πιο εύκολο, γρήγορο και με ακρίβεια.

Αντικείμενο της διπλωματικής εργασίας

Η συγκεκριμένη διπλωματική εργασία έχει να κάνει με το αυτόματο κούρδισμα των χορδών της κιθάρας. Πιο συγκεκριμένα, το βασικότερο πρόβλημα των μουσικών οργάνων και κυρίως των έγχορδων όπως η κιθάρα, είναι το κούρδισμα των χορδών. Όσο περισσότερο χρησιμοποιούνται οι χορδές από τους κιθαρίστες τόσο αυτές φθείρονται και χαλαρώνουν με αποτέλεσμα να χάνουν το κούρδισμά τους. Επίσης, σημαντικό ρόλο παίζει η υγρασία και η θερμοκρασία του χώρου η οποία δημιουργεί συστολές ή διαστολές στις χορδές με αποτέλεσμα πάλι αυτές να χάνουν το κούρδισμά τους. Ακόμα, όταν σε μία κιθάρα έχουν τοποθετηθεί καινούργιες χορδές τότε αυτές χάνουν το κούρδισμά τους ακόμα πιο γρήγορα διότι δεν έχουν προλάβει να <<στρώσουν>>. Με βάση λοιπόν αυτά το βασικό πρόβλημα είναι το κούρδισμα των χορδών της κιθάρας, το οποίο αρκετές φορές είναι χρονοβόρο κυρίως για τους αρχάριους κιθαρίστες. Γι' αυτό στη συγκεκριμένη διπλωματική εργασία αναλύεται ένα σύστημα αυτομάτου ελέγχου το οποίο θα κουρδίζει τις χορδές εύκολα, γρήγορα με ακρίβεια και θα είναι οικονομικό.

Ο προσδιορισμός του κυρίου θέματος / προβλήματος της εργασίας, με αναφορά στους λόγους για τους οποίους έχει ενδιαφέρον / είναι σημαντικό και επίκαιρο το θέμα.

Σκοπός και στόχοι

Σκοπός της συγκεκριμένης διπλωματικής εργασία είναι η δημιουργία ενός συστήματος αυτομάτου ελέγχου για το κούρδισμα των χορδών της κιθάρας το οποίο είναι λειτουργικό και ταυτόχρονα ασφαλές ως προς τον χρήστη αλλά και ως προς το μουσικό όργαβο. Οι στόχοι του συστήματος αυτομάτου ελέγχου θα πρέπει να είναι γρήγορο, ακριβές, ευέλικτο, εύχρηστο και οικονομικό. Επίσης, πρέπει να περιλαμβάνει κάποιες καινοτομίες που το κάνουν ιδιαίτερο σε σχέση με άλλα παρόμοια συστήματα.

Μεθοδολογία

Αρχικά, αποκτήθηκε το αναγκαίο θεωρητικό υπόβαθρο τόσο για την κιθάρα και την μουσική, καθώς και για τρόπους υλοποίησης παρόμοιων συστημάτων μελετώντας αντίστοιχες εργασίες άλλων συναδέλφων. Στην συνέχεια, αποφασίστηκε ο τρόπος με τον οποίο θα υλοποιηθεί το σύστημα αυτομάτου ελέγχου και έγινε αγορά των απαραίτητων υλικών για την κατασκευή. Στην συνέχεια, δοκιμάστηκε κάθε κομμάτι του συστήματος ξεχωριστά και μετά έγινε σύνδεση μεταξύ τους για να προκύψει το τελικό σύστημα αυτομάτου ελέγχου. Έπειτα, έγινε η σχεδίαση των πλαστικών τμημάτων της βάσης πάνω

στην οποία θα στηριχτεί το σύστημα αυτομάτου ελέγχου και με την σειρά του πάνω στην κιθάρα. Εν συνεχεία, έγιναν δοκιμές του συστήματος πάνω σε ηλεκτρική κιθάρα και τελικές διορθώσεις στον προγραμματισμό. Τέλος, διαπιστώθηκαν τα διάφορα σημεία βελτίωσης του συστήματος τα οποία θα μπορούσαν να αποτελέσουν μελλοντικές εξελίξεις για το σύστημα αυτομάτου ελέγχου για το κούρδισμα των χορδών της κιθάρας.

Καινοτομία

Η καινοτομία της συγκεκριμένης διπλωματικής εργασίας είναι ότι με την ίδια λογική και μέθοδο να δημιουργηθούν αντίστοιχα συστήματα και για άλλα έγχορδα όργανα, όπως μπάσο, πιάνο, μπουζούκι, λαούτο, βιολί κτλ. Επίσης, περιλαμβάνει έξι κινητήρες για την περιστροφή των κλειδιών, δηλαδή ένας για κάθε χορδή. Επιπλέον, το σύστημα θα πραγματοποιεί πάνω από ένα κούρδισμα και θα δίνει την δυνατότητα στον χρήστη τον χειροκίνητο έλεγχο του συστήματος για την αντικατάσταση των χορδών.

Δομή

Στο 1^{ο} κεφάλαιο αναλύεται το αναγκαίο θεωρητικό υπόβαθρο για την συγκεκριμένη διπλωματική εργασία. Στην 1^{η} ενότητα, γίνεται αναφορά στα είδη και στην ανατομία της κιθάρας. Στην 2^{η} ενότητα, αναλύονται οι μουσικές ιδιότητες της κιθάρας καθώς και οι παράγοντες που την επηρεάζουν. Στην 3^{η} ενότητα, αναφέρονται κάποιες παρόμοιες εργασίες οι οποίες μελετήθηκαν και παρείχαν την βάση για την υλοποίηση της παρούσας διπλωματικής εργασίας. Στην 4^{η} ενότητα, πραγματοποιείται μια ανάλυση πάνω στο προγραμματιστικό περιβάλλον του Arduino. Έπειτα, στην 5^{η} ενότητα, αναλύεται η τρισδιάστατη εκτύπωση και οι εφαρμογές για την τρισδιάστατη σχεδίαση.

Στο 2^{ο} κεφάλαιο παρουσιάζεται το εργαστηριακό μέρος της διπλωματικής εργασίας. Στην 1^{η} ενότητα, γίνεται ανάλυση των κυκλωμάτων και των υποσυστημάτων που πρέπει να δημιουργηθούν για την υλοποίηση του συστήματος αυτομάτου ελέγχου για τον συντονισμό των χορδών της κιθάρας. Στη συνέχεια, στην 2^{η} ενότητα, αναλύεται διεξοδικά κάθε ένα υποσύστημα τόσο ως προς το υλισμικό (Hardware), όσο και ως προς το λογισμικό (Software), ώστε να καταστεί εφικτός ο έλεγχός τους. Έπειτα, στην 3^{η} ενότητα, αναλύονται τα χαρακτηριστικά των τρισδιάστατων εκτυπωτών που χρησιμοποιήθηκαν, καθώς και τα τρισδιάστατα σχέδια των τμημάτων της βάσης πάνω στην οποία θα στηριχτούν τα ηλεκτρονικά εξαρτήματα και εν συνέχεια μαζί πάνω στην κιθάρα.

Έπειτα, στο 3^{ο} κεφάλαιο γίνεται αναφορά στην λειτουργία του συστήματος αυτομάτου ελέγχου τόσο στο επίπεδο προγραμματισμό όσο και στο επίπεδο του χρήστη. Ουσιαστικά, στην 1^{η} ενότητα, αναλύεται ο κώδικα προγραμματισμού που έχει γραφτεί για το Arduino Mega 2560 πάνω στο οποίο βασίζεται η λειτουργία όλου του συστήματος. Στην συνέχεια, στην 2^{η} ενότητα παρέχονται οδηγίες προς τον χρήστη για το πώς θα πρέπει να χειριστεί σωστά και με ασφάλεια το σύστημα αυτομάτου ελέγχου για τον συντονισμό των χορδών.

Τέλος, συνοψίζεται το έργο που εκπονήθηκε, και καταγράφονται τα συμπεράσματα που προέκυψαν. Επίσης, αναφέρονται οι καινοτομίες που εμπεριέχονται στην συγκεκριμένη εργασία οι οποίες την καθιστούν ιδιαίτερη. Επιπλέον, αναφέρονται οι δυσκολίες, τα εμπόδια και οι αδυναμίες του συγκεκριμένου συστήματος. Επιπρόσθετα, αναφέρονται και κάποιες βελτιώσεις που θα μπορούσαν να πραγματοποιηθούν στο συγκεκριμένο σύστημα τόσο για την κιθάρα, όσο και για άλλα έγχορδα μουσικά όργανα.

1 ΚΕΦΑΛΑΙΟ 1ο : Θεωρητικό Μέρος

Σε αυτό το κεφάλαιο αναλύεται το αναγκαίο θεωρητικό υπόβαθρο πάνω στο οποίο στηρίζεται η συγκεκριμένη διπλωματική εργασία. Πιο συγκεκριμένα, παρουσιάζονται η ιστορία και τα είδη της κιθάρας και επεξηγείται η ανατομία της. Επιπλέον, αναφέρονται οι μουσικές ιδιότητες της κιθάρας και οι παράγονται που την επηρεάζουν. Ακόμα, παρουσιάζονται σχετικές εργασίες οι οποίες μελετήθηκαν και αποτέλεσαν την βάση για την υλοποίηση της συγκεκριμένης διπλωματικής εργασίας. Τέλος, αναλύεται η τρισδιάστατη εκτύπωση, τα ηλεκτρονικά κυκλώματα και τα ηλεκτρονικά εξαρτήματα που χρησιμοποιήθηκαν για την υλοποίηση της συγκεκριμένης διπλωματικής εργασίας.
Εισαγωγική παράγραφος του κεφαλαίου.

1.1 Η ιστορία της κιθάρας και η ανατομία της

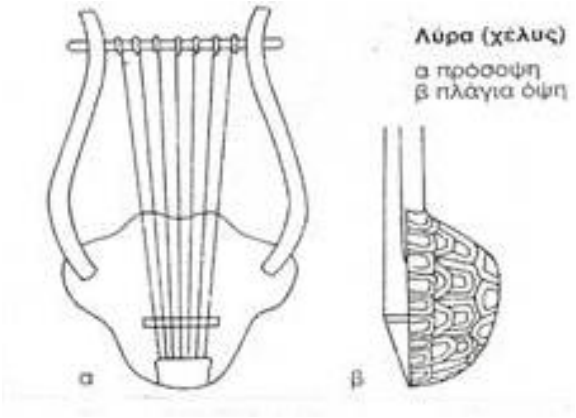
Η κιθάρα είναι ένα από τα πιο γνωστά μουσικά όργανα και ευρέως πιο χρησιμοποιούμενα όργανα και μπορεί να βρεθεί σε πάρα πολλά είδη μουσικής. Η κιθάρα κατατάσσεται στην κατηγορία των έγχορδων οργάνων και χωρίζεται κυρίως σε τρεις βασικές κατηγορίες κλασσική, ακουστική και ηλεκτρική διαφέροντας όχι μόνο οπτικά, αλλά και ακουστικά.

1.1.1 Ιστορική αναδρομή

Οι πρώτες αναφορές της κιθάρας εμφανίζονται στην αρχαία Ελλάδα η οποία ονομαζόταν κίθαριν σύμφωνα με μουσικοϊστορικές αναφορές από τα ομηρικά έπη [1]. Στα τέλη του 6ου αιώνα π.Χ. και στις αρχές του 7ου αιώνα π.Χ. υπήρχαν μουσικές μορφές –είδη τα οποία ονομάζονταν «νόμοι». Ουσιαστικά, οι νόμοι ήταν μελωδίες πάνω στις οποίες τραγουδούσαν ποιήματα στους θεούς. Ο λόγος που ονομάστηκαν «νόμοι» οφείλεται στο ότι απαγορευόταν αυστηρά η παρέκκλιση και η απομάκρυνση από τις θεμελιώδες αρχές τους. Υπήρχαν και οι κιθαρωδικοί νόμοι, δηλαδή μελωδίες που προέκυπταν από την συνοδεία κιθάρας, οι οποίοι θεσπίστηκαν από τον Τέρπανδρο.

Η λύρα και η κιθάρα είναι συγγενικά όργανα [2] [3] και αποτελούν δύο ιδιαίτερες ομάδες τύπου λύρας. Πιο συγκεκριμένα, η λύρα αποτελείται από δύο λεπτούς βραχίονες-πήχεις οι οποίοι προσαρμόζονται πάνω στο μικρό ηχείο το οποίο συνήθως αποτελείται από το καύκαλο χελώνας και γι αυτό αναφερόταν και ως χελύς ή χελώνη. Στο επάνω μέρος της λύρας υπάρχει ένας ζυγός πάνω στον οποίο τοποθετούνταν μια βέργα στην οποία δένονταν οι χορδές, ενώ η άλλη άκρη των χορδών δενόταν πάνω στο κυρίως σώμα της λύρας (Εικόνα 1.1).

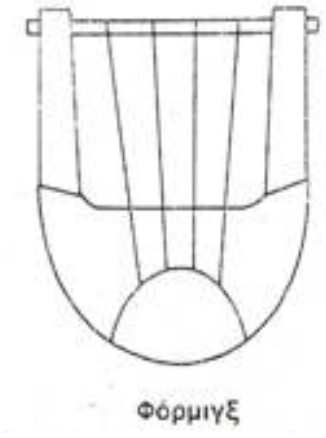
Από την άλλη, η κιθάρα έχει σχετικά μεγάλο ηχείο με χοντρούς βραχίονες-πήχεις, οι οποίοι είτε προσαρμόζονταν πάνω στο ηχείο είτε σχημάτιζαν ένα ενιαίο σώμα με αυτό. Πιο συγκεκριμένα, το σώμα της κιθάρας αποτελούταν από δύο ξύλινες επιφάνειες είτε επίπεδες είτε ελαφρώς κυρτές και ενωνόντουσαν με ένα φαρδύ ξύλινο περίγραμμα δημιουργώντας έτσι το αντηχείο. Στο επάνω μέρος της είχε έναν ζυγό πάνω στον οποίο τοποθετιόταν μια βέργα πάνω στην οποία δένοντας οι χορδές. Σε άλλες αναφορές [6] οι χορδές δένοντας πάνω σε έμβολα ή κρίκους. Η άλλη άκρη των χορδών δένονταν πάνω στο κυρίως σώμα του οργάνου και κάποιες φορές περνούσαν πάνω από μία επίπεδη γέφυρα (Εικόνα 1.2).



Εικόνα 1.1: Η μορφή της λύρα στην αρχαία Ελλάδα [48].

Εικόνα 1.2: Η μορφή της κιθάρα στην αρχαία Ελλάδα [49].

Επίσης ένα ακόμα συγγενικό όργανο με την κιθάρα στην αρχαία Ελλάδα είναι ο φόρμιγξ (φόρμιγγα) ο οποίος κατασκευαστικά μοιάζει με την κιθάρα, δηλαδή ένα μεγάλο ηχείο και δύο χοντρούς βραχίονες-πήχεις οι οποίοι είτε σχηματίζουν ενιαίο σώμα με το ηχείο είτε είναι προσαρμοσμένοι σε αυτό [4]. Όπως και η κιθάρα έτσι και ο φόρμιγξ είχε έναν ζυγό πάνω στον οποίο υπήρχε μια βέργα στην οποία δένονταν η μία άκρη των χορδών, ενώ η άλλη άκρη των χορδών δενόταν πάνω στο κυρίως σώμα του οργάνου (Εικόνα 1.3).



Εικόνα 1.3: Η μορφή του φόρμιγξ (φόρμιγγα) στην αρχαία Ελλάδα [50].

Ο φόρμιγξ ή αλλιώς φόρμιγγά συνήθως είχε 4 χορδές, αλλά υπήρχαν και περιπτώσεις που είχε 2, 3, 5 ή και 6 χορδές και παιζόταν κυρίως με πλήκτρο ένα είδος πένας για να διεγείρει ο μουσικός τις χορδές και να παράγεται έτσι ήχος. Από την άλλη, η λύρα παίζεται και αυτή με πλήκτρο και στην αρχή είχε τέσσερις χορδές. Στην συνέχεια, αναφέρεται πως ο Αμφίωνας πρόσθεσε ακόμα τρεις χορδές στην λύρα κάνοντάς ‘τες στο σύνολο επτά [5]. Στην ελληνική μυθολογία αναφέρεται πως ο Αμφίωνας και ο αδερφός του ο Ζήθος συντέλεσαν στο χτίσιμο των τειχών της Θήβας, πιο συγκεκριμένα ο Ζήθος με την δύναμή του μετέφερε τις πέτρες και ο Αμφίωνας με την μουσική του τις συνταίριαζε. Τα τείχη της Θήβας μετά την ολοκλήρωσή τους είχαν επτά πύλες που συμβόλιζαν τις επτά χορδές της λύρας του Αμφίωνα. Εν’ συνεχεία, αναφέρεται μέσα από αγγεία της εποχής, ότι η κιθάρα είχε αρχικά επτά χορδές, ενώ μέσα από αναφορές οι επιδέξιοι κιθαρωδοί χρησιμοποιούσαν περισσότερες χορδές. Άλλες πηγές αναφέρουν ότι ο Ορφέας ήταν αυτός που επινόησε την κιθάρα και αύξησε τις χορδές από επτά σε εννιά. Η κιθάρα παιζόταν και αυτή με πλήκτρο το οποίο το κρατούσε ο κιθαρωδός στο δεξί του χέρι και με το αριστερό χέρι κρατούσε τις επιθυμητές χορδές ώστε να σιγούν κάθε φορά που διέγειρε όλες τις χορδές μαζί.

Η σύγχρονη εκδοχή της κιθάρας εμφανίστηκε τον 15^{ο} αιώνα στην Ισπανία η οποία λεγόταν Vihuela (βιχουέλα), ήταν μικρότερες σε μέγεθος και αποτελούνταν από τέσσερα ζεύγη χορδών, ενώ τον 16^{ο} αιώνα αποτελούνταν από πέντε ζεύγη χορδών (Εικόνα 1.4) [7] [8]. Στις αρχές, οι χορδές που χρησιμοποιούνταν στις ισπανικές κιθάρες αποτελούνταν από έντερα αγελάδας και λίγο αργότερα από νεύρα διαφόρων ζώων. Βέβαια στην εποχή μας οι χορδές αυτές αντικαταστάθηκαν με νάιλον και ατσάλινες χορδές. Οι Ισπανικές κιθάρες, ή αλλιώς κλασσικές κιθάρες τον 18^{ο} αιώνα εμφανίστηκαν με έξι μονές χορδές αντί για ζεύγη χορδών . Από τα μέσα έως το τέλος του 19^{ου} αιώνα ο κατασκευαστής κιθάρων Manual Torres ανέπτυξε την κλασσική κιθάρα με την μορφή που ξέρουμε σήμερα τοποθετώντας μηχανικά κλειδιά και τοποθετώντας μεγαλύτερο ηχηρό σώμα δηλαδή μεγαλύτερο ηχείο (Εικόνα 1.5).

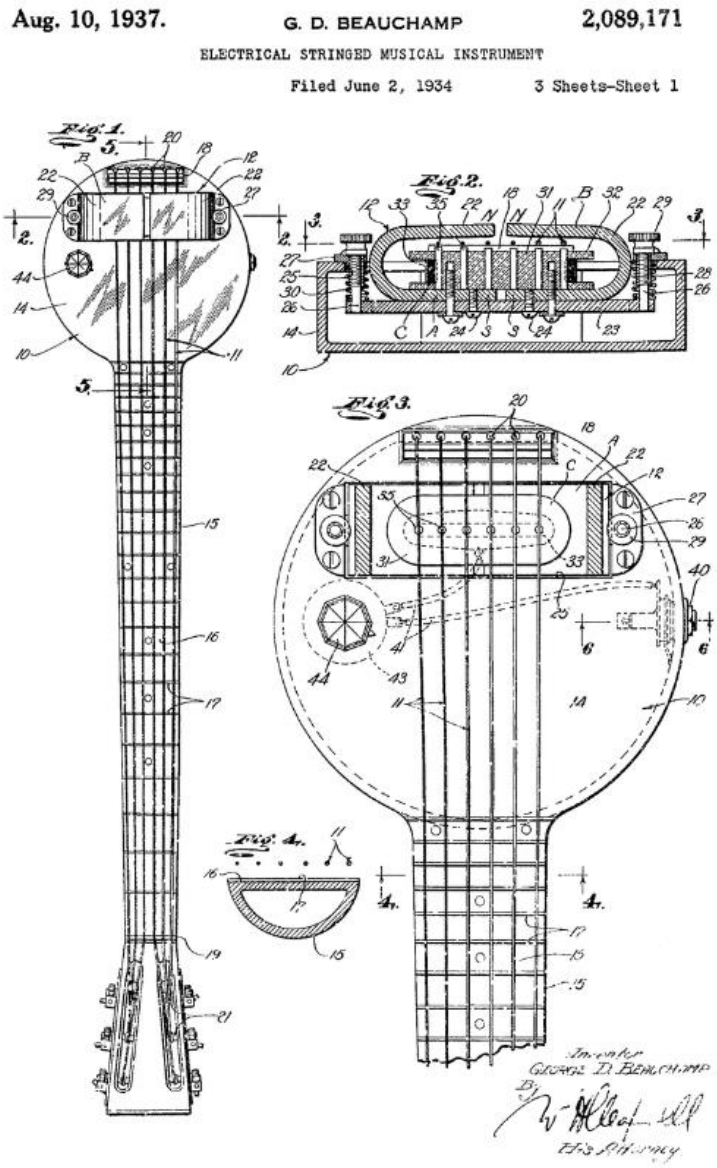
Η εξέλιξη της κλασσικής κιθάρας ξεκινά την δεκαετία του 1930 όπου κάνει την εμφάνισή της η ηλεκτρική κιθάρα, όπου με την χρήση του ηλεκτρισμού παράγεται πιο ενισχυμένος ήχος [3]. Σύμφωνα με πηγές, υπήρχε ανάγκη για την δημιουργία κιθάρας με ενισχυμένη ένταση, διότι οι τότε τζαζ ορχήστρες είχαν αυξήσει σε αριθμό κυρίως τα χάλκινα πνευστά με αποτέλεσμα να κάποια όργανα όπως οι κιθάρες να μην έχουν την δυνατότητα να ακουστούν. Ο πρώτος που πειραματίστηκε ώστε να αυξήσει την ένταση της κιθάρας ήταν ο Lester William Polsfuss ή αλλιώς γνωστός ως Les Paul ο οποίος

πειραματίστηκε προσαρμόζοντας μικρόφωνα σε κιθάρες της εποχής του. Ο Les Paul ήταν ένας Αμερικανός κιθαρίστας και τραγουδοποιός ο οποίος έπαιζε τζαζ, κάουντρι και μπλουζ και μέσα από τους πειραματισμούς του, του δόθηκε και ο «τίτλος» του εφευρέτη [9] [10] [11] [12].

Εικόνα 1.4: Η Vihuela τον 15ο και 16ο αιώνα [51].

Εικόνα 1.5: Η σύγχρονη μορφή της κλασσικής κιθάρας [52].

Οι πρώτες καθαρά ηλεκτρικές κιθάρες εμφανίστηκαν το 1932 από την εταιρία Electro String Instrument Corporation (πρώην Ro-Pat-In Electro Patent Instruments) στο Los Santos, οι οποίες είχαν κούφιο σώμα, ατσάλινες χορδές και ηλεκτρομαγνήτες από νήμα βολφραμίου οι οποίοι έμοιαζαν με πέταλο αλόγου και ονομάζονταν Rickenbacker A-22 lap steel ή αλλιώς τηγάνι (frying pan) (Εικόνα 1.6) και δημιουργήθηκε από τον τεχνίτη της εταιρίας Harry Watson. Η παλιότερη καταγεγραμμένη εμφάνιση ηλεκτρικής κιθάρας σύμφωνα με της πηγές φαίνεται να ήταν το 1932 στην Wichita του Kansas όπου ο Gage Brewer πρόβαλε δύο τύπους κιθάρας, την Electric Hawaiian A-25 και την Electric Spanish [9] [11]. Άλλες πηγές, αναφέρουν ότι η πρώτη ηλεκτροακουστική κιθάρα Rickenbacker που κατασκευάστηκε το 1931 χρησιμοποιούσε κάψα για να μπορέσει να γίνει «σύλληψη» του ήχου και εν συνεχεία ενίσχυση αυτού [13] [14] [15] [16].

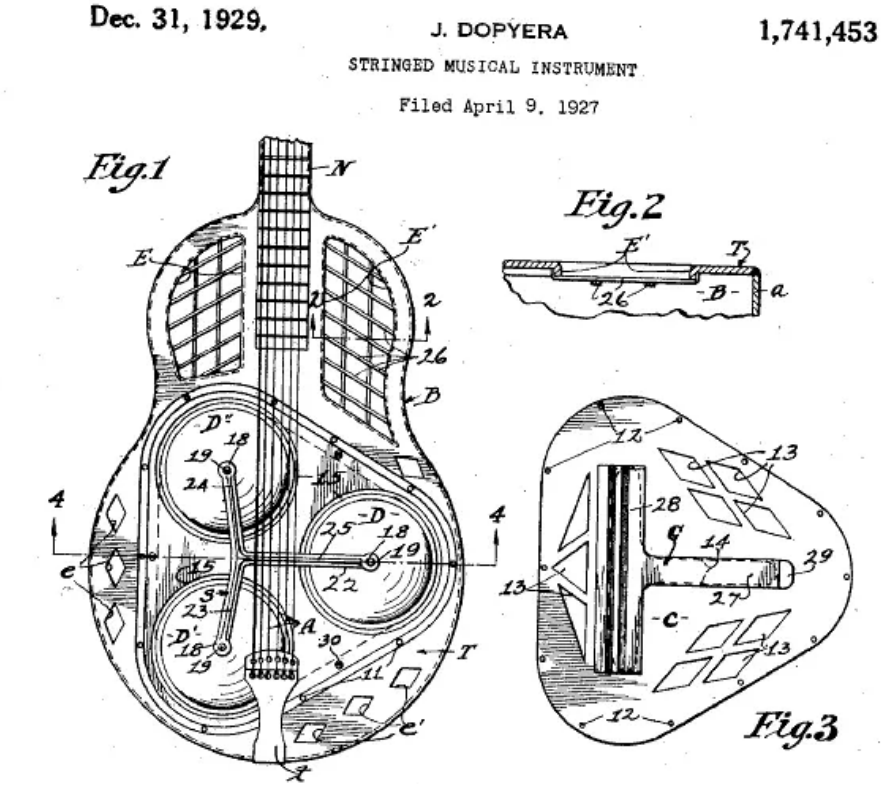


Εικόνα 1.6: Σχέδια της Rickenbacker A-22 lap-steel ή αλλιώς τηγάνι (frying pan) [53].

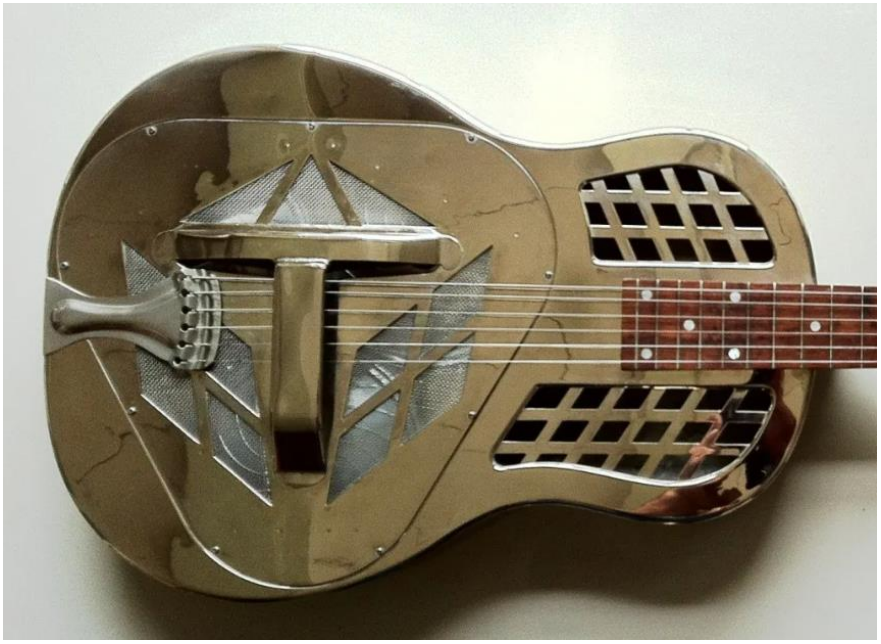
Η έννοια lap-steel προέρχεται από την Χαβάη όπου οι ντόπιοι έπαιζαν ακουστική κιθάρα η οποία ήταν ξαπλωμένη στα γόνατά τους και οι χορδές κοίταζαν προς τα επάνω. Η τεχνική που χρησιμοποιούσαν ήταν να διεγείρουν τις χορδές με το ένα χέρι, ενώ με το άλλο έσερναν μία μεταλλική μπάρα πάνω στο λαιμό της κιθάρας έτσι ο ήχος ακουγόταν σαν γλίστρημα από έναν συνδυασμό νοτών σε έναν άλλον. Αυτή η τεχνική παιξίματος ονομαζόταν Hawaiian style. Επειδή αυτή η μουσική ήταν πολύ δημοφιλής εκείνο τον καιρό και επειδή σε μεγάλο ακροατήριο ήταν δύσκολο να ακουστεί το συγκεκριμένο όργανο, γι αυτό φτιάχτηκε η Rickenbacker A-22 (Εικόνα 1.7) η οποία παιζόταν από τους μουσικούς με τον τρόπο που προαναφέρθηκε [17].

Εικόνα 1.7: Η Rickenbacker A-22 ή αλλιώς τηγάνι (frying pan) [54].

Σύμφωνα με πηγές, η πρώτη κατασκευάστρια εταιρία ηλεκτρικής κιθάρας, όπως αναφέρθηκε ήταν η Electro String Instrument Corporation το 1932, η οποία μετονομάστηκε σε Rickenbacker το 1934 ως φόρος τιμής στον Adolph Rickenbacker και ανέπτυξε την Rickenbacker A-22 ή αλλιώς τηγάνι (frying pan) ή αλλιώς Ro-Pat-In (κάποιες πηγές την αναφέρουν με αυτό το όνομα εξαιτίας της προηγούμενης ονομασίας της εταιρίας electRO-PATent-INstruments) [15]. Για την ιστορία όμως, πριν από την Rickenbacker υπήρχαν κι άλλες εταιρίες που έφτιαχναν κιθάρες όχι όμως ηλεκτρικές, αλλά ακουστικές. Μία από αυτές τις εταιρίες ήταν National String Instrument Corporation η οποία ιδρύθηκε το 1927 και έφτιαχνε αρχικά μπάντζο και στην συνέχεια κιθάρες, γιουκαλίλι και μαντολίνα [18]. Η εταιρία δημιουργήθηκε από τον George Beachamp (μετέπιπτα ιδρυτής της Electro String Instrument Corporation) και από τον John Dopyera [16]. Οι κιθάρες της National String Instrument Corporation είχαν μεταλλικό σώμα και βασιζόντουσαν σε ένα τρικονικό σύστημα αντήχησης, δηλαδή τρία αλουμινένιους κώνους σε σχήμα «Τ» (Εικόνα 1.8 & Εικόνα 1.9) [21]. Το 1928, ο John Dopyera μετά την αποτυχία του να πείσει τους συνεργάτες του να δημιουργήσουν κιθάρες με έναν κώνο, φεύγει από την National String Instrument Corporation και μαζί με τα τέσσερα αδέρφια του ιδρύει την Dobro Manufacturing Company ώστε να δημιουργήσει ένα ανταγωνιστικό σχεδιασμό αντηχείου [19]. Πάραυτα, συνέχισε να διατηρεί μετοχές στην National String Instrument Corporation. Οι κιθάρες Dobro (DOpyera BROthers) είχαν τον κώνο του αντηχείου ανεστραμμένο και ήταν πιο προσιτό οικονομικά αλλά και παρήγαγε πιο δυνατό ήχο από την κιθάρα Tri-Cone. Σε μικρό χρονικό διάστημα, η National δημιούργησε την δικιά της κιθάρα σχεδιασμένη με μονό αντηχείο γνωστή και ως «Biscuit», όπου υπήρχε ένας κώνος στην κορυφή του οποίου υπήρχε η γέφυρα ή αλλιώς καβαλάρης (Εικόνα 1.10, Εικόνα 1.11 και Εικόνα 1.12) [18] [21]. Ο John Dopyera ισχυρίστηκε ότι την είχε σχεδιάσει πριν φύγει από την εταιρία National, αλλά το δίπλωμα ευρεσιτεχνίας κατοχυρώθηκε από τον George Beauchamp.



Εικόνα 1.8: Σχέδια της μεταλλικής ακουστικής κιθάρας με τρικονικό σύστημα αντήχησης της National String Instruments Corporation [55].

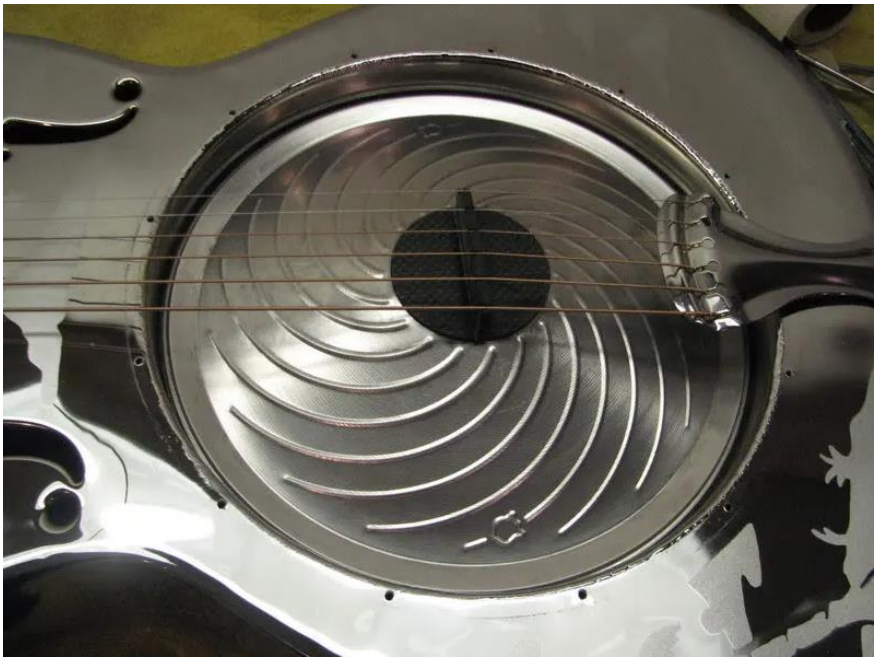


Εικόνα 1.9: National Tri-cone Guitar [56].

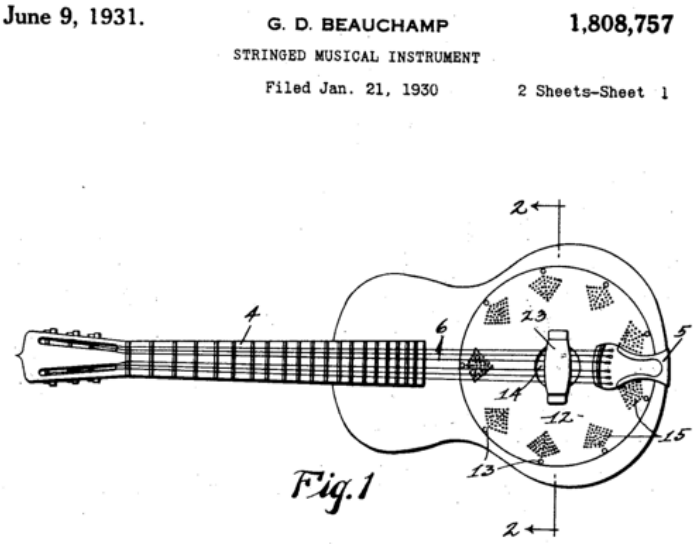
Το 1930 στην National String Instrument Corporation υπήρχαν οκτώ βασικοί συνεργάτες μέσα στους οποίους συμπεριλαμβάνονται οι Adolph Rickenbacker, George Beauchamp, Jack Levy, Harry Watson και Paul Barth. Το 1932 όμως οι αδερφοί Dopyera εξασφάλισαν τον έλεγχο τόσο της National String Instrument Corporation όσο και της Dobro Manufacturing Company και μετά την συγχώνευση των δύο εταιριών σχηματίστηκε η National Dobro Corporation [18]. Η κιθάρα Dobro αποτελεί το τρίτο σχέδιο του Dopyera, αλλά ουσιαστικά το δεύτερο που μπήκε σε παραγωγή μίας και το δεύτερο σχέδιο το

παρήγαγε η National String Instrument Corporation αφότου είχε φύγει ο Dopyera από την εταιρία.

Εικόνα 1.10: National single resonator guitar ή αλλιώς National Biscuit Guitar [57].



Εικόνα 1.11: Το εσωτερικό της National single resonator guitar ή αλλιώς National Biscuit Guitar το οποίο μοιάζει με μπισκότο [58]


June 9, 1931. G. D. BEAUCHAMP 1,808,757

STRINGED MUSICAL INSTRUMENT

Filed Jan. 21, 1930 2 Sheets-Sheet 1

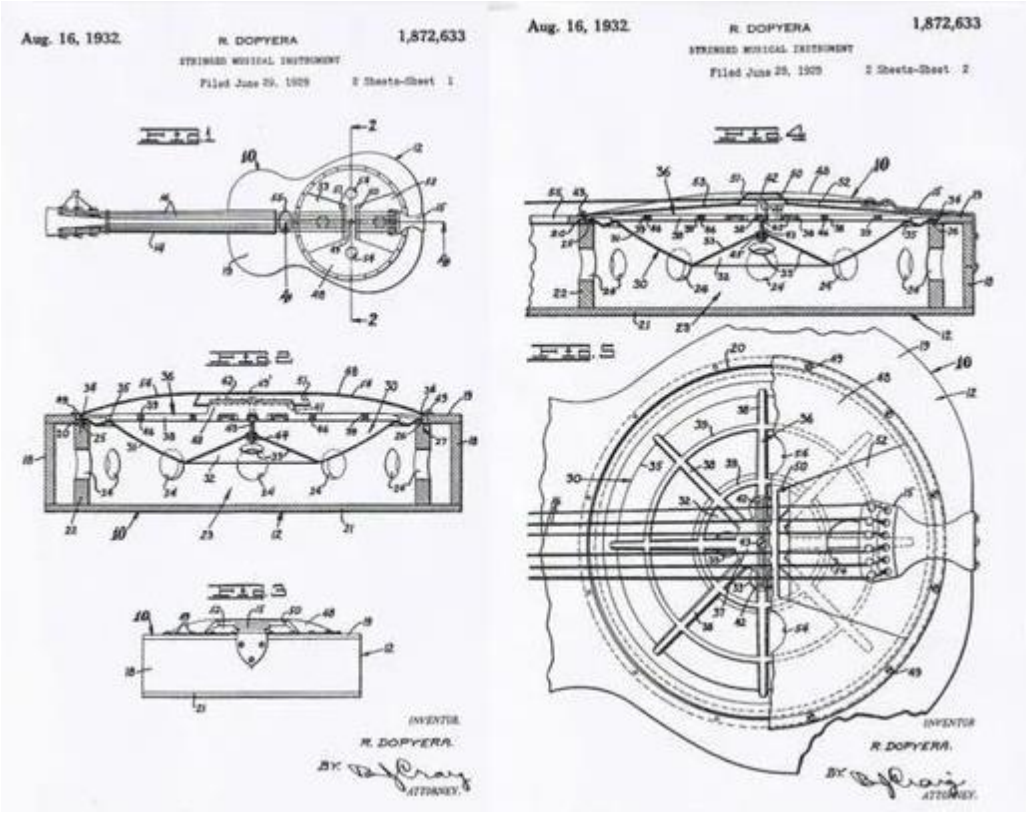
Fig. 1

Εικόνα 1.12: Το σχέδιο της National Single resonator guitar (National Biscuit Guitar)[59].

Η Dobro, αποτελείτο από ένα μονό αντηχείο το οποίο ήταν στραμμένο προς τα έξω, ενώ η επιφανειακή του κοιλότητα ήταν στραμμένη προς τα πάνω. Ουσιαστικά, το αντηχείο της κιθάρας Dobro έμοιαζε με μπολ (bowl-shaped) και η γέφυρα ή αλλιώς καβαλάρης τοποθετείται πάνω σε μία βάση χυτού αλουμινίου η οποία στηρίζεται στην περίμετρο του κώνου και μοιάζει με μία αράχνη που έχει οκτώ πόδια (Spider-shape resonator) (Εικόνα 1.13, Εικόνα 1.14 και Εικόνα 1.15) [20] [21] [22].

Εικόνα 1.13: Το αντηχείο της κιθάρας Dobro με την «αραχνοειδές» (Spider-shape) βάση της γέφυρας [60].

Εικόνα 1.14: Η κιθάρα Dobro [61].



Εικόνα 1.15: Τα σχέδια της κιθάρας Dobro [62].

Μία άλλη κατασκευάστρια εταιρία ηλεκτρικών κιθάρων ήταν η Audio Vox Manufacturing Co, η οποία δημιουργήθηκε από τον μουσικό και εφευρέτη Paul Tutmac το 1934 [20]. Ο Paul Tutmac, στις αρχές τις δεκαετίας του 30’ άρχισε να πειραματίζεται με τον ηλεκτρισμό και πιο συγκεκριμένα, μαζί με τον Arthur “Art” J. Stimson. Πιο συγκεκριμένα, εμπνευσμένοι από τον μηχανισμό του τηλεφώνου, τύλιξαν μία σιδερένια λεπίδα με σύρμα χαλκού γύρω της και την σύνδεσαν σε έναν μεγάλο μαγνήτη ο οποίος

είχε σχήμα πετάλου. Έπειτα, τοποθέτησαν τον μαγνήτη στην κιθάρα του Tutmac και την σύνδεσαν σε ένα τροποποιημένο ραδιόφωνο ώστε να ενισχυθεί ο ήχος του οργάνου. Το αποτέλεσμα ήταν να παραχθεί ένας ενισχυμένος και όμορφος ήχος. Ο Tutmac προσπάθησε να προστατέψει την ιδέα γι' τον μαγνήτη μέσω του γραφείου διπλωμάτων ευρεσιτεχνίας, αλλά μετά από επένδυση 300$ σε δικηγόρους όπου μαζί με την κυβέρνηση ερευνούσαν για διπλώματα ευρεσιτεχνίας, η ιδέα του απορρίφθηκε διότι οι εταιρίες τηλεφωνίας είχαν ήδη κατοχυρώσει παρόμοιες συσκευές. Μετά από λίγο καιρό, ο συνεργάτης του Tutmac, ο Arthur "Art" J. Stimson, πούλησε το σχέδιο του μαγνήτη στην εταιρία Dobro για 600$. Ο Tutmac το έμαθε αυτό μέσω της αίτησης που είχε γίνει από την Dobro για δίπλωμα ευρεσιτεχνίας η οποία ανέφερε ως εκχωρητή τον Art Stipson. Με αυτό τον τρόπο, η εταιρία Dobro ξεκίνησε να παράγει ηλεκτρικές κιθάρες ισπανικού στυλ το 1933. Μετά από αυτό το γεγονός ο Tutmac άρχισε να κατασκευάζει και να πουλά την δική του μάρκα ηλεκτρικής κιθάρας. Αποφασισμένος πια να δημιουργήσει μια ηλεκτρική κιθάρας ανώτερης κλάσης και μέσα από τα πειράματα που πραγματοποίησε δημιούργησε τον πρώτο στον κόσμο humbucking μαγνήτη. Το σχέδιο του humbucket μαγνήτη άρχισαν να το μιμούνται εταιρίες όπως η Dobro, η National και πολλές άλλες με την πάροδο του χρόνου. Το 1934 ο Tutmac επισημοποίησε την επιχείρησή του ως Audiovox Manufacturing Company και με την πάροδο του χρόνου οι ηλεκτρικές κιθάρες του απέκτησαν τεράστια φήμη για τον δυνατό και καθαρό ήχο. Ωστόσο, εμπορευόταν μαζί με της ηλεκτρικές κιθάρες του και του συνοδευτικούς ενισχυτές του. Το 1935, ο Tutmac παρουσίασε την τελευταία του εφεύρεση η οποία ήταν ένα ηλεκτρικό μπάσο με συμπαγές σώμα σε μέγεθος τσέλου το οποίο το ονόμασε Audiovox 736 (Εικόνα 1.16 & Εικόνα 1.17) [23] [24] [25].

Εικόνα 1.16: Ο Paul Tutmac σε φωτογραφία με το μπάσο Audiovox 736 και τις ηλεκτρικές κιθάρες Audiovox [63].

Εικόνα 1.17: Το μπάσο Audiovox 736 [64].

Η εταιρία Volu-tone το 1933, σχεδίασε ένα σετ μαγνήτη και ενισχυτή για χρήση σε οποιαδήποτε υπάρχουσα κιθάρα. Ο μαγνήτης της Volu-tone ουσιαστικά έπαιρνε τις δονήσεις του οργάνου με μία μέθοδο η οποία ήταν θανατηφόρα. Πιο συγκεκριμένα, ο μαγνήτης έπρεπε να φορτιστεί με ένα απότομο ρεύμα υψηλής τάσης που παρείχε ο ενισχυτής μέσω μιας ειδικής υποδοχής στο σασί του. Σύμφωνα με τις οδηγίες, το βύσμα της κιθάρας έπρεπε να βγει σχεδόν αμέσως από την υποδοχή ενεργοποίησης. Με αυτό τον τρόπο ενεργοποιούνταν οι χορδές και η διαδικασία θα έπρεπε να επαναληφθεί κάθε φορά που αλλαζόταν κάποια από της χορδές. Δεν θα έπρεπε όμως το βύσμα της κιθάρας να μένει για πάνω από ένα με δύο δευτερόλεπτα στην υποδοχή ενεργοποίησης διότι θα μπορούσε να προκληθεί βλάβη στο όργανο. Μετά από την διαδικασία αυτή, το βύσμα της κιθάρας συνδεόταν σε άλλη υποδοχή του ενισχυτή και ο μουσικός ήταν έτοιμος να παίξει. Η κατασκευή της υποδοχής ενεργοποίησης που διέθετε ο ενισχυτής δεν ήταν η ασφαλέστερη, μιας και το κέλυφος της υποδοχής ήταν μεταλλικό. Ακόμα, το καλώδιο με το οποίο συνδεόταν ο μαγνήτης της κιθάρας με τον ενισχυτή είχε τέσσερις ακροδέκτες ώστε να εξασφαλιστεί ότι δεν μπορεί να ταιριάξει σε κανέναν άλλο μαγνήτη. Με αυτό τον τρόπο οποιοσδήποτε άλλος μαγνήτης δεν θα μπορούσε να δεχτεί 300+ volt για να λειτουργήσει. Ωστόσο και το καλώδιο που συνδεόταν ο ενισχυτής με τον μαγνήτη δεν ήταν το ασφαλέστερο, μιας και το βύσμα το οποίο συνδεόταν στον μαγνήτη ήταν αρσενικό! Αυτό είχε ως αποτέλεσμα σε περίπτωση που συνδεόταν πρώτα το καλώδιο στον ενισχυτή, οι ακροδέκτες του αρσενικού βύσματος στην άλλη άκρη να είχαν υψηλή τάση [26]!

Μία άλλη εταιρία η οποία ξεκίνησε να παράγει ηλεκτρικές κιθάρες το 1935 ήταν η Vega Company [26]. Πιο συγκεκριμένα, Vega Company ξεκίνησε στην Βοστώνη της Μασαχουσέτης από τα αδέρφια Julius και Carl Nelson και από μία ομάδα συνεργατών

στους οποίους εμπεριέχονταν ο John Pahn και ο John Swenson. Τα αδέρφια Nelson πριν ιδρύσουν την δικιά τους εταιρία εργάζονταν σε άλλη εταιρία κατασκευής μουσικών οργάνων ως επιστάτες κατασκευής κιθάρων και μαντολίνων. Η εταιρία Vega Company κατασκεύαζε έγχορδα όργανα, κιθάρες, μαντολίνα και διάφορα είδη μπάντζο, καθώς και χάλκινα όργανα όπως τρομπέτες, κόρνες και σάλπιγγες. Το 1930, η Vega Company κατασκεύαζε κιθάρες τύπου archtop, όπου η πλάτη και η κεφαλή της κιθάρας ήταν σκαλισμένα με τον τρόπο των βιολιών (Εικόνα 1.18) [27]. Αυτού του τύπου οι κιθάρες έγιναν ευρέως γνωστές στους μουσικούς της τζαζ και η εταιρία παρήγαγε και την ηλεκτρική εκδοχή της συγκεκριμένης κιθάρας το 1935. Ωστόσο, η Vega Company παρήγαγε μία κιθάρα η οποία είχε μια διαμήκης διόγκωση κατά μήκος της πλάτης της αλλά και της κορυφής της [28].

Εικόνα 1.18: Η ακουστική κιθάρα τύπου Arch Top της εταιρίας Vega Company το 1938 Vegaphone C-75 [65].

Μία άλλη εταιρία που παρήγαγε ηλεκτρικές κιθάρες το 1935, η οποία είναι γνωστή μέχρι και σήμερα, είναι η Epiphone [29] [30]. Η ιστορία της Epiphone ξεκινάει από το 1873 στην Σμύρνη, όπου ο Αναστάσιος Σταθόπουλος ως έμπορος ξυλείας επισκεύαζε και κατασκεύαζε επίσης βιολιά, λαούτα και ούτι. Το 1903, μεταναστεύει στην Αμερική όπου συνεχίζει να κατασκευάζει τα ίδια όργανα, ενώ το 1909 ξεκίνησε και την κατασκευή μαντολίνων στο εργοστάσιό του που βρισκόταν στο Κουίνς (Queens) της Νέας Υόρκης (New York). Το 1915 όμως μετά τον θάνατο του Αναστάσιου Σταθόπουλου, αναλαμβάνει ο γιός του Επαμεινώνδας ο οποίος είχε το παρατσούκλι Έπι (Epi) και η εταιρία γίνεται γνωστή ως το Σπίτι του Σταθόπουλου (The House of Stathopoulo). Μετά το τέλος του 1^{ου} παγκοσμίου πολέμου, η εταιρία ξεκίνησε να κατασκευάζει μπάντζο. Το 1923 μετά τον θάνατο της μητέρας του ο Επαμεινώνδας αναλαμβάνει τον πλήρη έλεγχο της εταιρίας, και το 1928 μετονομάζει την εταιρία σε Epiphone Banjo Company. Το ίδιο έτος, η εταιρία

ξεκίνησε να παράγει τις πρώτες της ακουστικές κιθάρες και πιο συγκεκριμένα τα μοντέλα Epiphone Recording (Εικόνα 1.19) [31].

Εικόνα 1.19: Η Epiphone Recording του 1928 [66].

Το 1931 η Epihone παράγει τις κιθάρες Masterbilt (Εικόνα 1.20), οι οποίες έγιναν πασίγνωστες και ανταγωνίστηκαν την αντίστοιχη γκάμα της Gibson (L-5) [32] . Το 1935, η εταιρία Epiphone ξεκίνησε να παράγει τις δικές της ηλεκτρικές κιθάρες και πιο συγκεκριμένα τα μοντέλα Electar Electrophone (Εικόνα 1.21) [33]. Επίσης, η Epiphone το 1935 ξεκίνησε παράγει και τους δικούς της ενισχυτές για να υποστηρίξουν τις ηλεκτρικές κιθάρες της. Το 1939, ο μουσικός και εφευρέτης Les Paul περνούσε κάποια βράδια στο εργοστάσιο της Epiphone κατασκευάζοντας ένα δικό του μουσικό όργανο. Πιο συγκεκριμένα, πήρε ένα σώμα Archtop της Epiphone το οποίο το είχε κόψει στην μέση και το γέμισε με μασίφ ξύλο πάνω στο οποίο είχε τοποθετήσει τα ηλεκτρονικά την γέφυρα και τους μαγνήτες, ενώ ο λαιμός της συγκεκριμένης κιθάρας ήταν από μία άλλη κιθάρα της εταιρίας Gibson. Αυτό το υβρίδιο που κατασκεύασε ο Les Paul αποτέλεσε ένα μεγάλο βήμα για την εξέλιξη της ηλεκτρικής κιθάρας. Ωστόσο, ο Les Paul πούλησε την ιδέα του για μία συμπαγής ηλεκτρική κιθάρα στην εταιρία Gibson [29]. Μετά τον θάνατο του Επαμεινώνδα Σταθόπουλου (Epi) το 1943, αναλαμβάνουν το έλεγχο της εταιρίας τα αδέρφια του Ορφή και Φρίξος. Το 1951 εξαιτίας μιας τετράμηνης απεργίας τα κεντρικά γραφεία της Epiphone μεταφέρονται στην Φιλαδέλφεια. Το 1957 όμως η εταιρία Epiphone εξαγοράζεται από τον μεγαλύτερο ανταγωνιστή της την εταιρία Gibson και συγχωνεύτηκαν. Ουσιαστικά, η εταιρία Gibson θα πρόσφερε όργανα ισοδύναμης ποιότητας μόνο που θα έφεραν την ετικέτα Epiphone οι οποίες θα μπορούσαν να πουληθούν από άλλες αντιπροσωπίες αυξάνοντας έτσι τις συνολικές αγορές. Όμως το 1969, η Gibson έπαψε να βγάζει προϊόντα υψηλής ποιότητας υπό την ετικέτα USA Epiphone και ξεκίνησε να τις παράγει στην Ιαπωνία ώστε να είναι χαμηλότερης ποιότητας και χαμηλότερης τιμής. Την δεκαετία του ’70 η Epiphone ξεκίνησε να παράγεται στην Κορέα όπου μέχρι τα μέσα της δεκαετίας του ’80 όλες οι Epiphone παραγόντουσαν στην

Κορέα. Τέλος, από το 1997 μέχρι και σήμερα, οι Epiphone παράγονται στην Κίνα και την Ινδονησία [30].

Εικόνα 1.20: Η Epiphone Masterbilt του 1931 [67].

Εικόνα 1.21: Η Epiphone Electar του 1935 [68].

Στην σημερινή εποχή ωστόσο μία από τις πιο διαδεδομένες κατασκευάστριες εταιρίες κιθάρων είναι η Gibson. Πιο συγκεκριμένα, ο Ovrille Gibson άρχισε να κατασκευάζει μουσικά όργανα από το 1894 κυρίως μαντολίνα και κιθάρες χρησιμοποιώντας ένα καινούργιο για την εποχή στυλ. Πιο συγκεκριμένα, οι κεφαλές των οργάνων δεν ήταν λυγισμένες, αλλά σκαλισμένες και τοξωτές όπως η κεφαλή ενός βιολιού. Το 1898, ο Ovrille Gibson κατασκεύασε το πιο ανθεκτικό μαντολίνο της εποχής, το οποίο αποτελείτο από ένα μονοκόμματο ξύλο και όχι από λυγισμένες λεπτές λωρίδες

και είχε σκαλισμένη κεφαλή και πλάτη. Πιο συγκεκριμένα, με το μονοκόμματο ξύλο ήταν πιο εύκολο να κατασκευαστεί ένα μαντολίνο δίνοντάς του όγκο. Ωστόσο, την δεκαετία του 1890, κατασκεύασε κιθάρες με σκαλιστές κεφαλές και με οβάλ οπή στο ηχείο. Αυτό είχε ως αποτέλεσμα να αυξηθεί η ένταση του ήχου που παρήγαγε το όργανο σε σχέση με αντίστοιχα όργανα της εποχής και να τεθούν τα πρότυπα για το μέλλον της κιθάρας τύπου Archtop. Το 1902 ίδρυσε την εταιρία Gibson Mandolin-Guitar Mfg. Co. Ltd στο Καλαμαζού του Μίσιγκαν, με την χρηματοδότηση πέντε επιχειρηματιών. Ο Gibson συνέχισε να συνεργάζεται με την εταιρία μέχρι το 1907 και μεγάλο μέρος των εσόδων του προερχόντουσαν από τα δικαιώματα των πατεντών του, ενώ ταυτόχρονα εργαζόταν ως εφευρέτης και ως δάσκαλος μουσικής. Επίσης, το 1907, ο Ovrille Gibson κατασκεύασε ένα υβρίδιο ακουστικής κιθάρας και άρπας (Εικόνα 1.22) αποδεικνύοντας έτσι την τεχνικές του ικανότητες και τις πρωτοποριακές ιδέες για την εποχή του [35] [36].

Εικόνα 1.22: Η κιθάρα-άρπα του Gibson το 1907 [69].

Το 1908 η εταιρία καταβάλει στον Gibson ετήσια αμοιβή των 500$, ενώ η πληρωμή των 2500$ για την αποκλειστικότητα των δικαιωμάτων του διπλώματος ευρεσιτεχνίας του έγινε σε δόσεις των 41.99$ των μήνα. Στις 19 Αυγούστου του 1918, ο Ovrille Gibson κλείνει τα μάτια του από ενδοκαρδίτιδα σε ηλικία 62 ετών [35]. Λίγο πριν τον θάνατό του ο Gibson σχεδίασε μία ακουστική κιθάρα με πρωτοποριακό σχήμα για την εποχή, την Style O Acoustic Archtop (Εικόνα 1.23) [36]. Το επόμενο έτος η εταιρία Gibson προσλαμβάνει τον Lloyd Loar για να σχεδιάσει νέα μουσικά όργανα όπως την ακουστική κιθάρα L-5 archtop το 1922 (Εικόνα 1.24) η οποία αποτελούσε την ναυαρχίδα της εταιρίας και το μαντολίνο Gibson F-5, ενώ το 1924 έφυγε από την εταιρία [34] [35]. Το 1936, η εταιρία Gibson κάνει ένα μεγάλο βήμα κατασκευάζοντας την πρώτη της ηλεκτρική κιθάρα (Electric Spanish Guitar) και πιο συγκεκριμένα το μοντέλο ES-150 (Εικόνα 1.25) η οποία παραγόταν μέχρι το 1957. Ουσιαστικά, η Gibson ES-150 μοιάζει με την Gibson L-5 με την μόνη διαφορά ότι έχει προσαρτηθεί ένα μαγνήτης και τα απαραίτητα ηλεκτρονικά [36].

Εικόνα 1.23: Η Gibson Style O Acoustic Archtop Guitar του 1918 [70].

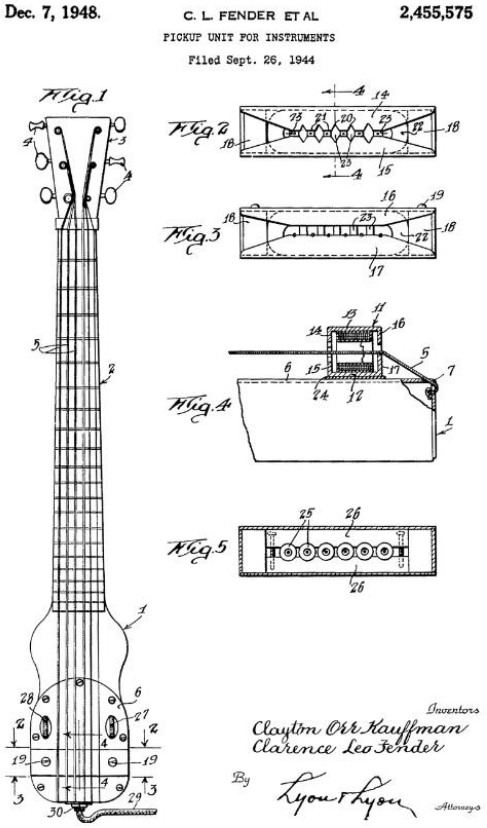
Εικόνα 1.24: Η ναυαρχίδα της Gibson, η L-5 Archtop Guitar του 1922-1933 [71].

Εικόνα 1.25: Η Gibson ES-150 (Electric Spanish) 1936-1957 [72].

Η μορφή και ο τύπος της ηλεκτρικής κιθάρας που είναι γνωστός μέχρι και σήμερα έχει συμπαγής μορφή και δημιουργήθηκε από τον μουσικοσυνθέτη και εφευρέτη Les Paul

στα μέσα της δεκαετίας του 1940 κατά την διάρκεια των υπερωριών του στην εταιρία Epiphone Guitars. Αρχικά η κιθάρα αυτή ονομάστηκε κιθάρα κούτσουρο (log guitar) και τα κομμάτια που την αποτελούσαν ήταν ένα απλό τετράγωνο ξύλινο σώμα με δύο κούφιες διακοσμητικές τομές από σουηδικό ξύλο , το μπράτσο, τους χειροποίητους μαγνήτες και τα ηλεκτρικά συστήματα. Το μοντέλο του συμπαγούς σώματος κατοχυρώθηκε και αναφέρεται ως το πρώτο του είδους του. Πάραυτα, το μοντέλο που σχεδίασε ο Les Paul, δεν έχει καμία ομοιότητα τόσο ως προς τα υλικά, όσο και ως προς το σχεδιασμό με την κιθάρα Les Paul που έφτιαξε αργότερα η εταιρία Gibson. Στα χρόνια που ακολούθησαν μέχρι και σήμερα, η Gibson σχεδίασε και εξέλιξε αρκετά μοντέλα [35] [37].

Μία εξίσου μεγάλη και πασίγνωστη εταιρία κατασκευής κιθάρων μέχρι και σήμερα είναι η Fender. Αρχικά, η εταιρία ιδρύθηκε το 1938 με το όνομα Fender's Radio Service στο Φούλερτον της Καλιφόρνιας και ο ιδρυτής της ήταν ο Clarence Leonidas "Leo" Fender. Ο Fender επισκεύαζε ραδιόφωνα, φωτογραφικές μηχανές, ενισχυτές οικιακών ήχο-συστημάτων, συστήματα αναγγελιών και ενισχυτές μουσικών οργάνων. Στην συνέχεια, το 1943 ο Leonidas "Leo" Fender συνεργάζεται με τον μουσικό και εφευρέτη Clayton Orr "Doc" Kauffman και ιδρύουν την εταιρία K & F Manufacturing Corp η οποία κατασκευάζει ηλεκτρικά όργανα και ενισχυτές. Η παραγωγή ξεκινά το 1945 κατασκευάζοντας Χαβανέζικες lap steel κιθάρες με ενσωματωμένο μαγνήτη (Εικόνα 1.26) όπως η Champion. Επίσης κατασκεύαζαν ενισχυτές τους οποίους του πουλούσαν ως σετ με τις ηλεκτρικές κιθάρες. Ο Clarence Leonidas "Leo" Fender αντιλήφθηκε και πείστηκε ότι η κατασκευή οργάνων και ενισχυτών ήταν πιο κερδοφόρα από ότι η επισκευή και επικεντρώθηκε σε αυτό. Από την άλλη ο Clayton Orr "Doc" Kauffman έμεινε αμετάπειστος και στις αρχές του 1946 οι δρόμοι του χώρισαν φιλικά. Τότε ο Leo Fender μετονόμασε την εταιρία σε Fender Electric Instrument Company. Ωστόσο, το κατάστημα συντήρησης έμεινε ανοιχτό μέχρι το 1951, παρόλο που ο Leo Fender δεν το επέβλεπε προσωπικά από το 1947. Το 1946, ο Leo Fender κατασκεύασε την πρώτη συμπαγής ηλεκτρική κιθάρα με έναν ηλεκτρομαγνήτη η οποία ονομάστηκε Fender Esquire (Εικόνα 1.27) [43] και αποτέλεσε το πρώτο κατασκευαστικό πέρασμα προς την σύγχρονη εκδοχή της ηλεκτρικής κιθάρας όπως την ξέρουμε σήμερα. Ουσιαστικά οι κιθάρες που έχουν συμπαγές σώμα (solid body) είναι λιγότερο επιρρεπής στην ανατροφοδότηση του θορύβου από το περιβάλλον και μπορούν να παιχτούν πιο δυνατά [38] [39] [40].



Εικόνα 1.26: Το σχέδιο της ηλεκτρικής κιθάρας lap steel του Leo Fender το 1944 [73].

Εικόνα 1.27: Η Fender Esquier του 1946 [74].

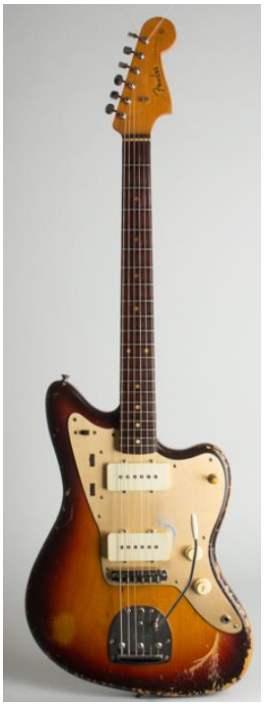
Η αντίστοιχη εκδοχή της Fender Esquire με δύο ηλεκτρομαγνήτες ονομάστηκε αρχικά Broadcaster οι οποίες όμως είχαν προβλήματα, καθώς σε καιρό με υγρασία άρχισαν να στραβώνουν. Ο Leo Fender αντιμετώπισε το πρόβλημα αυτό τοποθετώντας μια μεταλλική ράβδο εντός του μπράτσου του οργάνου. Μετά την διόρθωση του οργάνου, ο Leo Fender άλλαξε το όνομα του οργάνου από Broadcaster σε Fender Telecaster εξαιτίας της ομοηχίας με το σετ ντραμς Broadcaster [38]. Η μαζική παραγωγή της Fender Telecaster (Εικόνα

1.28) ξεκίνησε το 1950 ενώ τον ίδιο χρόνο εισήγαγε την νέα του εφεύρεση το πρώτο μπάσο της Fender Precision Bass (ή αλλιώς P Bass) μαζί με τον πρώτο ενισχυτή Bassman [41].

Εικόνα 1.28: Η Fender Broadcaster και μετέπειτα Telecaster του 1950 [75].

Τον Αύγουστο του 1954, ο Leo Fender αποκάλυψε το νέο μοντέλο της Fender και πιο συγκεκριμένα την πασίγνωστη σε όλους του κιθαρίστες την Fender Stratocaster (Εικόνα 1.29) [41]. Ουσιαστικά, η Fender Stratocaster διαθέτει ένα περίγραμμα διπλής κοπής, τρεις μαγνήτες μονού πηνίου και ένα συγχρονισμένο σύστημα vibrato με το οποίο ο κιθαρίστας μπορούσε να χαλαρώσει τις χορδές πιο εύκολα για να κάνει εφέ. Δηλαδή, πέτυχε τον ήχο που μοιάζει με ατσάλινο πεντάλ (σαν ολίσθηση του ήχου) τεχνική που χρησιμοποιούσαν πολύ οι καλλιτέχνες του είδους Country. Εν συνεχεία, το 1959 ο Leo Fender ολοκλήρωσε ένα νέο μοντέλο την Fender Jazzmaster (Εικόνα 1.30) η οποία αποσκοπούσε στους κιθαρίστες της Jazz μουσικής [45]. Η Fender Jazzmaster ήταν διαφορετική από τα προηγούμενα σχέδια κιθάρας, καθώς η αντιστάθμιση του σώματος, το σύστημα vibrato που διέθετε και τα καινοτόμα ηλεκτρονικά που σχεδιάστηκαν αποσκοπούσαν στην κατάκτηση της αγοράς των Jazz κιθάρων όπου μέχρι τότε κυριαρχούσαν οι ακουστικές κιθάρες [38].

Εικόνα 1.29: Η Fender Stratocaster του 1954 [76].



Εικόνα 1.30: Η Fender Jazzmaster του 1959 [77].

Ωστόσο το 1960 ο Leo Fender ολοκλήρωσε την κατασκευή του μπάσου Fender Jazz Bass, ενώ το 1962 εμφάνισε την κιθάρα Fender Jaguar (Εικόνα 1.31) [38] [44]. Τον Ιανουάριο του 1965 ο Leo Fender πουλάει τις επιχειρήσεις του στην εταιρία Columbia Broadcasting System (CBS) η οποία με αυτόν τον τρόπο μπήκε στην αγορά των μουσικών οργάνων. Το 1985 ξεκίνησε μια εσωτερική εκστρατεία από τον πρόεδρο του τμήματος μουσικών οργάνων της CBS η οποία είχε ως αποτέλεσμα οι υπάλληλοι της Fender Electric Instruments Manufacturing Company να αγοράσουν την εταιρία από την CBS και να την μετονομάσουν σε Fender Musical Instruments Corporation (FMIC). Ωστόσο, η αγορά από την CBS δεν περιελάμβανε το παλιό εργοστάσιο στο Fullerton και έτσι η FMIC έπρεπε να

χτίσει μια νέα εγκατάσταση στην Corona της Καλιφόρνια. Η πλειοψηφία των κιθάρων Fender που πουλήθηκαν το 1985 ήταν κατασκευασμένες στην Ιαπωνία. Τέλος, το 1991, η Fender Musical Instrument Corporation μετέφερε την έδρα της στο Scottsdale της Αριζόνα ενώ το 2016 μετέφερε την έδρα της στην καρδιά του Los Angeles της Καλιφόρνια στο Hollywood [39] .

Εικόνα 1.31: Η Fender Jaguar του 1962 [78].

Τα επόμενα χρόνια που ακολούθησαν οι εταιρίες που προαναφέρθηκαν και πολλές ακόμα που δημιουργήθηκαν στην πορεία σχεδίασαν και κατασκεύασαν κιθάρες κλασσικές, ακουστικές και ηλεκτρικές σε διάφορα σχέδια. Το επόμενο μεγάλο βήμα στην σύγχρονη εκδοχή της κιθάρας, το οποίο αποτέλεσε έμπνευση για την συγκεκριμένη διπλωματική εργασία, έρχεται να το κάνει η εταιρία Gibson, η οποία το Δεκέμβριο του 2007 δημιουργεί την Gibson Robot Guitar. Ουσιαστικά, η Gibson Robot Guitar ήταν μία κιθάρα Les Paul πάνω στην οποία είχε προσαρτηθεί ένας μηχανισμός με το αυτόματο σύστημα ελέγχου για το κούρδισμα των χορδών της κιθάρας. Εν συνεχεία, το σύστημα αυτό σχεδιάστηκε για να τοποθετηθεί και σε άλλα μοντέλα της Gibson όπως τις κιθάρες SG, Flying-V X-plorer, Les Paul Studio και Les Paul Junior. Το σύστημα αυτομάτου ελέγχου για το κούρδισμα των χορδών αναπτύχθηκε από τον Chris Adams το οποίο αποτελείται από έναν ενσωματωμένο μικροϋπολογιστή ο οποίος είναι υπεύθυνος για το κούρδισμα των χορδών. Τα αντίστοιχα συστήματα αυτομάτου ελέγχου για το κούρδισμα των χορδών στις κιθάρες της Gibson αναπτύχθηκαν από την Tronical Company στην Γερμανία και ονομάζονται Powertune Systems (Εικόνα 1.32). Πιο συγκεκριμένα, στις Gibson Robot Guitars έχει γίνει μια τροποποίηση στις γέφυρες τύπου Tune-o-matic στις οποίες έχουν τοποθετηθεί πιεζοσέλες (σέλες πιεζοκρυστάλλου) (Εικόνα 1.33) ώστε να αντιλαμβάνονται τις δονήσεις της κάθε χορδής και να τις μεταδίδουν στον μικροεπεξεργαστή [49]. Στην συνέχεια ο μικροεπεξεργαστή, αναλύει το σήμα και ελέγχει τα αντίστοιχα κλειδιά για να κουρδίσει τις χορδές, τα οποία ονομάζονται Powerhead Locking Tuners. Ουσιαστικά, στο εσωτερικό το κλειδιών αυτών περιλαμβάνονται ένας

μικρός σερβοκινητήρας και ένα σύστημα από γρανάζια τα οποία περιστρέφουν το κάθε κλειδί κατά τέτοιον τρόπο αλλάζοντάς του την τάση και κατά συνέπεια τον τόνο-νότα της αντίστοιχης χορδής ώστε να επιτευχθεί το επιθυμητό αποτέλεσμα. Επίσης, το σύστημα τροφοδοτείται από μία μπαταρία νικελίου υδριδίου μετάλλου η οποία είναι επαναφορτιζόμενη [46] [47] [49] [51] [52].

Εικόνα 1.32: Το Power Tune Systems της Tronical Company στις Gibson Robot Guitars [79].

Παρόλο όμως που η εταιρία Gibson παρουσίαζε την κιθάρα αυτή ως την πρώτη του είδους της παγκοσμίως, παρόμοια συστήματα είχαν αναπτυχθεί εδώ και δεκαετίες από την Transperformance. Το μειονέκτημα όμως του συστήματος Trasperformance (Εικόνα 1.34) είναι ότι απαιτεί πολλές τροποποιήσεις πάνω στην κιθάρα. Από την άλλη το πλεονέκτημα είναι ότι μπορεί να αποθηκεύσει 240 διαφορετικά κουρδίσματα που μπορεί να δημιουργήσει ο κιθαρίστας. Οποιοδήποτε κούρδισμα μπορεί να επιλεχθεί είτε μέσω της πρόσοψης της κιθάρας είτε με την χρήση κάποιας πεταλιέρας στην οποία η επιλογή γίνεται πατώντας τα αντίστοιχα κουπιά με το πόδι. Επίσης, διατίθεται μία οθόνη ανάγνωσης LCD τοποθετημένη στο επάνω μέρος της κιθάρας στο οποίο προβάλετε το τρέχων κούρδισμα και τα στατιστικά του συστήματος [50].

Εικόνα 1.33: Η γέφυρα τύπου Tune-o-matic με τις πιεζοσέλες οι οποίες μεταδίδουν το σήμα στον μικροεπεξεργαστή [80].

Εικόνα 1.34: Το σύστημα Transperformance self-tuning guitar [81].

Το Powertune System της Tronical Company είναι πιο εύχρηστο και πιο απλό σε σχέση με το Transperformance self-tuning guitar το οποίο απευθύνεται σε μία ελίτ κατηγορία. Ουσιαστικά, το Powertune System περιλαμβάνει έναν επιλογέα με τον οποίο ενεργοποιώντας τον και διεγείροντας απαλά τις χορδές το σύστημα ξεκινάει να κουρδίζει τις χορδές. Ωστόσο περιλαμβάνει και μερικά άλλα κουρδίσματα εκτός του βασικού, αλλά δεν δίνει την δυνατότητα στον χρήστη να αποθηκεύσει κάποιο δικό του κούρδισμα. Πάραυτα, η Tronical Company δεν έμεινε μόνο στο Powertune System, αλλά έβγαλε ένα άλλο μοντέλο μετά από κάποια χρόνια με περισσότερες δυνατότητες το οποίο το ονόμασε G-Force (Εικόνα 1.35) [48]. Το G-Force, χρησιμοποιεί και αυτό γέφυρα τύπου Tune-o-

Matic με σέλες πιεζοκρυστάλλου οι οποίες μεταδίδουν το σήμα στον μικροεπεξεργαστή. Επίσης, το G-Force, δίνει την δυνατότητα να κουρδιστούν οι χορδές μία προς μία ή όλες μαζί σχεδόν ταυτόχρονα, παρέχει μια ποικιλία από τα πιο δημοφιλή κουρδίσματα, δίνει στον χρήστη την δυνατότητα να αποθηκεύσει τα δικά του κουρδίσματα καθώς και να χειριστεί χειροκίνητα τα κλειδιά ώστε να αλλάξει τις χορδές. Τέλος, η Tronical Company έβγαλε το G-Force όχι μόνο για τις κιθάρες της Gibson, αλλά και για τις Epiphone, τις Fender, τις Jackson, τις Ibanez και για πολλές άλλες. Η μόνη διαφορά είναι ότι η Gibson και η Epiphone, που είναι ουσιαστικά ίδιες εταιρίες, κάποια μοντέλα τους το έχουν ήδη πάνω στην κιθάρα, ενώ στις υπόλοιπες μάρκες θα πρέπει ο χρήστης να αγοράσει το αντίστοιχο G-Force από την στιγμή που το επιθυμεί από την Tronical Company και το τοποθετήσει εξτρά πάνω στην κιθάρα του [51].

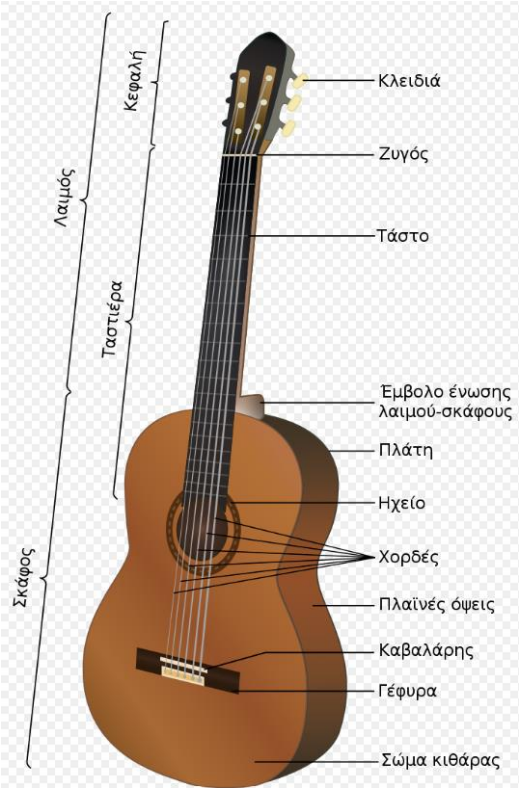
Εικόνα 1.35: Το G-Force tuning system της Tronical company [82].

1.1.2 Η ανατομία της κιθάρας

Σε αυτό το κεφάλαιο γίνεται αναφορά στην ανατομία της κιθάρας, δηλαδή τα μέρη από τα οποία αποτελείται. Είναι σημαντικό να γνωρίζει κάποιος τα μέρη και το πώς λειτουργεί ένα σύστημα πριν σχεδιάσει το οτιδήποτε πάνω σε αυτό, γι' αυτό τον λόγο γίνεται αυτή η αναφορά.

Αρχικά, η κιθάρα χωρίζεται σε δύο βασικά μέρη, το σκάφος ή αλλιώς το κυρίως σώμα και τον λαιμό (Εικόνα 1.36 & Εικόνα 1.37). Πιο συγκεκριμένα, το σκάφος στην κλασσική και στην ακουστική κιθάρα αποτελείται από το κυρίως σώμα-καπάκι, τις πλαϊνές όψεις, την πλάτη και το ηχείο. Επίσης, πάνω στο κυρίως σώμα είναι προσαρμοσμένοι η γέφυρα και ο καβαλάρης όπου πάνω από τον καβαλάρη περνάνε οι χορδές και στην συνέχεια δένονται πάνω στην γέφυρα. Η γέφυρα και ο καβαλάρης είναι το σημείο κλειδί για την κιθάρα, διότι πάνω στον καβαλάρη τεντώνονται οι χορδές οι οποίες όταν διεγείρονται παράγουν δονήσεις όπου μέσω του καβαλάρη περνάνε στην γέφυρα και από εκεί στο σώμα της κιθάρας. Ουσιαστικά, τα τοιχώματα του οργάνου αποτελούν ένα αντηχείο, δηλαδή τα τοιχώματα συντονίζονται με τις δονήσεις και ενισχύεται έτσι ο ήχος. Με αυτόν τον τρόπο μέσα στο κούφιο σώμα της κιθάρας γίνεται ενίσχυση του ήχου με φυσικό τρόπο και στην συνέχεια ο ήχος βγαίνει από το όργανο από την τρύπα που βρίσκεται στο κυρίως σώμα, δηλαδή το ηχείο. Επίσης, το ηχείο τις περισσότερες φορές διακοσμείται γύρω του με διάφορα μοτίβα και σχέδια τα οποία ονομάζονται ροζέτες. Σημαντικό ρόλο στην κλασσική και στην ακουστική κιθάρα παίζει το είδος του ξύλου με

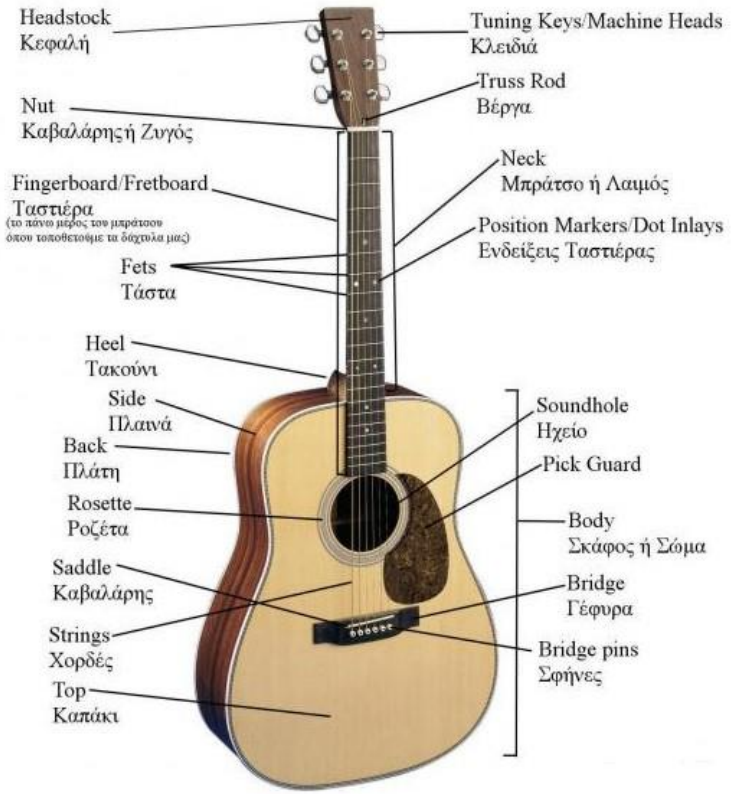
το οποίο έχει κατασκευαστεί, το σχέδιο και η ποιότητα κατασκευής. Αυτά τα χαρακτηριστικά επηρεάζουν την ποιότητα και την ένταση του ήχου που θα βγάλει το όργανο [3] [53].



Εικόνα 1.36: Η ανατομία της κλασσικής κιθάρας [83].

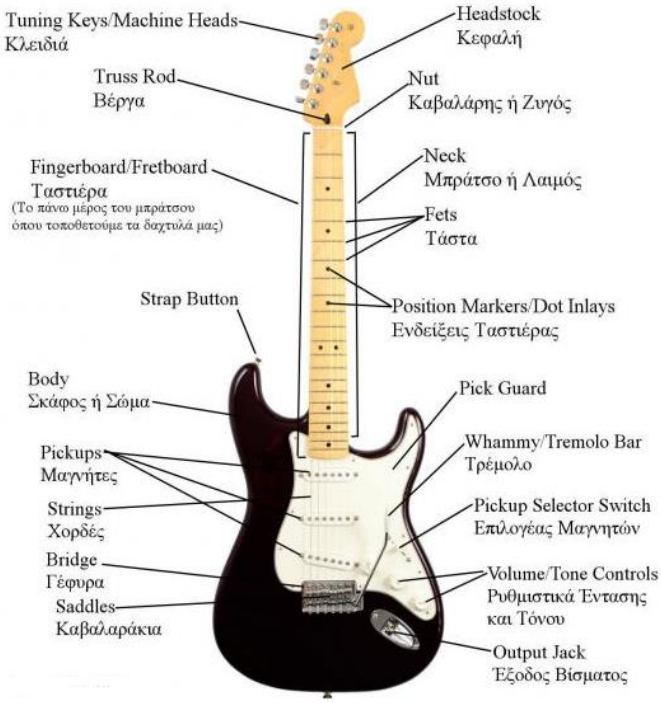
Από την άλλη, ο λαιμός ή αλλιώς μπράτσο της κιθάρας, είναι πιο λεπτό και πιο μακρύ σε σχέση με το κυρίως σώμα της κιθάρας και αποτελείται από την ταστιέρα, τον ζυγό και τα κλειδιά. Το μπράτσο στις κλασσικές και οι ακουστικές κιθάρες είναι ενσωματωμένο με το κυρίως σώμα μέσω ενός εμβόλου ένωσης. Το μπράτσο, έχει επίπεδη μορφή στην μπροστινή όψη της ταστιέρας και καμπυλωτή μορφή στο πίσω μέρος της. Στην επίπεδη όψη της ταστιέρας και σε όλο το μήκος της, υπάρχουν λεπτά σιδεράκια κάθετα στο μπράτσο τα λεγόμενα τάστα. Τα τάστα όσο πλησιάζον το κυρίως σώμα της κιθάρας τόσο μικρότερη είναι η απόσταση μεταξύ τους. Τα τάστα παίζουν σημαντικό ρόλο στην κιθάρα, διότι αλλάζουν το μήκος ταλάντωσης της χορδής το οποίο με την σειρά του μεταβάλει την νότα. Ουσιαστικά, όσο μικρότερο είναι το μήκος της χορδής που ταλαντώνεται τόσο πιο «ψιλή» ακούγεται η νότα. Η καμπυλωτή μορφή στο πίσω μέρος του μπράτσου αποσκοπεί στην στήριξη και στην τεχνική του χεριού με το οποίοι πατιούνται οι χορδές. Στην κορυφή της ταστιέρας βρίσκεται ο ζυγός ο οποίος αποτελεί το δεύτερο σημείο τεντώματος των χορδών. Πάνω από τον ζυγό βρίσκεται η κεφαλή της κιθάρας η οποία περιλαμβάνει τα κλειδιά του οργάνου πάνω στα οποία δένονται οι χορδές. Σε κάθε κλειδί μπορεί να δεθεί μόνο μία χορδή. Τα κλειδιά αποτελούνται από έναν απλό μηχανισμό γραναζιών και περιστρέφοντάς τα είτε δεξιόστροφα είτε αριστερόστροφα η αντίστοιχη χορδή σφίγγει-τεντώνει ή ξεσφίγγεται- χαλαρώνει. Επίσης, και στο μπράτσο της κιθάρας παίζει σημαντικό ρόλο το ξύλο από το οποίο έχει κατασκευαστεί. Για την κατασκευή της ταστιέρας, συνήθως χρησιμοποιείτε ξύλο τριανταφυλλιάς το οποίο προσδίδει καλύτερη αίσθηση στο παίξιμο και αντέχει στις φθορές από τις τριβές των δακτύλων. Στο καμπυλωτό μέρος του μπράτσου και στην κεφαλή, χρησιμοποιούνται ξύλα

τα οποία χαρακτηρίζονται για την αντοχή τους, μιας και εξαιτίας των χορδών μπορεί να εμφανιστεί σκέβρωμα στο μπράτσο με αποτέλεσμα να χαλάει ο ήχος της κιθάρας και να κάνει το παίξιμο πιο δύσκολο. Επίσης, οι περισσότερες ακουστικές κιθάρες στο εσωτερικό του μπράτσου τους έχουν ενσωματωμένη μια σιδερένια ράβδο. Η ράβδος αυτή με το κατάλληλο κλειδί ρυθμίζει την καμπυλότητά της και επομένως και την καμπυλότητα του μπράτσου επαναφέροντας τυχόν σκέβρωμα [53].



Εικόνα 1.37: Η ανατομία της ακουστικής κιθάρας [84].

Οι ηλεκτρικές κιθάρες όπως και οι κλασσικές και οι ακουστικές, αποτελούνται και αυτές από ένα σώμα-σκάφος και από ένα μπράτσο-λαιμό (Εικόνα 1.38). Η μόνη διαφορά είναι ότι οι ηλεκτρικές κιθάρες έχουν πιο συμπαγές σώμα με αποτέλεσμα να είναι πιο λεπτοκομμένες. Επίσης, το συμπαγές σώμα προστατεύει τους μαγνήτες, τα ποτενσιόμετρα με τα οποία ρυθμίζεται η ένταση και ο τόνος του σήματος, καθώς και άλλα ηλεκτρονικά που μπορεί να υπάρχουν όπως ο επιλογέας μαγνητών και η υποδοχή του βύσματος. Πάνω στο σώμα της ηλεκτρικής κιθάρας, υπάρχει η γέφυρα πάνω στην οποία βρίσκονται μικροί καβαλάρηδες και πιο συγκεκριμένα ένας για κάθε χορδή. Οι καβαλάρηδες αποσκοπούν στην στήριξη, το τέντωμα και την ευθυγράμμιση των χορδών. Επίσης, κάποιες ηλεκτρικές κιθάρες έχουν συγκεκριμένος είδος γέφυρας η οποία είναι κατασκευασμένη για να δουλεύει με μπάρα τρέμολο. Επιπλέον, σε κάποιες ηλεκτρικές, όπως και σε κάποιες ακουστικές κιθάρες υπάρχει ένα πλαστικό προστατευτικό πάνω στο σώμα το οποίο σκοπός του είναι να προστατεύει το σώμα της κιθάρας από τυχόν γρατζουνίσματα της πένας. Παρόλο που η ηλεκτρική κιθάρα χρησιμοποιεί μαγνήτες για την «σύλληψη» του ήχου, η ποιότητα του ξύλου παίζει σημαντικό ρόλο διότι επηρεάζεται ο τρόπος με τον οποίο δονείτε όλο το όργανο και κατά συνέπεια ο παραγόμενος ήχος [3][53].



Εικόνα 1.38: Η ανατομία της ηλεκτρικής κιθάρας [85].

Το μπράτσο της ηλεκτρικής κιθάρας, όπως και στην κλασσική και την ακουστική κιθάρας, αποτελείται από την ταστιέρα και από την κεφαλή. Πιο συγκεκριμένα, το μπράτσο της ηλεκτρικής κιθάρας είναι πιο λεπτό σε σχέση με της κλασσικής και της ακουστικής κιθάρας. Επίσης, το μπράτσο μπορεί να είναι ένα με το συμπαγές σώμα της ηλεκτρικής κιθάρας ή αποσπώμενο. Η ταστιέρα είναι και αυτή επίπεδη από την μπροστινή της όψη, ενώ η πίσω πλευρά της είναι καμπυλωτή, ώστε να μπορεί ο κιθαρίστας να στηρίξει το χέρι με το οποίο πατάει τις χορδές και να χρησιμοποιήσει την σωστή τεχνική στην λαβή. Επίσης, η ταστιέρα και στην ηλεκτρική κιθάρα χωρίζεται με λεπτά σιδεράκια σχηματίζοντας τα τάστα. Συνήθως τα τάστα στις ηλεκτρικές κιθάρες έχουν μικρότερη απόσταση μεταξύ τους με αποτέλεσμα ο κιθαρίστας να μπορεί να τα πατάει με μεγαλύτερη ταχύτητα τα τάστα. Επιπροσθέτως, πάνω στην ταστιέρα υπάρχουν κάποιες κουκίδες ή διακοσμητικά τα οποία χρησιμεύουν στο να βρίσκει ο κιθαρίστας πιο γρήγορα το τάστο που θέλει να πατήσει. Ενδιάμεσα από την κεφαλή και από την ταστιέρα της ηλεκτρικής κιθάρας υπάρχει ο ζυγός ο οποίος αποτελεί το δεύτερο σημείο που ακουμπάνε και τεντώνονται οι χορδές της κιθάρας. Στην κεφαλή της κιθάρας υπάρχουν τα κλειδιά πάνω στα οποία δένονται οι χορδές ώστε να μπορεί ο κιθαρίστας περιστρέφοντάς τα να τις κουρδίζει κατάλληλα. Ακόμα, όλες οι ηλεκτρικές κιθάρες στο μπράτσο του έχουν ενσωματωμένη μία σιδερένια ράβδο η οποία με κλειδί (συνήθως τύπου allen) ρυθμίζεται η καμπυλότητά της και κατά συνέπεια η καμπυλότητα του μπράτσου σε περίπτωση που το όργανο έχει υποστεί κάποιο σκέβρωμα. . Επιπλέον, τα ξύλα που θα χρησιμοποιηθούν στο μπράτσο της κιθάρας παίζουν και αυτά σημαντικό ρόλο στην κατασκευή της κιθάρας. Πιο συγκεκριμένα, και στις ηλεκτρικές κιθάρες, η ταστιέρα συνήθως κατασκευάζεται από ξύλο τριανταφυλλιάς το οποίο είναι ανθεκτικό στις φθορές από τις τριβές των δακτύλων ενώ παράλληλα προσφέρει καλύτερη αίσθηση στο παίξιμο. Στο υπόλοιπο μέρος του μπράτσου χρησιμοποιούνται ξύλα τα οποία διακρίνονται για την αντοχής τους μιας και αυτό δέχεται τις δυνάμεις από το τέντωμα των χορδών [3] [53].

Οι κιθάρες, πέρα από κλασσικές, ακουστικές και ηλεκτρικές, ταξινομούνται και ως προς το πλήθος των χορδών. Συνήθως οι κιθάρες είναι εξάχορδες δηλαδή έχουν έξι χορδές, ωστόσο υπάρχουν κιθάρες με επτά, οχτώ, δέκα, δώδεκα ακόμα και με δεκαοχτώ χορδές (Εικόνα 1.39, 1.40, 1.41, 1.42, 1.43, 1.44 & 1.45) [54] [55] [56] [57] [58] [59] [60]. Ουσιαστικά με μεγαλύτερο πλήθος χορδών, ο κιθαρίστας έχει μεγαλύτερο εύρος νοτών και ηχοχρωμάτων που μπορεί να χρησιμοποιήσει. Όμως για να τοποθετήσει ο κατασκευαστής περισσότερες χορδές στην κιθάρα θα πρέπει να αλλάξει και τον τρόπο που θα κατασκευαστεί. Πιο συγκεκριμένα, ο λαιμός της κιθάρας θα πρέπει να αποκτήσει μεγαλύτερο φάρδος τόσο ως προς την ταστιέρα όσο και ως προς το καμπυλωτό μέρος του. Επίσης, στην κεφαλή θα πρέπει να τοποθετηθούν περισσότερα κλειδιά ώστε να μπορέσουν να δεθούν όλες οι χορδές, επομένως η κεφαλή θα πρέπει να μεγαλώσει σε μέγεθος. Ακόμα ο ζυγός, ο καβαλάρης και η γέφυρα της κιθάρας θα πρέπει να μεγαλώσουν σε μέγεθος με ώστε να μπορέσουν να ακουμπήσουν όλες οι χορδές. Τέλος στις ηλεκτρικές κιθάρες θα πρέπει να αλλάξει και το μέγεθος στους μαγνήτες ώστε να μπορούν να συλλέξουν τον ήχο από όλες τις χορδές.

Εικόνα 1.39: Επτάχορδη κλασσική κιθάρα [86].

Εικόνα 1.40: Επτάχορδη ηλεκτρική κιθάρα (Gibson Les Paul Standard) [87].

Εικόνα 1.41: Οκτάχορδη ηλεκτρική κιθάρα (Ibanez RG5328) [88].

Εικόνα 1.42: Δεκάχορδη κλασσική κιθάρα [89].

Εικόνα 1.43: Δεκάχορδη ηλεκτρική κιθάρα (Halo Guitars Octavia 10-string) [90].

Εικόνα 1.44: Δωδεκάχορδη ακουστική κιθάρα (Gibson J-45 Standard) [91].

Εικόνα 1.45: Δεκαοχτάχορδη ηλεκτρική κιθάρα (Gibson EDS-1275 Double Neck) [92].

1.1.3 Τα κουρδίσματα της κιθάρας

Το σημείο κλειδί στην κιθάρα αλλά και γενικώς στα έγχορδα όργανα είναι το σωστό κούρδισμα του οργάνου. Ωστόσο, στην κιθάρα υπάρχουν αρκετοί τύποι κουρδίσματος που μπορεί να χρησιμοποιήσει ο κιθαρίστας για να πετύχει το επιθυμητό ηχόχρωμα. Επίσης, το κούρδισμα εξαρτάται και από το πλήθος των χορδών που έχει η κιθάρα, καθώς και από το πάχος τους. Επίσης, η κιθάρα είναι ένα μουσικό όργανο μετατόπισης, δηλαδή η τονικότητα της κιθάρας αποσκοπεί μία οκτάβα ψηλότερα από την πραγματική τονικότητα. Αυτό έχει ως αποτέλεσμα την ελάττωση των βοηθητικών γραμμών για τις νότες που γράφονται εκτός πενταγράμμου. Ακόμα όσον αφορά το πεντάγραμμο, οι νότες σχεδιάζονται πάνω σε γραμμή ή ανάμεσα σε δύο γραμμές δηλαδή σε κενό. Η μουσική απόσταση των νοτών μεταξύ ενός κενού και της αμέσως επόμενης ή προηγούμενης γραμμής ονομάζεται ημιτόνιο. Ωστόσο, στην συγκεκριμένη παράγραφο παρουσιάζονται κάποια από τα πιο γνωστά κουρδίσματα που χρησιμοποιούνται στην κιθάρα.

1.1.3.1 Το κλασσικό-Standard κούρδισμα της κιθάρας

Αρχικά, το «κλασσικό» ή αλλιώς Standard κούρδισμα στις εξάχορδες κιθάρες (ηλεκτρικές, ακουστικές & κλασσικές) είναι το E, A, D, G, B & e ή αλλιώς ΜΙ(μπάσο), ΛΑ, ΡΕ, ΣΟΛ, ΣΙ & ΜΙ(καντίνι) (Πίνακας 1.1). Η νότα E (Μι μπάσο) μπορεί να έχει την ίδια ονομασία με την e (Μι καντίνι), αλλά έχουν διαφορετικές συχνότητες. Για την ακρίβεια, η Μι καντίνι είναι πολλαπλάσιο της συχνότητας της Μι μπάσο και μάλιστα το τετραπλάσιο της συχνότητας της Μι Μπάσο [70]. Πιο συγκεκριμένα, η διπλάσια συχνότητα μίας νότας αποτελεί την ίδια νότα, αλλά μία οκτάβα πάνω, ενώ το τετραπλάσιο

της συχνότητας αποτελεί την ίδια νότα, αλλά δύο οκτάβες πάνω. Όσον αφορά το «κλασσικό»-Standard κούρδισμα της κιθάρας, είναι το πρώτο πράγμα το οποίο διδάσκεται κάποιος κιθαρίστας. Επίσης, σε αυτό το κούρδισμα στηρίζονται πάρα πολλά τραγούδια και για κάποιους κιθαρίστες είναι το μοναδικό το οποίο χρησιμοποιούν. Πιο συγκεκριμένα, το «κλασσικό»-Standard κούρδισμα παρέχει απλό δακτυλισμό για τις κλίμακες και τις συγχορδίες σε όλα τα κλειδιά ματζόρε και μινόρε [61] [64].

Πίνακας 1.1: Το κλασσικό-Standard κούρδισμα της κιθάρας [1].

Χορδή	Νότα	Συχνότητα (Hz)
6η	E	82.4 Hz
5η	A	110 Hz
4η	D	146.83 Hz
3η	G	196 Hz
2η	B	246.94 Hz
1η	e	329.63 Hz

1.1.3.2 Εναλλακτικά κουρδίσματα της κιθάρας για ανοιχτές χορδές

Αυτή η κατηγορία κουρδισμάτων είναι ιδιαίτερη διότι το κάθε ένα από αυτά τα κουρδίσματα αποσκοπεί σε μία συγχορδία είτε μινόρε είτε ματζόρε όταν διεγείρονται όλες οι χορδές μαζί χωρίς να πιέζει ο κιθαρίστας κάποιο τάστο. Μια σειρά από ματζόρε συγχορδίες είναι πιο χρήσιμες σε σχέση με μία σειρά από μινόρε συγχορδίες. Ωστόσο, όταν έχουμε μία σειρά από μινόρε συγχορδίες είναι πιο εύκολο να δημιουργηθούν ματζόρε συγχορδίες πατώντας κάποιο τάστο. Επιπλέον, σε αυτά τα κουρδίσματα, μπορεί να χρησιμοποιηθεί η τεχνική slide με την οποία ο κιθαρίστας μπορεί να ολισθήσει από την μία συγχορδία στην άλλη κάνοντας ηχητικό εφέ. Τα πιο γνωστά κουρδίσματα ανοιχτών χορδών είναι η ανοιχτή Ρε ματζόρε (Open D), η ανοιχτή Σολ ματζόρε (Open G), η ανοιχτή Ντο ματζόρε (Open C), η Ρε, Λα, Ρε, Σολ, Λα & Ρε (DADGAD), η ανοιχτή Μι ματζόρε (Open E) και για τους πιο ψαγμένους η ανοιχτή Φα9 (Open F9).

Πιο συγκεκριμένα, το κούρδισμα για την ανοιχτή Ρε ματζόρε (Open D) στην κιθάρα ξεκινώντας από την 6η χορδή πάνω-πάνω, είναι Ρε, Λα, Ρε, Φα#, Λα & Ρε (D, A, D, F#, A & D) το οποίο ονομάζεται αλλιώς και Vestapol (Πίνακας 1.2). Επίσης, το κούρδισμα αυτό χρησιμοποιείται κυρίως στην μουσική Bluegrass και βολεύει σε τεχνικές που απαιτούν γρήγορο παίξιμο. Επιπλέον, διεγείροντας τις ανοιχτές χορδές η συγχορδία που ακούγεται είναι η Ρε ματζόρε.

Από την άλλη, το κούρδισμα για την ανοιχτή Σολ ματζόρε (Open G) στην κιθάρα αποτελείται συνήθως από τις εξής νότες: Ρε, Σολ, Ρε, Σολ, Σι & Ρε (D, G, D,G,B & D) (Πίνακας 1.3). Ωστόσο, υπάρχουν και παραλλαγές του κουρδίσματος σε Σολ, Σολ, Ρε, Σολ, Σι & Ρε (G, G, D, G, B, D) ή σε Σολ, Ρε, Σολ, Σι & Ρε (G, D, G, B & D) για πεντάχορδη κιθάρα. Από την άλλη στις επτάχορδες ρώσικες κιθάρες, το κούρδισμά της σε ανοιχτή Σολ ματζόρε είναι Ρε, Σολ, Σι, Ρε, Σολ, Σι & Ρε (D, G, B, D, G, B & D) [62] [63].

Πίνακας 1.2: Το κούρδισμα της ανοιχτής Ρε (open D) στην κιθάρα (Vestapol Tuning) [2].

Χορδή	Νότα	Συχνότητα (Hz)
6η	D	73.42 Hz
5η	A	110 Hz
4η	D	146.83 Hz
3η	F#	185 Hz
2η	A	220 Hz
1η	D	293.66 Hz

Πίνακας 1.3: Το κούρδισμα της ανοιχτής Σολ (open G) στην κιθάρα [3].

Χορδή	Νότα	Συχνότητα (Hz)
6η	D	73.42 Hz
5η	G	98 Hz
4η	D	146.83 Hz
3η	G	196 Hz
2η	B	246.94 Hz
1η	D	293.66 Hz

Ένα άλλο συνηθισμένο εναλλακτικό κούρδισμα στην κιθάρα είναι το κούρδισμα σε ανοιχτή Ντο ματζόρε (Open C). Πιο συγκεκριμένα, αυτό το κούρδισμα χρησιμοποιείται ευρέως σε δωδεκάχορδές κιθάρες, αλλά μπορεί να χρησιμοποιηθεί εξίσου και σε εξάχορδες κιθάρες. Τυπικά, οι νότες από τις οποίες αποτελείται είναι Ντο, Σολ, Ντο, Σολ, Ντο & Μι (C, G,C, G, C & E) (Πίνακας 1.4). Επίσης, μία παραλλαγή του συγκεκριμένου κουρδίσματος περιλαμβάνει τις εξής νότες: Ντο, Σολ, Ντο, Σολ, Ντο & Ντο (C, G, C, G, C & C). Η επανάληψη των νοτών Ντο (C) και Σολ (G) κάνει ιδιαίτερα διαισθητικό και χαρακτηριστικό τον ήχο της κιθάρας και είναι κατάλληλο για βαριά Riffs με την συνοδεία κιθάρας η οποία είναι κουρδισμένη σε «πεσμένη» Λα (La-drop tuning). Ωστόσο, όταν διεγείρεται η ανοιχτή χορδή Ντο στο συγκεκριμένο κούρδισμα παράγεται μια σειρά από αρχικές αρμονικές η οποίες δίνουν ένα ιδιαίτερο άκουσμα στον ήχο [63].

Πίνακας 1.4: Το κούρδισμα της ανοιχτής Ντο (open C) στην κιθάρα [4].

Χορδή	Νότα	Συχνότητα (Hz)

6η	C	35.41 Hz
5η	G	98 Hz
4η	C	130.81 Hz
3η	G	196 Hz
2η	C	261.63 Hz
1η	E	329.63 Hz

Ένα επίσης ενδιαφέρον κούρδισμα ανοιχτών χορδών περιλαμβάνεται από τις εξής νότες Ρε, Λα, Ρε, Σολ, Λα & Ρε (D, A, D, G, A & D) (Πίνακας 1.5). Πιο συγκεκριμένα, διεγείροντας τις χορδές τις κιθάρας χωρίς να πατιέται κάποιο τάστο, η συγχορδία που ακούγεται είναι η Ρε αυξημένη 4ης (Dsus4). Σαφώς και σε αυτόν τον τύπο κουρδίσματος, μπορούν να εφαρμοστούν οι τεχνικές που χρησιμοποιούνται και στα άλλα ανοιχτά κουρδίσματα. Το κούρδισμα αυτό το χρησιμοποιούν οι κιθάρες κυρίως στην μουσική Celtic, καθώς και σε άλλα είδη όπως Folk, Rock, Metal και πολλά άλλα. Επίσης, χρησιμοποιώντας τις κλίμακες Ρε ματζόρε (D major) και Σι μινόρε (B minor) πάνω σε αυτό το κούρδισμα, ο ήχος που παράγεται είναι εξαιρετικός. Ο ήχος της κιθάρας γίνεται πολύ χαρακτηριστικός, εξαιτίας της φύσης της «Αυξημένης» που χρησιμοποιείται σε αυτό το κούρδισμα. Αυτό οφείλεται στο ότι όταν διεγείρονται χορδές στις οποίες δεν πατιέται κάποιο τάστο σε συνδυασμό με άλλες χορδές στις οποίες είτε πατιέται κάποιο τάστο είτε όχι, οι ταλαντώσεις που δημιουργούνται προσδίδουν έναν πλούσιο συντονισμό του οργάνου κάνοντας έτσι τον ήχο πιο συμπαθητικό στο αυτί [62] [63].

Πίνακας 1.5: Το κούρδισμα της Ρε αυξημένης 4ης (Dsus4) στην κιθάρα [5].

Χορδή	**Νότα**	**Συχνότητα (Hz)**
6η	D	73.42 Hz
5η	A	110 Hz
4η	D	146.83 Hz
3η	G	196 Hz
2η	A	220 Hz
1η	D	293.66 Hz

Εν συνεχεία, ένα άλλο ευρέως γνωστό εναλλακτικό κούρδισμα είναι αυτό της ανοιχτής Μι ματζόρε (Open E). Πιο συγκεκριμένα, περιλαμβάνει τις νότες Μι, Σι, Μι, Σολ#, Σι & Μι (E, B, E, G#, B & E) (Πίνακας 1.6). Διεγείροντας λοιπόν τις ανοιχτές χορδές, η συγχορδία που ακούγεται είναι η Μι ματζόρε (E). Το κούρδισμα αυτό είναι κατάλληλο για την χρήση της τεχνικής με slide και για παίξιμο «ευάερων» riff μιας και

είναι εύκολο να δημιουργηθούν μελωδίες από ένα οικείο περιβάλλον που προσδίδει η Μι ματζόρε [63].

Πίνακας 1.6: Το κούρδισμα της ανοιχτής Μι (open E) στην κιθάρα [6].

Χορδή	Νότα	Συχνότητα (Hz)
6η	E	82.41 Hz
5η	B	123.47 Hz
4η	E	163.81 Hz
3η	G#	207.65 Hz
2η	B	246.94 Hz
1η	E	329.63 Hz

Επίσης, ένα άλλο γνωστό, μοντέρνο και ιδιαίτερο εναλλακτικό κούρδισμα ανοιχτών χορδών είναι αυτό της Φα9ης (F9). Το κούρδισμα αυτό χρησιμοποιήθηκε από την μπάντα American Football στο τραγούδι Never Meant και γι' αυτό τον λόγο ονομάστηκε και American Football Tuning. Οι νότες από τις οποίες αποτελείται το συγκεκριμένο κούρδισμα είναι οι εξής Φα, Λα, Ντο, Σολ, Ντο & Μι (F, A, C, G, C & E) (πίνακας 1.7). Επίσης, αυτό του είδους το κούρδισμα το χρησιμοποιούν πιο μοντέρνες μπάντες εξαιτίας των δημιουργικών αποτελεσμάτων που προκύπτουν όταν πατιέται κάποιο τάστο και μετά τραβώντας το δάκτυλο διεγείρεται η ανοιχτή χορδή [63].

Πίνακας 1.7: Το κούρδισμα της ανοιχτής Φα9ης (open F9) στην κιθάρα [7].

Χορδή	Νότα	Συχνότητα (Hz)
6η	F	82.41 Hz
5η	A	123.47 Hz
4η	C	163.81 Hz
3η	G	207.65 Hz
2η	C	246.94 Hz
1η	E	329.63 Hz

1.1.3.3 Τα «πεσμένα» κουρδίσματα (Drop Tunings) και διπλά «πεσμένα» κουρδίσματα (Double Dropped Tunings)

Αρχικά, τα «πεσμένα» κουρδίσματα ανήκουν στην κατηγορία των εναλλακτικών κουρδισμάτων και η λογική της διεργασίας τους ξεκινά από το κλασσικό- standard κούρδισμα της κιθάρας. Πιο συγκεκριμένα, στα απλά «πεσμένα» κουρδίσματα (Drop Tunings) τυπικά ελαττώνεται ο τόνος σε μία μόνο χορδή, σχεδόν πάντα η χορδή Μι

μπάσο, ενώ όλες οι υπόλοιπες χορδές παραμένουν στο κλασσικό-standard κούρδισμα. Από την άλλη στα διπλά «πεσμένα» κουρδίσματα ελαττώνεται ο τόνος σε δύο χορδές συνήθως στην Μι μπάσο και την Μι καντίνι. Ωστόσο, αυτού του είδους τα κουρδίσματα εισήχθησαν από τους κλασσικούς και τους μπλουζ κιθαρίστες, ενώ χρησιμοποιούνται ευρέως κυρίως στις ηλεκτρικές κιθάρες, τόσο στην ροκ όσο και στην μέταλ μουσική. Τα ευρέως γνωστά «πεσμένα» κουρδίσματα είναι τα εξής: σε «πεσμένη» Ρε (Drop D), σε διπλή «πεσμένη» Ρε (double Drop D), σε «πεσμένη» Ντο (Drop C), σε διπλή «πεσμένη» Ντο (double drop C). σε κλασσικό Ντο (standard C), σε «πεσμένη» Σι, σε διπλή «πεσμένη» Σι και σε κλασσικό Σι (standard B).

Γενικός, στην κατηγορία των «πεσμένων» κουρδισμάτων (Drop Tunings), το πιο γνωστό είναι αυτό της «πεσμένης» Ρε (Drop D). Το κούρδισμα αυτό μοιάζει πολύ με το κλασσικό-standard κούρδισμα και είναι μια καλή αρχή για να πειραματιστεί κάποιος κιθαρίστας. Ουσιαστικά, σε αυτό το κούρδισμα, η χορδή Μι μπάσο ελαττώνεται κατά έναν ολόκληρο τόνο ώστε να φτάσει την νότα Ρε. Επομένως, το κούρδισμα αποτελείται από τις νότες Ρε, Λα, Ρε, Σολ, Σι & Μι (D, A, D, G, B & E) (πίνακας 1.8). Με αυτή την ελάττωση της χορδής Μι μπάσο σε Ρε, ο ήχος της κιθάρας γίνεται πιο βαρύς και πιο σκοτεινός σε σχέση με τον ήχο που παράγει η κιθάρα κουρδισμένη με τον κλασσικό-standard τρόπο. Επίσης, οι κιθαρίστες μπορούσαν να αυξήσουν την ταχύτητά τους στις συγχορδίες διότι με αυτό το κούρδισμα μπορούσαν να χρησιμοποιήσουν ένα δάκτυλο. Αντίστοιχα, το κούρδισμα της διπλής «πεσμένης Ρε (Double Drop D), περιλαμβάνει δύο χορδές η οποίες κουρδίζονται στην νότα Ρε, την Μι μπάσο και την Μι καντίνι. Οι νότες από τις οποίες αποτελείται το κούρδισμα σε διπλή «πεσμένη» Ρε είναι οι εξής Ρε, Λα, Ρε, Σολ, Σι & Ρε (D, A, D, G, B & D) (πίνακας 1.9). Κάνοντας χρήση αυτού του κουρδίσματος ο ήχος της κιθάρας γίνεται ακόμα πιο βαρύς, αλλά και πιο ομοιόμορφός εξαιτίας της πλειοψηφίας σε νότες Ρε οι οποίες διαφέρουν στις συχνότητες [63] [67] [68].

Πίνακας 1.8: Το κούρδισμα της κιθάρας σε «πεσμένη» Ρε (Drop D tuning) [8].

Χορδή	Νότα	Συχνότητα (Hz)
6η	D	73.42 Hz
5η	A	110 Hz
4η	D	146.83 Hz
3η	G	196 Hz
2η	B	246.94 Hz
1η	E	329.63 Hz

Πίνακας 1.9: Το κούρδισμα της κιθάρας σε διπλή «πεσμένη» Ρε (Double Drop D tuning) [9].

Χορδή	Νότα	Συχνότητα (Hz)
6η	D	73.42 Hz

5η	A	110 Hz
4η	D	146.83 Hz
3η	G	196 Hz
2η	B	246.94 Hz
1η	E	293.66 Hz

Επιπροσθέτως, το κούρδισμα σε «πεσμένη» Ντο (Drop C) είναι εξίσου γνωστό και ευρέος χρησιμοποιούμενο όπως αυτό της «πεσμένης» Ρε (Drop D). Ουσιαστικά, το κούρδισμα σε «πεσμένη» Ντο είναι αρκετά κοντά στο κάτω όριο σε χαμήλωμα τονικότητας της 6ης χορδής (djent territory), πράγμα που σημαίνει ότι γι αυτό το κούρδισμα δεν χρειάζεται κάποια ιδιαίτερη κιθάρα με λοξά τάστα (Fanned Fret Guitar) ή με περισσότερες χορδές (Multi Scale Guitar) [65] [66]. Επίσης, εκτός από την κατηγορία του οργάνου, θα πρέπει να ληφθεί υπ' όψιν και η κατηγορία των χορδών που χρησιμοποιούνται. Πιο συγκεκριμένα, ξεσφίγγοντας μία χορδή η τάση της μειώνεται καθώς και η τονικότητα του ήχου κάνοντας τον πιο βαρύ-μπάσο. Όμως, μικραίνοντας την τάση των χορδών η διατήρηση της νότας είναι δύσκολο και αρκετές φορές ακούγεται τρίξιμο όταν διεγείρονται. Από την άλλη, με το σφίξιμο της χορδής η τάση των χορδών και η τονικότητα του ήχου αυξάνονται. Όμως σφίγγοντας αρκετά την χορδή, η τάση της αυξάνεται τόσο πολύ με αποτέλεσμα να σπάσει. Γι' αυτό τον λόγο με το κατέβασμα της τονικότητας της κιθάρας, θα πρέπει το πάχος των χορδών να μεγαλώνει ώστε να αντισταθμιστεί το χαμήλωμα της τάσης της χορδής. Ωστόσο, το κούρδισμα της κιθάρας σε «πεσμένη» Ντο (Drop C) αποτελείται από τις νότες Ντο, Σολ, Ντο, Φα, Λα & Ρε (C, G, C, F, A & D) (πίνακας 1.10). Τέλος, το κούρδισμα σε διπλή «πεσμένη» Ντο (double Drop C) είναι παρόμοιο με αυτό της «πεσμένης» Ντο και στο μόνο που διαφέρει είναι στην νότα της 1ης χορδής. Πιο συγκεκριμένα το κούρδισμα σε διπλή «πεσμένη» Ντο αποτελείται από τις εξής νότες: Ντο, Σολ, Ντο, Φα, Λα & Ντο (C, G, C, F, A & C) (πίνακας 1.11) [62] [63] [67].

Πίνακας 1.10: Το κούρδισμα της κιθάρας σε «πεσμένη» Ντο (Drop C tuning) [10].

Χορδή	**Νότα**	**Συχνότητα (Hz)**
6η	C	65.41 Hz
5η	G	98 Hz
4η	C	130.81 Hz
3η	F	174.61 Hz
2η	A	220 Hz
1η	D	293.66 Hz

Πίνακας 1.11: Το κούρδισμα της κιθάρας σε διπλή «πεσμένη» Ντο (Double Drop C tuning) [11].

Χορδή	Νότα	Συχνότητα (Hz)
6η	C	65.41 Hz
5η	G	98 Hz
4η	C	130.81 Hz
3η	F	174.61 Hz
2η	A	220 Hz
1η	C	261.63 Hz

Εν συνεχεία, ένα διαφορετικό κούρδισμα το οποίο κατατάσσεται στην κατηγορία των πεσμένων κουρδισμάτων είναι αυτό της κλασσικής Ντο (Standard C). Αυτό το κούρδισμα παρέχει πολύ μπάσο και βαρύ ήχο. Πιο συγκεκριμένα, αποτελείται από τις νότες Ντο, Φα, Σιb, Μιb, Σολ & Ντο (C, F, Bb, Eb, G & C) (πίνακας 1.12). Το ηχητικό αποτέλεσμα που προκύπτει δίνει όγκο στον ήχο, είναι πλούσιο σε τόνο και προσδίδει μια βαθιά ηχητική βαρύτητα [67].

Πίνακας 1.12: Το κούρδισμα της κιθάρας σε κλασσική Ντο (Standard C tuning) [12].

Χορδή	Νότα	Συχνότητα (Hz)
6η	C	65.41 Hz
5η	F	87.31 Hz
4η	A#	116.54 Hz
3η	D#	155.56 Hz
2η	G	196 Hz
1η	C	261.63 Hz

Ένα διαφορετικό, αλλά αρκετά γνωστό κούρδισμα για την κιθάρα είναι αυτό της «πεσμένης» Σι (Drop B). Το συγκεκριμένο κούρδισμα είναι αρκετά αγαπημένο σε πιο νέες μουσικές γενιές μέταλ διότι προσδίδει έναν αρκετά βαρύ και επιθετικό ήχο καθώς επίσης δημιουργεί μια βαθιά και σκοτεινή ατμόσφαιρα. Πιο συγκεκριμένα, οι νότες από τις οποίες αποτελείται το κούρδισμα αυτό είναι Σι, Φα#, Σι, Μι, Σολ# & Ντο# (B, F#, B, E, G# & C#) (πίνακας 1.13). Ωστόσο, το κούρδισμα σε διπλή «πεσμένη» Σι δεν διαφέρει πολύ από αυτό της «πεσμένης» Σι, απλά ο ήχος γίνεται ακόμη πιο βαρύς σε σχέση με πριν. Πιο συγκεκριμένα αποτελείται από τις νότες, Σι, Φα#, Σι, Μι, Σολ# & Σι (B, F#, B, E, G# & B) (πίνακας 1.14) [62] [67].

Πίνακας 1.13: Το κούρδισμα της κιθάρας σε «πεσμένη» Σι (Drop B tuning) [13].

Χορδή	Νότα	Συχνότητα (Hz)
6η	B	61.74 Hz
5η	F#	92.5 Hz
4η	B	123.47 Hz
3η	E	164.81 Hz
2η	G#	207.65 Hz
1η	C#	277.18 Hz

Πίνακας 1.14: Το κούρδισμα της κιθάρας σε διπλή «πεσμένη» Σι (Double Drop B tuning) [14].

Χορδή	Νότα	Συχνότητα (Hz)
6η	B	61.74 Hz
5η	F#	92.5 Hz
4η	B	123.47 Hz
3η	E	164.81 Hz
2η	G#	207.65 Hz
1η	B	246.94 Hz

Τέλος, υπάρχει και το κούρδισμα σε κλασσικό Σι (standard B), το οποίο κατατάσσεται και αυτό στα πεσμένα κουρδίσματα. Το κούρδισμα αυτό το χρησιμοποιούν αρκετές μέταλ μπάντες διότι τους παρέχει βαθύ και σκληρό τόνο ο οποίος μοιάζει με γρύλισμα. Αυτό έχει ως αποτέλεσμα ο τόνος που δημιουργεί το συγκεκριμένο κούρδισμα να τον καθιστά κατάλληλο για αργά και βαριά μουσικά μοτίβα (riffs). Το συγκεκριμένο κούρδισμα, αποτελείται από τις νότες Σι, Μι, Λα, Ρε, Φα# & Σι (B, E, A, D, F# & B) (πίνακας 1.15) [63] [67].

Πίνακας 1.15: Το κούρδισμα της κιθάρας σε κλασσική Σι (Standard B tuning) [15].

Χορδή	Νότα	Συχνότητα (Hz)
6η	B	61.74 Hz
5η	E	82.41 Hz
4η	A	110 Hz

3η	D	146.83 Hz
2η	F#	185 Hz
1η	B	246.94 Hz

1.1.3.4 Το κούρδισμα του μισού ημιτόνου κάτω ή αλλιώς το κούρδισμα σε Μιb (Half-Step Down Tuning or Eb Tuning)

Το κούρδισμα αυτό είναι από το πιο γνωστά στον κόσμο της κιθάρας. Ουσιαστικά όλες οι χορδές από το κλασσικό κούρδισμα (standard tuning), ελαττώνονται κατά ένα ημίτονο κάνοντας τον ήχο μοναδικό και ευέλικτο. Το κούρδισμα αυτό εξελίχτηκε από μουσικούς που ήθελαν να αναδείξουν τους εαυτούς τους ως κάτι εναλλακτικό κυρίως στην μπλουζ και ροκ μουσική. Οι βασικότεροι λόγοι για τους οποίους αρκετοί κιθαρίστες επιλέγουν το συγκεκριμένο κούρδισμα, είναι ότι η κιθάρα αποκτά βαθύτερο και πλουσιότερο τόνο, ενώ ο ήχος που παράγει είναι πιο ζεστός. Επιπλέον, ένα ακόμα πλεονέκτημα που προσφέρει το συγκεκριμένο κούρδισμα είναι το εύκολο και άνετο πάτημα των τραγουδιστών πάνω στην συγκεκριμένη τονικότητα χωρίς να καταπονούν την φωνή τους. Βολεύει ιδιαίτερα τους τραγουδιστές οι οποίοι έχουν χαμηλό εύρος φωνητικών. Επίσης, με το κούρδισμα αυτό απαιτείται πιο απαλό πάτημα στις χορδές, διότι με το χαμήλωμα του τόνου αυτές μειώνουν την τάση τους με αποτέλεσμα το παίξιμο να είναι πιο εύκολο τόσο ως προς τις συγχορδίες όσο και ως προς τα riff (μουσικά μοτίβα). Τώρα όσον αφορά τις νότες από τις οποίες αποτελείται το συγκεκριμένο κούρδισμα είναι οι εξής: Μιb, Λαb, Ρεb, Σολb, Σιb & Μιb (Eb, Ab, Db, Gb, Bb & Eb) (πίνακας 1.16) [64] [69].

Πίνακας 1.16: Το κούρδισμα της κιθάρας σε μισό ημίτονο κάτω ή αλλιώς σε κούρδισμα Μιb (Half-Step down tuning or Eb tuning) [16].

Χορδή	Νότα	Συχνότητα (Hz)
6η	Eb	77.78 Hz
5η	Ab	103.83 Hz
4η	Db	138.59 Hz
3η	Gb	185 Hz
2η	Bb	233.08 Hz
1η	Eb	311.13 Hz

1.2 Αναφορά σε σχετικές εργασίες

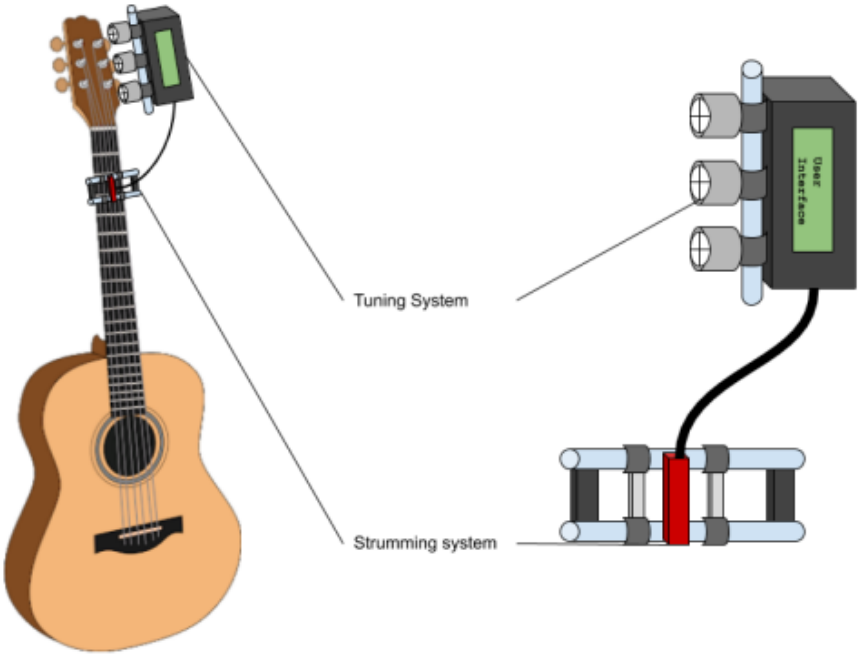
Στην συγκεκριμένη υποενότητα, αναλύονται σχετικές εργασίες. Ως επί των πλείστων, από όλες τις εργασίες που μελετήθηκαν συλλέχτηκαν πληροφορίες για την υλοποίηση της συγκεκριμένης διπλωματικής εργασίας. Ωστόσο, έγινε προσπάθεια για συνδυασμό των ιδεών και των πληροφοριών από τις εργασίες που μελετήθηκαν για την εκπόνηση και την βελτιστοποίηση της συγκεκριμένης διπλωματικής εργασίας. Η γενική

ιδέα για το στήσιμο και την ανάπτυξη της συγκεκριμένης διπλωματικής εργασίας βασίζεται στην χρήση ενός μικροελεγκτή, ο οποίος θα είναι υπεύθυνος για την εύρεση της συχνότητας της κάθε χορδής που διεγείρεται και κάθε φορά θα περιστρέφει τον αντίστοιχο κινητήρα προς την σωστή κατεύθυνση ώστε να επιτευχθεί το επιθυμητό κούρδισμα. Έτσι μέσα από την μελέτη παρόμοιων εργασιών συλλέχτηκαν πληροφορίες για το πως αυτή η ιδέα μπορεί να γίνει εφικτή.

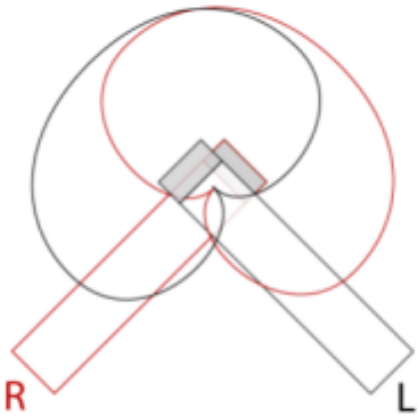
1.2.1 **Fully Automated Guitar Tuner**

Η εργασία αυτή ήταν αρκετά ενδιαφέρουσα όσον αφορά τον τρόπο σκέψης και υλοποίησης του πλήρως αυτοματοποιημένου συστήματος για το κούρδισμα των χορδών (Εικόνα 1.46). Πιο συγκεκριμένα, υπάρχουν πέντε βασικά συστήματα από τα οποία αποτελείται το πλήρως αυτοματοποιημένο σύστημα κουρδίσματος κιθάρας, το σύστημα τροφοδοσίας, το σύστημα λήψης του ήχου, το σύστημα επεξεργασίας του σήματος, το σύστημα ελέγχου στροφών και το τελικό σύστημα κουρδίσματος το οποίο περιστρέφει τα κλειδιά της κιθάρας. Ουσιαστικά, για την περιστροφή των κλειδιών γίνεται χρήση βηματικών κινητήρων (Adafruit 858 οι οποίοι ελέγχονται από τον ελεγκτή L293D), ενώ για την διέγερση της χορδής χρησιμοποιείται ένας σερβοκινητήρας (ROB-09065 SparkFun). Τόσο ο κάθε βηματικός κινητήρας όσο και ο σερβοκινητήρας ελέγχονται από το σύστημα επεξεργασίας για το ποια χορδή πρέπει να κουρδιστεί. Επιπλέον, επάνω στον σερβοκινητήρα θα υπάρχει μία πένα ώστε να διεγείρει την επιθυμητή χορδή με την ροπή που θα αποκτά κατά την περιστροφή. Η κίνηση που θα κάνει ο σερβοκινητήρας θα είναι τέτοια ώστε η πένα να διεγείρει την χορδή μία με κίνηση προς τα επάνω και μία με κίνηση προς τα κάτω. Επιπλέον, ο σερβοκινητήρας κινείται πάνω σε ράγες, ώστε να μπορεί να διεγείρει κάθε φορά την επιθυμητή χορδή. Η κύρια απαίτηση του συστήματος κουρδίσματος και του συστήματος διέγερσης είναι να ξεκινήσει το κούρδισμα και την διέγερση της χορδής αντίστοιχα εντός δύο δευτερολέπτων από την στιγμή που θα πάρει την εντολή από τον μικροελεγκτή [71].

Όσον αφορά το σύστημα του ήχου, είναι υπεύθυνο για την λήψη του ήχου και την αποστολή δεδομένων στο σύστημα επεξεργασίας, ώστε αυτό με την σειρά του να υπολογίσει την ροπή που πρέπει να ασκήσουν οι κινητήρες που συντελούν το σύστημα κουρδίσματος. Πιο συγκεκριμένα, το σύστημα ήχου αποτελείται από δύο καρδιοειδής μικρόφωνα (SPW2430 MEMS) (Εικόνα 1.47) ώστε να καταστέλλεται ο θόρυβος και η αντήχηση του δωματίου ώστε να επιτευχθεί η μέγιστη καθαρότητα του ήχου. Επίσης, για την ενίσχυση των δύο σημάτων που προέρχονται από το κάθε ένα μικρόφωνο αντίστοιχα, γίνεται χρήση δύο ενισχυτών (SSM2319) οι οποίοι είναι συμβατοί με τα μικρόφωνα και το σύστημα επεξεργασίας. Τα σήματα των μικροφώνων θα πρέπει να δίνουν το πλήρες εύρος στον αναλογικό-ψηφιακό μετατροπέα (ADC), ώστε να υπάρχει η ελάχιστη επιβάρυνση και παραμόρφωση από τον ενισχυτή. Η απαίτηση για το σύστημα του ήχου είναι ο εντοπισμός του τόνου-νότας της χορδής με ακρίβεια [71].



Εικόνα 1.46: Το πλήρως αυτοματοποιημένο σύστημα κουρδίσματος της κιθάρας (Fully Automated Guitar Tuner) [93].



Εικόνα 1.47: Η μορφή των δύο καρδιοειδής μικροφώνων [94].

Από την άλλη, το σύστημα επεξεργασίας, πρέπει να εντοπίζει την διαφορά του τρέχον τόνου της διεγερμένης χορδής σε σχέση με τον επιθυμητό, δηλαδή τον τόνο που θα έπρεπε να είχε η χορδή. Τα δεδομένα της διαφοράς αυτής στέλνονται στο σύστημα κουρδίσματος, το οποίο με την σειρά του περιστρέφει το κατάλληλο κλειδί προς την κατάλληλη κατεύθυνση. Αυτό επιτυγχάνεται με την χρήση του σωστού αλγόριθμο ο οποίος είναι υπεύθυνος για την σύγκριση του πραγματικού τόνου σε σχέση με τον επιθυμητό καθώς και για την περιστροφή του σωστού κλειδιού αντίστοιχα. Ακόμα, το σύστημα κουρδίσματος είναι αυτό το οποίο περιλαμβάνει την διεπαφή με τον χρήστη και την σύνδεση με το σύστημα τροφοδοσίας [71].

Ο αλγόριθμος που χρησιμοποιείται στην συγκεκριμένη διπλωματική εργασία μοιάζει με αυτόν του Fujishima στην πιο απλοποιημένη του μορφή. Ουσιαστικά, λαμβάνεται ένα στιγμιότυπο με τα δεδομένα το οποίο περιέχει εξασθενισμένους τους πλευρικούς λοβούς και στην συνέχεια μετασχηματίζεται με Γρήγορο Μετασχηματισμό Fourier (F.F.T.). Ο μετασχηματισμός έχει σκοπό την μεταφορά του στιγμιότυπου από το πεδίου το χρόνου στο πεδίο της συχνότητας. Επίσης, το μετασχηματισμένο κατά F.F.T. σήμα διέρχεται μέσα από ένα φίλτρο διέλευσης ζώνης με σκοπό την αποκοπή των αρμονικών συχνοτήτων

και την διατήρηση της κύριας συχνότητας. Ουσιαστικά, οι αρμονικές συχνότητες είναι ακέραια πολλαπλάσια της κύριας συχνότητας του σήματος. Ωστόσο η ζώνης αποκοπής πρέπει να είναι τέτοια ώστε να μην αποκόπτονται τα δεδομένα που πρέπει να κρατηθούν. Πιο συγκεκριμένα, για την κιθάρα οι συχνότητες που πρέπει να αποκοπούν είναι κάτω των 60Hz και άνω των 2000Hz. Επειδή λοιπόν οι αρμονικές συχνότητες κάποιες φορές έχουν μεγαλύτερο πλάτος από την κύρια συχνότητα με αποτέλεσμα να γίνεται δυσκολότερη η διάκριση των συχνοτήτων, την λύση την δίνει το γινόμενο φάσματος αρμονικών (Harmonic Product Spectrum ή αλλιώς HPS). Το HPS φιλτράρει τις υψηλές συχνότητες που αποτελούν τις αρμονικές, αφήνοντας την κύρια συχνότητα ή ένα μικρό μέρος των αρμονικών συχνοτήτων [71].

Σημαντικό ρόλο στο συγκεκριμένο σύστημα παίζει η λογική ελέγχου η οποία καθορίζει την σειρά των μηχανικών γεγονότων καθώς και την βαθμονόμηση κατά το κούρδισμα των κλειδιών μιας και κάθε χορδή ασκεί διαφορετική τάση στο αντίστοιχο κλειδί. Πιο συγκεκριμένα, ο μικροελεγκτής που επιλέχθηκε στην συγκεκριμένη διπλωματική εργασία είναι ο τύπος STM32F042x4 με περίβλημα τύπου LQFP48 και έχει την δυνατότητα να κάνει πολύπλοκους υπολογισμούς οι οποίοι χρειάζονται τόσο για την ανίχνευση της συχνότητας των νοτών, όσο και για την λογική ελέγχου όλου του συστήματος. Ουσιαστικά αυτός ο τύπος μικροελεγκτή περιλαμβάνει αναλογικό-ψηφιακό μετατροπέα (ADC) των 12 bit, υποστήριξη I2C για το μικρόφωνο και βασίζεται στην 32bit αρχιτεκτονική της ARM για προηγμένους υπολογισμούς. Αυτό λοιπόν το σύστημα αποτελεί το σύστημα επεξεργασίας και απαιτείται η ταυτόχρονη επικοινωνία με όλα τα μηχανικά συστήματα, ενώ το κούρδισμα δεν θα υπερβαίνει το ένα λεπτό. Για τον έλεγχο των απαιτήσεων αυτών τοποθετούνται όλα τα εξαρτήματα πάνω στην κιθάρα και η διέγερση και το κούρδισμα θα πρέπει να συνεχίζονται μέχρι να κουρδιστεί πλήρως, ενώ για τον χρόνο κουρδίσματος γίνεται χρήση χρονομέτρου και μετριέται ο χρόνος που χρειάζεται μέχρι να κουρδιστεί πλήρως η κιθάρα [71].

Τέλος, το σύστημα τροφοδοσίας όλου του συστήματος αποτελείται από δύο μπαταρίες λιθίου συνδεδεμένες σε σειρά, δίνοντας συνολική ονομαστική τάση στα 7,4Volt. Επειδή όμως όλα τα ηλεκτρονικά του συστήματος χρειάζονται 5Volt για να λειτουργήσουν, χρησιμοποιείται και ένας μετατροπέας υποβάθμισης τύπου D24V22F5 ο οποίος στην έξοδό του δίνει 5Volt τάση και ρεύμα από 1.9A έως 2.5A. Το ρεύμα το κάθε ένα από τα ηλεκτρονικά μέρη του συστήματος είναι 4mA ο κάθε ενισχυτής, 500mA ο μικροελεγκτής, 600mA οι βηματικοί κινητήρες και το σύστημα διέγερσης περίπου 165mA. Επομένως αφού οι μπαταρίες λιθίου έχουν χωρητικότητα έως 3500mAh αυτό σημαίνει ότι μπορούν να δώσουν την μέγιστη ισχύ τους για πάνω από μία ώρα. Επίσης, για μεγαλύτερη ασφάλεια και προς αποφυγήν του ενδεχομένου φωτιάς λόγο υπερθέρμανσης των μπαταριών, γίνεται χρήση ενός ημιαγώγιμου αισθητήρα θερμοκρασίας και ενός LED ώστε να ειδοποιείτε ο χρήστης σε περίπτωση που η θερμοκρασία φτάσει τους 55^{o}C [71].

Τέλος, στην διπλωματική εργασία αυτή αναλύεται το κόστος των υλικών και της εργασίας, καθώς και το πρόγραμμα της πορείας της εργασίας. Επίσης, εμφανίζονται αναλυτικά και οι πηγές που μελετήθηκαν για την υλοποίηση της συγκεκριμένης διπλωματικής εργασίας [71].

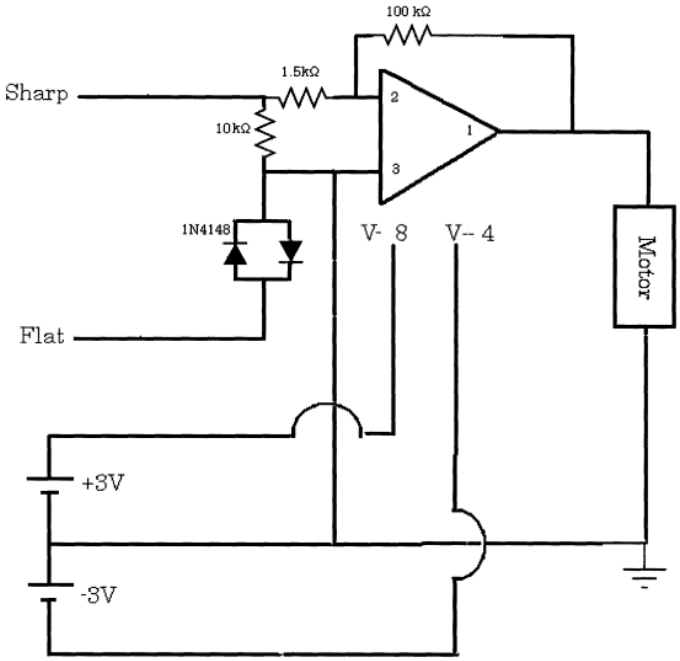
1.2.2 **Automatic Acoustic Guitar Tuner**

Σε αυτήν τη διπλωματική εργασία που μελετήθηκε χρησιμοποιείται μία καταπληκτική ιδέα όσον αφορά τον σχεδιασμό ενός αυτομάτου κουρδιστηριού για ακουστική κιθάρα. Αρχικά, είναι γνωστό ότι οι περισσότερες κιθάρες είναι κατασκευασμένες από ξύλο και αυτό έχει ως αποτέλεσμα με τις μεταβολές της θερμοκρασίας να συστέλλεται η να διαστέλλεται το όργανο. Εξαιτίας αυτού του φαινομένου επηρεάζεται το κούρδισμα της κιθάρας με αποτέλεσμα το όργανο να ξεκουρδίζεται. Δημιουργώντας λοιπόν ένα αυτόματο κουρδιστήρι, ο χρόνος κουρδίσματος ελαττώνεται σημαντικά και επωφελή τόσο τους αρχάριους οι οποίοι δυσκολεύονται με το κούρδισμα της κιθάρας, αλλά και τους επαγγελματίες δίνοντάς τους ακριβές κούρδισμα [72].

Το σύστημα που σχεδιάστηκε χρησιμοποιεί το κουρδιστήρι GA-30 της εταιρίας Korg (Εικόνα 1.48) το οποίο χρησιμοποιείται και για κούρδισμα κιθάρας αλλά και μπάσου [73]. Το συγκεκριμένο κουρδιστήρι περιλαμβάνει ένα μικρόφωνο από το οποίο λαμβάνει τον ήχο της κάθε χορδής και ανάβει το αντίστοιχο LED στην πρόσοψη ως ένδειξη για το αν η χορδή είναι κουρδισμένη ή όχι. Πιο συγκεκριμένα, το κουρδιστήρι περιλαμβάνει τρία LED όπου το ένα αντιστοιχεί στην ύφεση της νότας (flat note) που εμφανίζεται στην οθόνη, το άλλο στην δίεση της νότας (sharp note) που εμφανίζεται στην οθόνη και το άλλο στο ότι η νότα που εμφανίζεται στην οθόνη έχει ακριβός την συχνότητα που πρέπει. Ουσιαστικά, το κουρδιστήρι GA-30 της Korg χρησιμοποιείται σε έναν κλειστό σύστημα αυτομάτου ελέγχου το για το κούρδισμα των χορδών της κιθάρας σε μικρό χρονικό διάστημα. Με αυτό το σύστημα του αυτομάτου κουρδιστηριού, οι συχνότητες των χορδών μπορούν να γίνουν αντιληπτές με μεγαλύτερη ακρίβεια και αυτό έχει ως αποτέλεσμα και το κούρδισμα της κιθάρας να γίνεται και αυτό με ακρίβεια. Εν συνεχεία, ο βρόγχος ανατροφοδότησης είναι αυτός ο οποίος ενεργοποιεί το μηχανικό μέρος του συστήματος για την περιστροφή του κλειδιού και συνεπώς το κούρδισμα της χορδής. Η λογική πίσω από τον σχεδιασμό του κυκλώματος αυτομάτου ελέγχου ξεκίνησε με την απλή σκέψη ότι ένα σήμα από το κουρδιστήρι και συγκεκριμένα από τα LED της ύφεσης και της δίεσης, θα οπλίζει ένα αντίστοιχο ρελέ το οποίο θα ενεργοποιεί τον κινητήρα και θα περιστρέφει το κλειδί της χορδής προς την επιθυμητή κατεύθυνση. Μόλις σβήσει το LED της ύφεσης ή της δίεσης αντίστοιχα, το ρελέ θα επιστρέψει σε κατάσταση ηρεμίας ανοίγοντας το κύκλωμα τροφοδοσίας του κινητήρα με αποτέλεσμα να σταματήσει την περιστροφή. Ένα από τα βασικότερα προβλήματα της σκέψης αυτής είναι ο ηλεκτρονόμος ή αλλιώς ρελέ ο οποίος είναι ογκώδης και δεν έχει τον ιδανικό χρόνο απόκρισης. Επίσης, η προδιαγραφές της τάσης για τον οπλισμό του ηλεκτρονόμου είναι πολύ μεγαλύτερες από αυτή που μπορεί να δώσει ένα LED. Η μία ιδανική επιλογή είναι η χρήση ενός γραμμικού ενισχυτή τύπου LT1886 DUAL 700Mhz με έξοδο έως 200mA. Ουσιαστικά, στον ενισχυτή αυτόν θα συνδεθούν τα LED από το κουρδιστήρι GA-30 της Korg, αντιστάσεις, δίοδοι, η τροφοδοσία και ο κινητήρας (Εικόνα 1.49). Οι δίοδοι, είναι συνδεδεμένες αντί-παράλληλα ώστε να αποκοπεί ο θόρυβος ο οποίος θα δημιουργούσε εσφαλμένο σήμα με αποτέλεσμα την ανεπιθύμητη απόκριση του κινητήρα. Από την άλλη, οι αντιστάσεις στις εισόδους του ενισχυτή έχουν σκοπό την αύξηση του κέρδους μεταξύ αυτών και των LED καθώς και της επιθυμητής τάσης της εξόδου που θα εφαρμοστεί στον κινητήρα. Επιπλέον μία αντίσταση συνδέεται μεταξύ των δύο εισόδων η οποία έχει σκοπό τον εμποδισμό της ροής του ρεύματος σε περίπτωση υπέρβασης. Όμως στην έξοδό του ο ενισχυτής αυτός δίνει 200mA το οποίο είναι πολύ λίγο για σε περίπτωση που απαιτηθεί από τον κινητήρα

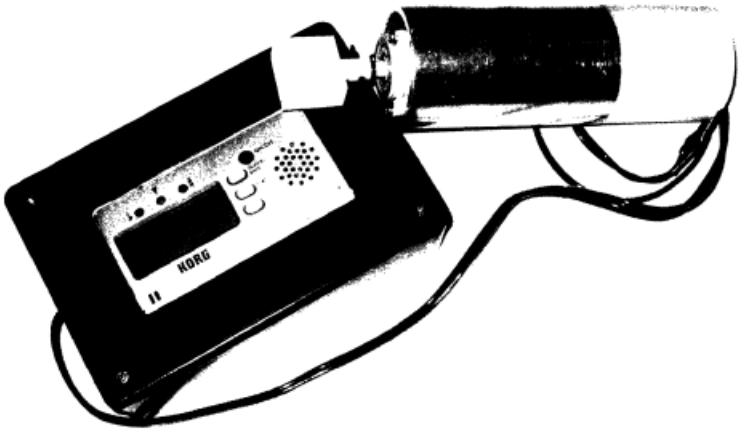
να περιστραφεί με μεγαλύτερη ροπή. Γι' αυτόν τον λόγο, εναλλακτικά θα μπορούσε να χρησιμοποιηθεί ο ενισχυτής LT1210 Linear Technologies ο οποίος παρέχει ρεύμα εξόδου έως 1.1A [72].

Εικόνα 1.48: Το κουρδιστήρι GA-30 της εταιρίας Korg [95].



Εικόνα 1.49: Το ηλεκτρονικό κύκλωμα που συνδέει το κουρδιστήρι με τον κινητήρα [96].

Από την άλλη πλευρά, το μηχανολογικό σχέδιο βασίζεται στην ροπή που πρέπει να ασκηθεί στο κλειδί για το κούρδισμα της χορδής. Η ροπή που θα ασκηθεί σε κάθε χορδή είναι 2.4in·lbs ή αλλιώς 0.03 kg·m. Ο τύπος κινητήρα που επιλέχθηκε είναι ο Tamiva RC-260, ο οποίος λειτουργεί με καρβουνάκια άνθρακα για μεγαλύτερη διάρκεια ζωής, ενώ η τάση λειτουργίας του είναι μεταξύ 3Volt και 4.5Volt. Για να αποφευχθεί το σπάσιμο-κόψιμο της χορδής από υπερβολική πίεση, η περιστροφή του κλειδιού θα πρέπει να γίνει αργά. Γι' αυτό τον λόγο η περιστροφή απαιτείται να γίνεται με συχνότητα 0.5hz, δηλαδή μισή στροφή το δευτερόλεπτο. Επίσης πρέπει να σχεδιαστεί ένας αντάπτορας ο οποίος θα εφαρμόζει από την μία στα κλειδιά της κιθάρας και από την άλλη στον άξονα του κινητήρα. Ο αντάπτορας αυτός θα πρέπει να σχεδιαστεί έτσι ώστε να ταιριάζει και σε κλειδιά άλλου μεγέθους που έχουν άλλες κιθάρες. Επιπλέον. Ο κινητήρας τοποθετείται και βιδώνεται σε έναν αλουμινένιο κύλινδρο ώστε ο μουσικός να μπορέσει να τον κρατήσει με το χέρι του και να τον μεταφέρει από το ένα κλειδί στο άλλο (Εικόνα 1.50) [72].



Εικόνα 1.50: Το τελικό σύστημα για το αυτόματο κουρδιστήρι ακουστικής κιθάρας [97].

Τέλος, η ζήτηση για τέτοιου είδους κουρδιστήρια στην αγορά είναι μεγάλη και αξίζει να μπει σε μαζική παραγωγή. Όμως για να είναι ένα τέτοιο προϊόν ελκυστικό στην αγορά, σημαντικό ρόλο παίζει το μέγεθος και η αισθητική. Δηλαδή, τα ηλεκτρονικά του και οι μπαταρίες του προϊόντος θα μπορούσαν να έχουν μικρότερο μέγεθος ώστε το τελικό προϊόν να είναι πιο ελκυστικό και πιο εύχρηστο. Ωστόσο, μία παραλλαγή του αυτομάτου συστήματος κουρδίσματος ακουστικής κιθάρας, θα μπορούσε να εξελίξει το σύστημα ακόμα περισσότερο. Μία πιθανή παραλλαγή του αυτόματου κουρδιστηριού θα ήταν να προσκολλάται πάνω στην κεφαλή της κιθάρας και να την κουρδίζει με την διέγερση της κάθε χορδής. Θα πρέπει όμως ένας βραχίονας να επεκτείνεται σε κάθε ένα από τα κλειδιά με αποτέλεσμα να γίνεται δύσκολη η μεταφορά της απαραίτητης ροπής στον βραχίονα από τον κινητήρα. Μία άλλη πιθανή παραλλαγή η οποία λύνει και το πρόβλημα της μεταφοράς της ροπής από τον κινητήρα στον βραχίονα είναι να τοποθετηθεί το αυτόματο κουρδιστήρι κατευθείαν πάνω στα κλειδιά [72].

1.2.3 **A Digital Guitar Tuner**

Η μελέτη αυτή της διπλωματικής εργασίας είχε αρκετό ενδιαφέρον και πολλές πληροφορίες για το πώς μπορεί κάποιος να σχεδιάσει ένα ψηφιακό κουρδιστήρι με την χρήση του Matlab. Ουσιαστικά, στην συγκεκριμένη εργασία γίνεται ανάλυση των απαραίτητων παραμέτρων που χρειάζονται για τον σχεδιασμό κουρδιστηριών κιθάρας, αναπτύσσεται κατάλληλος αλγόριθμος για τον εντοπισμό της συχνότητας της χορδής που διεγείρεται και δημιουργείτε μία γραφική διεπαφή φιλική προς το χρήστη ώστε να αλληλεπιδρά με το σύστημα. Πιο συγκεκριμένα, το ψηφιακό αυτό κουρδιστήρι με την βοήθεια του Matlab θα μπορεί να φιλτράρει, να μετράει, να αναλύει και να δίνει τις απαραίτητες οδηγίες στον χρήστη για το κούρδισμα του οργάνου [74].

Όπως είναι γνωστό, οι νότες στην κιθάρα όπως και σε πολλά άλλα όργανα αποτελούνται από ένα φάσμα συχνοτήτων το οποίο αποτελείται κυρίως από την θεμελιώδη συχνότητα και τις αρμονικές συχνότητες. Οι αρμονικές συχνότητες είναι ουσιαστικά πολλαπλάσια της θεμελιώδης συχνότητας. Η συχνότητα όμως στην οποία βασίζεται ο αλγόριθμος ώστε να κουρδιστεί σωστά η αντίστοιχη χορδή είναι η θεμελιώδης συχνότητα. Για την εύρεση της θεμελιώδης συχνότητας γίνεται δειγματοληψία του ήχου που παράγει η διεγερμένη χορδή. Εν συνεχεία, με την χρήση φίλτρων γίνεται εξάλειψη του θορύβου από το σήμα και με τον μετασχηματισμό Fourier μεταφέρεται το σήμα από το πεδίο του

χρόνου στο πεδίο της συχνότητας. Ουσιαστικά, με την χρήση του FFT εντοπίζεται η θεμελιώδης συχνότητα της χορδής που διεγείρεται η οποία θεωρητικά είναι αυτή με το μεγαλύτερο πλάτος. Για την ευκολία του εντοπισμού της θεμελιώδης συχνότητας, πριν τον μετασχηματισμό Fourier (FFT) το σήμα διέρχεται μέσα από ένα φίλτρο ζώνης διέλευσης για να μειωθεί ο θόρυβος από τις αρμονικές συχνότητες που παράγονται. Πιο συγκεκριμένα, οι αρμονικές συχνότητες είναι πολλαπλάσια της θεμελιώδους συχνότητας και δυσκολεύουν την εύρεση της πραγματικής συχνότητας. Επειδή, όμως ο αλγόριθμος απαιτεί τουλάχιστον τρεις αρμονικές συχνότητες καθώς και την θεμελιακή συχνότητα, θα πρέπει το φίλτρο να κόβει αρμονικές συχνότητες πάνω από 1320Hz (αφού η μέγιστη θεμελιακή συχνότητα είναι 440Hz) και αρμονικές συχνότητες κάτω από 75Hz. Ο υπολογισμός της ανάλυσης της συχνότητας μεταξύ των δειγμάτων γίνεται με την χρήση του ακόλουθου τύπου:

$$Resolution = \frac{Sampling\ Rate}{Number\ of\ Samples}$$

Στην προκειμένη περίπτωση θα πρέπει η ανάλυση της συχνότητας του FFT να είναι τουλάχιστον 0.5Hz η οποία αντιστοιχεί στην ευαισθησία του ανθρώπινου αυτιού. Επομένως αυτό επιτυγχάνεται με την εναλλαγή της διάρκειας δειγματοληψίας (Sample Rate). Ωστόσο, πρέπει να ληφθεί υπ' όψιν ότι όσο μεγαλύτερη η διάρκεια δειγματοληψίας, τόσο αυξάνεται και ο χρόνος απόκτησης δεδομένων [74].

Από την άλλη όμως μόνο ο μετασχηματισμός Fourier δεν μπορεί να δώσει με ακρίβεια την συχνότητα της χορδής. Δηλαδή, αν μία αρμονική συχνότητα έχει μεγαλύτερο πλάτος από την θεμελιώδη συχνότητα τότε θα είναι δύσκολο να εντοπιστεί η πραγματική συχνότητα της χορδής. Επομένως, για μεγαλύτερη ακρίβεια στον εντοπισμό της συχνότητας λαμβάνεται υπ' όψιν η ιδιότητα των αρμονικών συχνοτήτων οι οποίες εμφανίζονται σε πολλαπλάσια της θεμελιακής συχνότητας. Πιο συγκεκριμένα, κάνοντας πρόσθεση ή πολλαπλασιασμό τα σήματα που έχουν προκύψει από την δειγματοληψία του ίδιου σήματος αλλά με διαφορετικούς παράγοντες, τότε το σήμα που προκύπτει παράγει την μέγιστη κορυφή στην θεμελιακή συχνότητα. Ουσιαστικά, στην τελική υλοποίηση γίνεται δύο φορές δειγματοληψία του σήματος με διαφορετικό συντελεστή την κάθε φορά και έπειτα τα σήματα που προκύπτουν προστίθενται. Εν συνεχεία, το άθροισμα μετασχηματίζεται κατά Fourier και έτσι εμφανίζεται η μέγιστη κορυφή στην θεμελιακή συχνότητα [74].

Στο επόμενο βήμα, αφού έχει εντοπιστεί η πραγματική συχνότητα της χορδής, θα πρέπει να συγκριθεί με την επιθυμητή συχνότητα. Όπως είναι γνωστό οι νότες μαζί με τις διέσεις & υφέσεις είναι δώδεκα και αποτελούν μία οκτάβα, ενώ το διάστημα ανάμεσά τους ονομάζεται ημίτονο και δύο ημίτονα αποσκοπούν σε έναν τόνο. Για τον υπολογισμό της συχνότητας μιας νότας μπορεί να γίνει χρήση του ακόλουθου τύπου:

$$Freq = 440 \cdot 2^{(\frac{n}{12})}$$

Όπου n ο αριθμός των ημιτόνων από την Λα της 4ης οκτάβας (A4).

Οι νότες απέχουν μεταξύ τους λογαριθμικά διαστήματα πράγμα που σημαίνει ότι στις πιο χαμηλές συχνότητες οι νότες απέχουν μεταξύ τους μικρότερα διαστήματα σε σχέση με τις

υψηλότερες συχνότητες. Αυτό έχει ως αποτέλεσμα η ανίχνευση της συχνότητας των μπάσων χορδών να είναι πιο δύσκολη εξαιτίας του θορύβου. Αυτό αντιμετωπίζεται στην προκειμένη περίπτωση με την χρήση φίλτρου. Όμως τα δείγματα που χρησιμοποιούνται στον μετασχηματισμό Fourier αποτελούν γραμμικά διαστήματα που ισαπέχουν μεταξύ τους κάνοντας έτσι ακόμα πιο δύσκολο τον εντοπισμό της θεμελιώδης συχνότητας, διότι σε ένα δείγμα μπορούν να εντοπιστούν πολλές νότες. Μία μέθοδος που θα μπορούσε να λύσει το συγκεκριμένο πρόβλημα περιλαμβάνει την δημιουργία αλγορίθμου όπου ο μετασχηματισμός Fourier του σήματος γίνεται σε λογαριθμική κλίμακα. Για την υλοποίηση της συγκεκριμένης εργασίας όμως, η διαφορά αυτή θα μπορούσε να θεωρηθεί αμελητέα [74].

Ένα εξίσου σημαντικό κομμάτι που πρέπει να ληφθεί υπ' όψιν είναι το πόσο πρέπει να περιστραφεί το κλειδί για να κουρδιστεί η συγκεκριμένη χορδή. Ουσιαστικά, το πόσο πρέπει να περιστραφεί το κλειδί της κάθε χορδής, καθορίζεται από την διαφορά της επιθυμητής τιμής της συχνότητας και της πραγματικής συχνότητας της χορδής. Για την εύρεση της μεταβολής της συχνότητας κατά την περιστροφή του κλειδιού της κάθε χορδής επαναλαμβάνεται μία διαδικασία τρεις φορές. Πιο συγκεκριμένα, λαμβάνονται τρεις φορές δείγματα από την κάθε χορδή όταν έχουν περιστραφεί κατά 180^{o} ωρολογιακά και αντί-ωρολογιακά το κάθε κλειδί. Παρατηρείτε λοιπόν ότι η μεταβολή της συχνότητας σε μια χορδή όταν το αντίστοιχο κλειδί περιστρέφεται κατά την ίδια γωνία και από τις δύο πλευρές είναι ίδια. Όμως η μεταβολή της συχνότητας είναι διαφορετική μεταξύ των χορδών και αυτό οφείλετε στα κατασκευαστικά χαρακτηριστικά της κάθε χορδής (Εικόνα 1.51). Έτσι λοιπόν, η μεταβολή της συχνότητας της κάθε χορδής διαιρείτε κατά 180^{o} ώστε να προκύψει η μεταβολή της συχνότητας ανά μοίρα. Συνεπώς, διαιρώντας την διαφορά της επιθυμητής με την πραγματική συχνότητα της κάθε χορδής με την αντίστοιχη τιμή συχνότητας ανά μοίρα, προκύπτει η γωνία που πρέπει να περιστραφεί το κλειδί ώστε να κουρδιστεί η αντίστοιχη χορδή. Το πρόσημο στην γωνία που προκύπτει καθορίζει την φορά περιστροφής [74].

Επιπλέον, στην συγκεκριμένη διπλωματική εργασία που μελετήθηκε, αναπτύσσεται μία γραφική διεπαφή με την χρήση του Matlab GUI builder ώστε να μπορεί να αλληλεπιδρά ο χρήστης με το σύστημα. Πιο συγκεκριμένα, η διεπαφή αυτή εμφανίζει αρκετές πληροφορίες και δυνατότητες στον χρήστη, όπως η επιλογή της χορδής που επιθυμεί να κουρδίσει ο χρήστης με το πάτημα αντίστοιχων κουμπιών, εμφανίζει την γραφική παράσταση του καταγεγραμμένου σήματος καθώς και του επεξεργασμένου σήματος, παρέχει κουμπί «Test String» για την εκκίνηση της δοκιμής της χορδής, παρέχει την συχνότητα που θα πρέπει να έχει η χορδή που διεγείρεται καθώς επίσης εμφανίζει σε έναν πίνακα πότε πρέπει να διεγερθεί η χορδή και πότε πρέπει να σταματήσει. Επιπλέον, στην γραφική διεπαφή εμφανίζεται σε πίνακα η συχνότητα της κάθε χορδής καθώς και η προσεγγιστική γωνία που πρέπει να περιστραφεί το κάθε ένα για το κούρδισμα της κάθε χορδής.

STRING	E (STRING 6)		
	Change in frequency		
	Test 1	Test 2	Test 3
Turning clockwise 180°	+ 4 Hz	+ 4 Hz	+ 4Hz
Turning anticlockwise 180°	- 4 Hz	- 4 Hz	- 4 Hz
STRING	**A (STRING 5)**		
	Change in Frequency		
	Test 1	Test 2	Test 3
Turning clockwise 180°	+ 4.5 Hz	+ 4.5 Hz	+ 4.5 Hz
Turning anticlockwise 180°	- 4.5 Hz	- 4.5 Hz	- 4.5 Hz
STRING	**D (STRING 4)**		
	Change in Frequency		
	Test 1	Test 2	Test 3
Turning clockwise 180°	+ 5.5 Hz	+ 5 Hz	+ 5 Hz
Turning anticlockwise 180°	- 5 Hz	- 5 Hz	- 5 Hz
STRING	**G (STRING 3)**		
	Change in Frequency		
	Test 1	Test 2	Test 3
Turning clockwise 180°	+ 12 Hz	+ 12 Hz	+ 12 Hz
Turning anticlockwise 180°	- 12 Hz	- 12 Hz	- 12 Hz
STRING	**B (STRING 2)**		
	Change in Frequency		
	Test 1	Test 2	Test 3
Turning clockwise 180°	+ 13.5 Hz	+ 13.5 Hz	+ 13.5 Hz
Turning anticlockwise 180°	- 12.5 Hz	- 13.5 Hz	- 13.5 Hz

STRING	E (STRING 1)		
	Change in Frequency		
	Test 1	Test 2	Test 3
Turning clockwise 180°	+ 16 Hz	+ 15.5 Hz	+ 15 Hz
Turning anticlockwise 180°	- 15 Hz	- 15 Hz	- 15 Hz

Εικόνα 1.51: Οι μεταβολές των συχνοτήτων των χορδών κατά την περιστροφή των αντίστοιχων κλειδιών κατά 180° αριστερόστροφα και δεξιόστροφα [98].

Ο χρήστης για να κουρδίσει την κιθάρα του θα πρέπει να ακολουθήσει μια συγκεκριμένη διαδικασία. Αρχικά, θα πρέπει να επιλέξει την χορδή την οποία επιθυμεί να κουρδίσει και να πατήσει το κουμπί «Test String». Στην συνέχεια εμφανίζεται ένα μήνυμα το οποίο προτρέπει τον χρήστη να διεγείρει την συγκεκριμένη χορδή και μετά από 2 δευτερόλεπτα να σταματήσει ώστε να ακολουθήσει η ανάλυση του σήματος. Τέλος, η διεπαφή εμφανίζει το αποτέλεσμα της ανάλυσης, δηλαδή την πραγματική συχνότητα της χορδής καθώς και την προσεγγιστική γωνία που πρέπει να περιστραφεί το κλειδί για να κουρδιστεί η συγκεκριμένη χορδή [74].

Τέλος, για την προσομοίωση και επέκταση του αυτοματοποιημένου κουρδίσματος της κιθάρας σε Hardware, θα μπορούσε να σχεδιαστεί ένα κύκλωμα ελέγχου ενός βηματικού κινητήρα DC ο οποίος θα περίστρεφε το κλειδί κουρδίζοντας την αντίστοιχη χορδή. Συνοπτικά, στην συγκεκριμένη εργασία αναλύεται η μέθοδος του γινομένου των αρμονικών (Harmonic Product Spectrum) ώστε να εντοπιστεί η θεμελιώδης συχνότητα της χορδής. Επίσης, αναφέρονται παράγοντες που επηρεάζουν τον εντοπισμό της θεμελιώδης συχνότητας καθώς και ο τρόπος με τον οποίο απλοποιήθηκε ο αλγόριθμος ανίχνευσης της συχνότητας με την χρήση του γινομένου των αρμονικών συχνοτήτων. Τέλος, εξηγείται ο

τρόπος με τον οποίο αναπτύχθηκε και λειτουργεί η γραφική διεπαφή με την οποία αλληλεπιδρά ο χρήστης για το κούρδισμα του οργάνου [74].

1.2.4 **Robotic Electric Guitar Tuner**

Σε αυτή την διπλωματική εργασία σχεδιάζεται ένα σύστημα κουρδίσματος των χορδών μιας κιθάρας. Το όλο σύστημα βασίζεται πάνω σε ένα μικροεπεξεργαστή Arduino UNO, σε ένα μικρόφωνο και έναν κινητήρα για την περιστροφή των κλειδιών της κιθάρας. Ουσιαστικά, το Arduino UNO περιλαμβάνει τους αλγόριθμους οι οποίοι επεξεργάζονται το σήμα που λαμβάνεται από το μικρόφωνο και παράγουν τα κατάλληλα PWM (Pulse Width Modulation) σήματα για τον έλεγχο του κινητήρα και κατά συνέπεια την περιστροφή του κλειδιού [75].

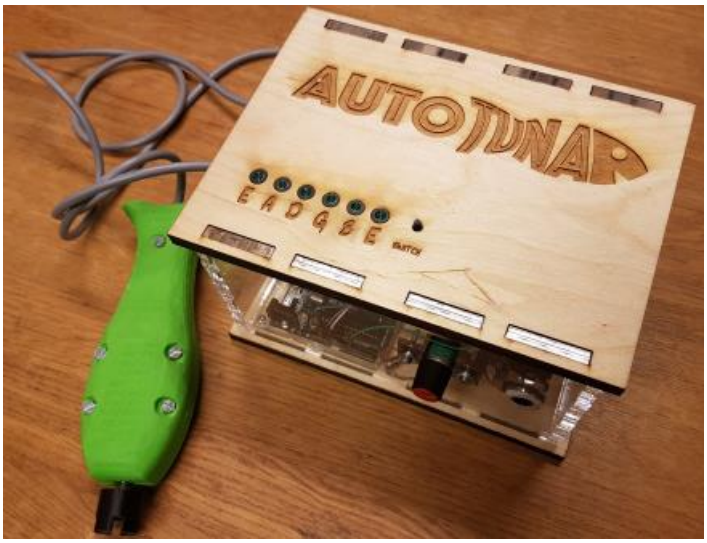
Παλιότερα, τις χορδές της κιθάρας της κούρδιζαν με την χρήση διαπασών ενώ με την εξέλιξη της τεχνολογίας, δημιουργήθηκαν τα πρώτα ηλεκτρονικά κουρδιστήρια τα οποία αντιλαμβάνονταν την συχνότητα της κάθε χορδής και καθοδηγούσαν τον κιθαρίστα για το κούρδισμα της κάθε μίας χορδής. Ωστόσο, υπάρχουν αυτόματα κουρδιστήρια τα οποία εφαρμόζονται πάνω στην κιθάρα και παρέχουν την δυνατότητα γρήγορου και αξιόπιστου κουρδίσματος [75].

Εν συνεχεία, ο μικροεπεξεργαστής Arduino UNO περιλαμβάνει τον μικροελεγκτή ATmega328P ο οποίος χρησιμοποιείτε για την επεξεργασία των σημάτων και για τον έλεγχο του κινητήρα για την περιστροφή των κλειδιών. Όσο για το σήμα που παράγεται από το μικρόφωνο πρέπει να ενισχυθεί με μία σταθερή DC συνιστώσα με την χρήση δύο ενισχυτών, ενώ με την χρήση ενός χαμηλοπερατού φίλτρου γίνεται η αφαίρεση του θορύβου. Για την επίτευξη ωστόσο του κουρδίσματος της χορδής θα πρέπει ο κινητήρας να περιστρέφει το κλειδί και από τις δύο πλευρές, συνεπώς γίνεται χρήση μιας γέφυρας τύπου «Η» [75].

Επιπροσθέτως, η συγκεκριμένη εργασία αποτελείται από ένα κουτί και μία βάση στα οποία τοποθετούνται όλα τα εξαρτήματα. Πιο συγκεκριμένα, στο κουτί τοποθετούνται ο μικροεπεξεργαστής, οι ενισχυτές και η γέφυρα τύπου «Η», ενώ στην λαβή τοποθετείταιαι ο κινητήρας. Η σύνδεση του κινητήρα στην λαβή με το υπόλοιπο σύστημα που βρίσκεται στο κουτί γίνεται με ένα καλώδιο (Εικόνα 1.52). Για να μπορέσει ο κινητήρας να περιστρέψει το κλειδί, σχεδιάστηκε ένας αντάπτορας ο οποίος εφαρμόζει από την μία πλευρά στο κλειδί και από την άλλη στον άξονα του κινητήρα. Όσον αφορά της στροφές του κινητήρα θα ελέγχονται με την χρήση ενός PID ελεγκτή του οποίου οι παράμετροι καθορίζονται με την μέθοδο δοκιμής και σφάλματος ώστε να επιτευχθεί η καλύτερη βηματική απόκριση [75].

Επιπλέον, για την ανίχνευση της συχνότητας της χορδής που διεγείρεται, γίνεται χρήση της μεθόδου αυτοσυσχέτησης YIN. Ουσιαστικά, οι νότες οι οποίες ακούγονται πιο ευχάριστα και γλυκά στο αυτί είναι αυτές που έχουν ακέραιο αριθμό συχνότητας. Όπως είναι γνωστό, μία οκτάβα χωρίζεται σε δώδεκα ίσα διαστήματα όπου το κάθε ένα από αυτά αποτελεί μία νότα. Το κάθε ένα διάστημα απέχει από το άλλο ένα ημίτονο και το κάθε ημίτονο αποτελείται από εκατό cents. Σύμφωνα με πειράματα, μία διαφορά μπορεί να γίνει αντιληπτή ανά έξι cents. Η λήψη της συχνότητας ενός σήματος μπορεί να γίνει αντιληπτή είτε στο πεδίο του χρόνου είτε στο πεδίο της συχνότητας. Στο πεδίο του χρόνου γίνεται χρήση της συνάρτησης αυτόματης συσχέτισης (AutoCorrelation Function ACF) η οποία πολλαπλασιάζει το σήμα με μία χρονική καθυστέρηση. Όταν όμως η καθυστέρηση του

χρόνου είναι ίδια με την περίοδο του σήματος τότε σε αυτό το σημείο εμφανίζεται τοπικό μέγιστο. Επομένως, μετά τον εντοπισμό της περιόδου του σήματος και γνωρίζοντας την συχνότητα δειγματοληψίας, μπορεί να υπολογιστεί η συχνότητα του σήματος. Επίσης, υπάρχουν πέντε βήματα τα οποία μπορούν να βελτιώσουν τη κανονική συνάρτηση αυτόματης συσχέτισης σε εφαρμογές μουσικής η οποία ονομάζεται YIN Autocorrelation. Τα πρώτα τέσσερα βήματα αναφέρονται στο αν η περίοδος του σήματος είναι πολλαπλάσια της περιόδου δειγματοληψίας, ενώ στο πέμπτο βήμα γίνεται χρήση παραβολικής παρεμβολής [75].

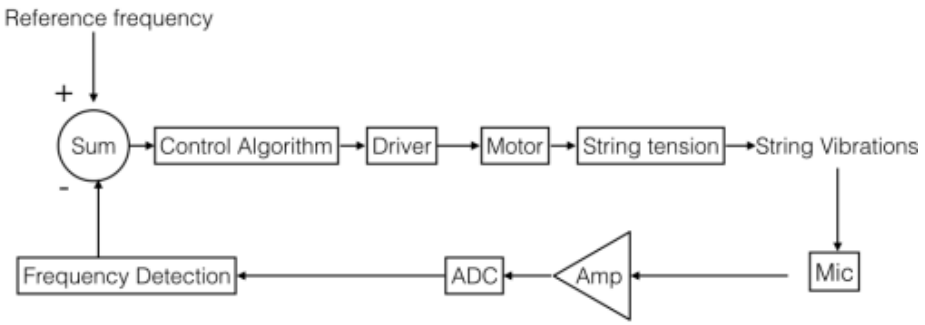


Εικόνα 1.52: Το τελικό σύστημα του Robotic Electric Guitar Tuner [99].

Ωστόσο, στο σύστημα γίνεται χρήση ενός PID ελεγκτή για τον έλεγχο των στροφών του κινητήρα, με σκοπό να ελαττωθεί το σφάλμα μεταξύ του επιθυμητού και του πραγματικού σήματος. Αυτό επιτυγχάνεται επηρεάζοντας αναλογικά το κέρδος του σήματος εξόδου. Ουσιαστικά, ένας PID ελεγκτής, αποτελείτε από έναν αναλογικό όρο (Proportional), έναν ολοκληρωτικό όρο (Integral) και έναν διαφορικό όρο (Derivative). Ο αναλογικός ελεγκτής (Proportional controller ή αλλιώς ελεγκτής τύπου P) περιλαμβάνει ένα αναλογικό κέρδος (K_p) το οποίο μειώνει τον χρόνο ανύψωσης μέχρι κάποιον βαθμό. Όμως, η υπερβολική αύξηση του αναλογικού κέρδους και αυξάνει την υπερύψωση του σήματος. Για την αντιμετώπιση της υπερύψωσης του σήματος προστίθεται και ο όρος ολοκλήρωσης (Integral) δημιουργώντας έτσι έναν ελεγκτή τύπου PI. Με την προσθήκη λοιπόν του ολοκληρωτικού κέρδους (K_i), η υπερύψωση μειώνεται και το μόνιμο σφάλμα εξαλείφεται. Όμως, με την υπερβολική αύξηση του αναλογικού και του ολοκληρωτικού κέρδους, το σύστημα οδηγείται στην αστάθεια. Για να μην βρεθεί το σύστημα σε αστάθεια, είτε θα γίνει μείωση του αναλογικού και του ολοκληρωτικού κέρδους είτε θα γίνει προσθήκη ενός διαφορικού όρου δηλαδή ενός διαφορικού κέρδους (K_D) δημιουργώντας έτσι τον ελεγκτή PID [75].

Το σύστημα κουρδίσματος Robot Tuner ουσιαστικά χωρίζεται στο κομμάτι της ενίσχυσης του σήματος και αναγνώρισης της συχνότητας του ήχου και στο κομμάτι ελέγχου της τάσης των χορδών της κιθάρας (Εικόνα 1.53). Πιο συγκεκριμένα, η συχνότητα του ήχου εντοπίζεται με την μέθοδο συσχέτισης YIN η οποία επεξεργάζεται από τον ελεγκτή PID και στην έξοδό του παράγεται ένα σήμα PWM (διαμόρφωση πλάτος παλμού) το οποίο καταλήγει στην γέφυρα τύπου «H» (H bridge) ενεργοποιώντας τον DC

κινητήρα. Επίσης, η συχνότητα θα πρέπει να παραχθεί όσο το δυνατόν πιο γρήγορα για να αποφευχθεί η υπερβολική ταλάντωση του PID ελεγκτή. Για να μπορέσει το σήμα που λαμβάνει το Arduino να είναι αξιόπιστο, θα πρέπει η ενίσχυση να έχει τουλάχιστον κέρδος δεκαπέντε διότι τα σήμα από τα μικρόφωνα είναι πολύ ασθενές και ο αναλογικό-ψηφιακός μετατροπέας του Arduino αντιλαμβάνεται μόνο σήματα από 0-5Volt. Γι' αυτό τον λόγο επιλέχθηκε ο ενισχυτής amp INA126PA και ένα ποτενσιόμετρο για την ρύθμιση του κέρδους. Επίσης, ο ενισχυτής θα προσθέσει μία DC τάση offset της τάξεως των 2.5Volt ώστε το Arduino να μπορέσει να διαβάσει την πλήρης κυματομορφή του σήματος. Για την offset ενίσχυση του σήματος επιλέχθηκε ο ενισχυτής amp CA3160E. Επιπλέον, το σήμα διέρχεται από ένα βαθυπερατό φίλτρο ώστε να μειωθεί ο θόρυβος και οποιαδήποτε σφάλματα από τις διάφορες αλλοιώσεις. Επιπροσθέτως, για την περιστροφή των κλειδιών της κιθάρας θα χρειαστεί αρκετή ροπή, επομένως ο κινητήρας θα πρέπει να συνδεθεί με ένα κιβώτιο ταχυτήτων. Ωστόσο, θα πρέπει να έχουν τέτοιο μέγεθος ώστε να χωρούν στο εσωτερικό της λαβής. Ακόμα, για να μπορέσει ο άξονας να περιστρέψει το κλειδί σχεδιάστηκε ένας αντάπτορας (Εικόνα 1.54), μέσω του προγράμματος Solid Edge ST7, ο οποίος εφαρμόζει από την μία πλευρά στον άξονα μετάδοσης κίνησης και η άλλη πλευρά στο κλειδί της κιθάρας [75].



Εικόνα 1.53: Το σχέδιο του συστήματος αυτομάτου ελέγχου για το Robotic Electric Tuner [100].

Εικόνα 1.54: Ο αντάπτορας που σχεδιάστηκε για να εφαρμόζει στον άξονα του κινητήρα και στα κλειδιά της κιθάρας για το Robotic Electric Guitar Tuner [101].

Όπως αναφέρθηκε προηγουμένως, η γέφυρα τύπου «Η» (H bridge) ελέγχει τον DC κινητήρα τόσο ως προς την ταχύτητα όσο και ως προς την φορά περιστροφής. Ουσιαστικά, για να το πετύχει αυτό η γέφυρα τύπου «Η» χρησιμοποιεί τέσσερα τρανζίστορ. Ωστόσο, η τροφοδοσία που απαιτεί η γέφυρα τύπου «Η» είναι στα 12 Volt [75].

Όσον αφορά το σήμα εισόδου στον αναλογικό-ψηφιακό μετατροπέα (ADC) του Arduino, όπως ειπώθηκε προέρχεται από τα μικρόφωνα και ο ρυθμός δειγματοληψίας του σήματος παίζει σημαντικό ρόλο στην συγκεκριμένη εφαρμογή. Για την εξασφάλιση του

σταθερού ρυθμού δειγματοληψίας, γίνεται χρήση διακοπών (Interrupts). Αυτό όμως έχει ως συνέπεια την καθυστέρηση της διεργασίας του κυρίως βρόγχου. Ουσιαστικά, με την χρήση διακοπών επιτυγχάνεται μεγαλύτερος έλεγχος στο ποσοστό δειγματοληψίας. Για την εύρεση της συχνότητας του σήματος, πρώτα τα δεδομένα εισόδου επεξεργάζονται με την χρήση της αθροιστικής συνάρτησης μέσης διαφοράς. Το αποτέλεσμα που προκύπτει εισάγεται σε μία διαδικασία ανίχνευσης κορυφής στην οποία γίνεται έλεγχος της τρέχουσας τιμής με την προηγούμενη. Αν η τρέχουσα τιμή είναι μικρότερη από την προηγούμενη, τότε η παράγωγος είναι αρνητική και οι τιμές από την συνάρτηση αθροιστική μέσης διαφοράς μειώνονται. Αν όμως η τρέχουσα τιμή είναι μεγαλύτερη από την προηγούμενη, τότε η παράγωγος είναι θετική και τότε η τετμημένη της προηγούμενης τιμής αναφέρεται στην περίοδο του σήματος. Με την χρήση μιας παραβολής που ταιριάζει με την τετμημένη, υπολογίζεται μία νέα τιμή της περιόδου η οποία εισάγεται στον PID ελεγκτή. Οι τιμές του αναλογικού, του ολοκληρωτικού και του διαφορικού όρου ρυθμίζονται με την μέθοδο δοκιμής και σφάλματος χωρίς την χρήση κάποιου μοντέλου για την εύρεση των βέλτιστων τιμών των παραμέτρων. Ουσιαστικά, ο ελεγκτής υπολογίζει το σφάλμα της επιθυμητής συχνότητας με την πραγματική συχνότητα και βγάζει ένα σήμα εξόδου το οποίο καταλήγει στην είσοδο του PID ελεγκτή. Ο PID ελεγκτής με την σειρά του παράγει ένα PWM σήμα με το οποίο ελέγχει την φορά και την ταχύτητα του κινητήρα. Αυτή η διαδικασία συνεχίζεται έως ότου η συχνότητα εισόδου πέσει μέσα στα όρια που έχουν καθοριστεί [75].

Επιπροσθέτως, γίνεται ένα πείραμα το οποίο έχει σκοπό την εύρεση του καλύτερου ρυθμού δειγματοληψίας με τον οποίο θα επιτευχθεί το ακριβέστερο κούρδισμα για όλες τις χορδές της κιθάρας. Ουσιαστικά, δίνεται ένα χρονικό όριο των 60 δευτερολέπτων ώστε να κουρδίσει το ρομπότ μία χορδή. Αν το ρομπότ μείνει σε αδράνεια για 5 δευτερόλεπτα, τότε το κούρδισμα της χορδής έχει ολοκληρωθεί. Επίσης, όλες οι χορδές για να έχουν ίδιες αρχικές συνθήκες κουρδίζονται δύο ημίτονα κάτω. Όταν ολοκληρωθεί ο χρόνος των 60 δευτερολέπτων ή αν ο κινητήρας μείνει σε αδράνεια για πάνω από 5 δευτερόλεπτα, τότε το η συχνότητα της χορδής μετριέται με παλμογράφο και η τιμή της αντιστοιχίζεται σε νότες. Σύμφωνα με τα αποτελέσματα που προέκυψαν, το σύστημα κατάφερε να κουρδίσει την κιθάρα σε οποιοδήποτε ξεκούρδισμα. Το πείραμα επαληθεύθηκε για έξι διαφορετικές συχνότητες δειγματοληψίας μεταξύ 10kHz και 35kHz. Πιο συγκεκριμένα για κάθε μία από τις έξι συχνότητες δειγματοληψίας, το κούρδισμα έγινε 5 φορές ώστε να προκύψει ένας μέσος όρος της συχνότητας για την κάθε χορδή. Ωστόσο, υπολογίστηκε και ο μέσος όρος των σφαλμάτων σε cents ώστε να βρεθεί ο ρυθμός δειγματοληψίας ο οποίος προσδίδει το πιο ακριβές κούρδισμα. Παρατηρήθηκε λοιπόν ότι στις χαμηλές συχνότητες δειγματοληψίας οι μπάσες χορδές (Μι ή E, Λα ή A & Ρε ή D) κουρδίζονται με μεγαλύτερη ακρίβεια σε σχέση με τις καντίνι χορδές (Σολ ή G, Σι ή B & Μι ή e). Αντίθετα, στις υψηλές συχνότητες δειγματοληψίας οι καντίνι χορδές κουρδίζονται με μεγαλύτερη ακρίβεια σε σχέση με την Μι μπάσο (E) και Λα (A) που έχουν μεγαλύτερο μέσο σφάλμα. Ουσιαστικά, όσο αυξάνεται ο ρυθμός δειγματοληψίας, τόσο μεγαλώνει και ο χρόνος ανάλυσης αλλά το συνολικό μέγεθος του παραθύρου που περιέχει το σήμα μειώνεται. Συνεπώς, με την ελάττωση του μεγέθους του παραθύρου που περιέχει το σήμα, η περίοδος που είναι απαραίτητη δεν υπάρχει και ο αλγόριθμος υπολογίζει λάθος συχνότητα. Ωστόσο, οι χαμηλότερες συχνότητες περιγράφονται με περισσότερα δείγματα και αυτό έχει ως αποτέλεσμα να γίνονται περισσότεροι υπολογισμοί για την εύρεση της περιόδου του σήματος. Αυτό έχει ως αποτέλεσμα την αργή ενημέρωση του σφάλματος το οποίο με την σειρά του οδηγεί τον κινητήρα σε υπερβολική περιστροφή ξεπερνώντας την επιθυμητή

τιμή της συχνότητας. Έτσι το σύστημα οδηγείτε σε αστάθεια και η μέθοδος αποτυγχάνει [75].

Επιπροσθέτως, θα μπορούσαν στο σύστημα να γίνουν κάποιες βελτιστοποιήσεις μελλοντικά. Πιο συγκεκριμένα, θα μπορούσε να γίνει βελτιστοποίηση των παραμέτρων του PID ελεγκτή, ή ακόμα καλύτερα κάθε χορδή να έχει τις δικές της παραμέτρους στον PID ελεγκτή. Επίσης, αντί για την χρήση του Arduino, θα μπορούσε να χρησιμοποιηθεί ένας αναλογικό-ψηφιακός μετατροπέας με περισσότερα bit ανάλυση, με μεγαλύτερο ρυθμό δειγματοληψίας και με μεγαλύτερη μνήμη ώστε να υπάρχει ακριβέστερη αναπαράσταση του σήματος. Επιπλέον, ένας επεξεργαστής για γρήγορες πράξεις κινητής υποδιαστολής θα έκανε τον PID ελεγκτή πιο γρήγορο. Καθώς επίσης θα μπορούσε να χρησιμοποιηθεί ένας πιο γρήγορος κινητήρας που θα παρέχει την απαραίτητη ροπή και ακρίβεια. Τέλος, θα μπορούσε να σχεδιαστεί ένα σύστημα το οποίο θα κρατάει σταθερό το σήμα για περισσότερο χρόνο αποφεύγοντας με αυτόν τον τρόπο τα σφάλματα [75].

1.2.5 **A review on guitars tuners**

Στην συγκεκριμένη δημοσίευση, αναφέρονται περιληπτικά κάποιες εργασίες που έχουν σχέση με το αυτόματο κούρδισμα της κιθάρας. Ουσιαστικά, με το κούρδισμα της κιθάρας γίνεται μεταβολή της τάσης των χορδών και κατά συνέπεια μεταβάλλεται η συχνότητα ταλάντωσης της κάθε χορδής. Αυτό με την σειρά του οδηγεί στην αλλαγή της τονικότητας του ήχου. Ωστόσο, το κούρδισμα για του αρχάριους κιθαρίστες είναι περίπλοκο και ειδικά όταν προσπαθούν να πειραματιστούν με εναλλακτικά κουρδίσματα. Σε αυτή την δημοσίευση εξετάστηκαν διάφορες μέθοδοι και τεχνικές για το κούρδισμα της κιθάρας [76].

Αρχικά, στην διπλωματική εργασία με τίτλο «Design and development of low cost automatic stringed instrument» κατασκευάζεται ένα σύστημα αυτομάτου ελέγχου για το κούρδισμα των χορδών της κιθάρας το οποίο αποτελείται από έναν μικροελεγκτή και υλικά χαμηλού κόστους τα οποία επιδεικνύουν υψηλή ακρίβεια για τον εντοπισμό των συχνοτήτων των χορδών του οργάνου. Επιπλέον, γίνεται χρήση μιας τεχνικής εντοπισμού της συχνότητας της κάθε χορδής μέσω των δονήσεων του οργάνου. Ουσιαστικά, με την ανάγνωση των δονήσεων του οργάνου, το σήμα δεν επηρεάζει την ακρίβεια της διάγνωσης. Επίσης, το μηχανολογικό σχέδιο του συστήματος συνδυάζει εργονομία και καλαισθησία μαζί με εξαιρετική λειτουργικότητα. Ωστόσο, τα ευρήματα της διπλωματικής εργασίας ανοίγουν δρόμους για περαιτέρω έρευνα πιο περίπλοκων μηχανισμών καθώς και για βελτίωση του ήδη υπάρχον συστήματος με την εισαγωγή νέων μουσικών κλιμάκων [76].

Εν συνεχεία, στην διπλωματική εργασία με τίτλο «A digital guitar tuner» δημιουργείτε ένας αλγόριθμος ο οποίος εντοπίζει την θεμελιώδη συχνότητα της χορδής που διεγείρεται καθώς και μία φιλική διεπαφή με την χρήση Matlab ώστε να μπορεί ο χρήστης να αλληλεπιδρά με το σύστημα. Το ψηφιακό κουρδιστήρι ουσιαστικά βασίζεται στο φιλτράρισμα του σήματος στην μέτρηση και στις ικανότητες ανάλυσης του Matlab. Οι μέθοδοι που αναλύονται μέσα στην εργασία για τον εντοπισμό της θεμελιακής συχνότητας της χορδής που διεγείρεται, βασίζονται στο γινόμενο των αρμονικών του φάσματος (Harmonic Product Spectrum). Ουσιαστικά, με την χρήση των αρμονικών συχνοτήτων των νοτών, ο υπολογισμός της θεμελιακής συχνότητας της χορδής που διεγείρεται γίνεται με μεγαλύτερη ακρίβεια διότι απλοποιείται ο αλγόριθμος ανίχνευσης. Ωστόσο, εντός της

εργασίας αναλύονται και επεξηγούνται παράγοντες οι οποίοι επηρεάζουν την διαδικασία εντοπισμού της θεμελιακής συχνότητας του σήματος. Επιπροσθέτως, μία σημαντική βελτίωση της συγκεκριμένης εργασίας, περιλαμβάνει την προσθήκη ενός κινητήρα ο οποίος θα περιστρέφει το κάθε κλειδί της κιθάρας κατά τέτοιον τρόπο ώστε να επιτευχθεί το κούρδισμα της χορδής [76].

Σε μία άλλη διπλωματική εργασία με τίτλο «Design and realization of the guitar tuner using myRIO» δημιουργείτε ένα ψηφιακό κουρδιστήρι το οποίο κάνει χρήση ενός μικροφώνου. Πιο συγκεκριμένα, το μικρόφωνο παρέχει μια συνεχόμενη αναλογική ροή του ήχου, δηλαδή ένα αναλογικό σήμα, το οποίο με την χρήση ενός αναλογικό-ψηφιακού μετατροπέα (ADC) μετατρέπεται σε ψηφιακό διακριτό σήμα. Επιπλέον, το ψηφιακό σήμα φιλτράρετε ώστε να αφαιρεθεί ο θόρυβος ο οποίος αλλοιώνει την πληροφορία. Ωστόσο, το αποτέλεσμα της επεξεργασίας του σήματος θα μπορούσε να εμφανιστεί σε ενδεικτικά LED ή σε μία οθόνη. Με αυτόν τον τρόπο θα εμφανίζεται το αποτέλεσμα στον χρήστη και αυτός με την σειρά του θα κουρδίζει την κάθε χορδή. Επίσης, με την δημιουργία του ψηφιακού κουρδιστηριού, φαίνεται ο τρόπος με τον οποίο ο μικροελεγκτής my RIO μπορεί να χρησιμοποιηθεί μέσω του περιβάλλοντος ανάπτυξης Lab View. Επιπροσθέτως, με τον κατάλληλο σχεδιασμό του ψηφιακού κουρδιστηριού θα μπορούσαν να κουρδιστούν κι άλλα μουσικά όργανα [76].

Ένα άλλο ενδιαφέρον άρθρο είναι το «ITUNE – Guitar tuner application for android driven mobile devices». Σκοπός της συγκεκριμένης εργασίας είναι η ανάπτυξη μίας εφαρμογής για Android λειτουργικά συστήματα, η οποία θα βοηθά τον χρήστη να κουρδίσει την κιθάρα του σε οποιονδήποτε τόνο. Η διεπαφή με την οποία αλληλεπιδρά ο χρήστης είναι εύχρηστη, λειτουργική και ευπαρουσίαστη. Για το στήσιμο της εφαρμογής, έγινε χρήση του ολοκληρωμένου περιβάλλοντος ανάπτυξης (IDE) στο οποίο χρησιμοποιήθηκε η γλώσσα προγραμματισμού JAVA και το Android SDK (software development kit) το οποίο παρέχει χρήσιμα εργαλεία για την ανάπτυξη Android εφαρμογών [76].

Μία άλλη εργασία που αναφέρεται στην συγκεκριμένη δημοσίευση είναι η «Design & Implementation of Digital Guitar Tuner Using Matlab». Σε αυτήν την εργασία μέσω του προγράμματος Simulink που παρέχει το Matlab, αναπτύσσεται μία λογική για τον εντοπισμό της θεμελιώδης συχνότητας μέσα από το φάσμα συχνοτήτων που εκπέμπει η χορδή που διεγείρεται. Ουσιαστικά, το πρόγραμμα καθορίζει αν η τονικότητα της χορδής είναι η επιθυμητή ή το αν θα πρέπει να ανέβει ή να κατέβει. Το μειονέκτημα είναι πως με το συγκεκριμένο πρόγραμμα μπορούν να κουρδιστούν μόνο έξι συγκεκριμένες νότες (E2, A2, D3, G3, B3 & E4) σε οποιοδήποτε όργανο [76].

Μια εξίσου ενδιαφέρον εργασία που περιγράφεται μέσα σε αυτή την δημοσίευση είναι η «Guitar tuner using cepstral analysis and fuzzy controller on arduino board». Σε αυτή την εργασία η θεμελιώδης συχνότητα της χορδής που διεγείρεται εντοπίζεται με την χρήση της ανάλυσης Cepstral. Η ανάλυση Cepstral χρησιμοποιείτε κυρίως για την επεξεργασία της ομιλίας, αλλά στην συγκεκριμένη περίπτωση δίνει καλά αποτελέσματα και για την επεξεργασία του ήχου στα μουσικά όργανα. Εν συνεχεία, η θεμελιώδης συχνότητα συγκρίνεται με την επιθυμητή και η διαφορά εισέρχεται σε έναν ασαφή ελεγκτή. Ο ασαφής ελεγκτής, με βάση του κανόνες που έχει δημιουργεί ένα σήμα PWM (Διαμόρφωση Πλάτος Παλμού) με συγκεκριμένο duty cycle ώστε να ελέγξει την φορά και την ταχύτητα περιστροφής ενός κινητήρα. Ο κινητήρας με την σειρά του περιστρέφει το

αντίστοιχο κλειδί της κιθάρας, μεταβάλλοντας την τάση της συγκεκριμένης χορδής. Έτσι η συχνότητα της χορδής μεταβάλλεται έως ότου φτάσει στον επιθυμητό τόνο [76].

Μια άλλη διπλωματική εργασία που αναφέρεται στην συγκεκριμένη δημοσίευση, είναι η «Laser Tuner: A Novel Approach to Pitch Detection on a Drumhead». Στην συγκεκριμένη διπλωματική εργασία κατασκευάζεται μία συσκευή φιλική προς τον χρήστη η οποία μετράει τις δονήσεις ενός τυμπάνου με την χρήση ανακλώμενου Laser. Ουσιαστικά, η συσκευή αυτή τοποθετείται γύρω από το τύμπανο παράγοντας ένα ορατό σήμα το οποίο ανακλάτε από το τύμπανο πίσω στην φωτοαντίσταση. Επίσης, σκοπός της εργασίας είναι να δείξει πως η μέτρηση των δονήσεων μέσω των Laser είναι σημαντικό και ακριβές εργαλείο για το κούρδισμα των τυμπάνων [76].

Εν συνεχεία, αναφέρεται η διπλωματική εργασία με τίτλο «Electronic Assisting Violin Tuner». Σε αυτή την εργασία, δημιουργείτε μία συσκευή για το κούρδισμα ενός βιολιού. Ουσιαστικά, ο ήχος του βιολιού διέρχεται μέσα από ένα φίλτρο Goertzel ώστε να εντοπιστεί η συχνότητα. Ανάλογα λοιπόν με την πληροφορία της συχνότητας και με την χρήση κατάλληλων ενεργοποιητών επιτυγχάνεται το κούρδισμα του βιολιού. Επιπλέον, πριν την ενίσχυση του σήματος έχει τοποθετηθεί ένα χαμηλοπερατό φίλτρο Chebyshev, ώστε να βελτιωθεί η αξιοπιστία του κουρδίσματος. Ακόμα, η συγκεκριμένη συσκευή παρέχει ασφάλεια ώστε κατά το κούρδισμα να μην σπάσει κάποια χορδή και τραυματιστεί ο βιολιστής, αλλά και ακρίβεια κουρδίσματος η οποία φτάνει το 1 Hz. Τέλος, αναφέρονται κάποιες βελτιώσεις της συγκεκριμένης συσκευής οι οποίες αφορούν την ταχύτητα του κουρδίσματος, την ανθεκτικότητα, το μέγεθος, το βάρος και την αξιοπιστία. Σημαντική βελτίωση θα ήταν το σύστημα να μπορεί να κουρδίζει τις τέσσερις χορδές του βιολιού ταυτόχρονα μειώνοντας σημαντικά τον χρόνο κουρδίσματος [76].

Μια άλλη εργασία που αναφέρεται στην συγκεκριμένη δημοσίευση έχει τίτλο «Mechanization of G, C and D chord playing system using servomotors for Acoustic Guitars». Σε αυτή την διπλωματική εργασία κατασκευάστηκε ένα σύστημα με την χρήση Arduino και σερβοκινητήρων για το πιάσιμο και το παίξιμο των συγχορδιών Σολ ματζόρε (G), Ντο ματζόρε (C) και Ρε ματζόρε (D). Το σύστημα αυτό μπορεί και παράγει μια συγκεκριμένη μελωδία καθώς επίσης μπορεί να επαληθεύσει τις συγχορδίες που παίζονται με την χρήση μεθόδων επεξεργασίας ήχου. Ουσιαστικά, με την χρήση αυτού του συστήματος, επιτυγχάνεται μείωση της ανθρώπινης προσπάθειας για το σωστό πιάσιμο των ακόρντων [76].

Επιπλέον, στην συγκεκριμένη δημοσίευση αναφέρεται και η διπλωματική εργασία με τίτλο «Guitar tuner and song performance evaluation using a NAO robot». Στην συγκεκριμένη διπλωματική εργασία δημιουργείτε ένα ρομπότ το οποίο διδάσκει μουσική ώστε τα παιδιά να βελτιώσουν τις ικανότητές τους στην ακουστική κιθάρα. Αυτό επιτυγχάνεται με τον συνδυασμό των χαρακτηριστικών του ρομπότ και τις τεχνικές επεξεργασίας σήματος. Ωστόσο, το συγκεκριμένο ρομπότ περιλαμβάνει κουρδιστήρι κιθάρα και αξιολόγηση της απόδοσης τραγουδιών. Από δοκιμές προέκυψε ότι το κουρδιστήρι ανταποκρίνεται πολύ καλά ακόμη και σε θορυβώδη περιβάλλοντα, αλλά δύσκολα ανιχνεύει νότες που έχουν αρμονικές κοντά στην περιοχή συχνοτήτων των 680Hz. Επιπροσθέτως, το ρομπότ μπορεί να ηχογραφήσει και να συμβαδίσει με τον ρυθμό του τραγουδιού. Οι δοκιμές του ρομπότ δεν έγιναν σε παιδιά, πάραυτα θα μπορούσε μέσα από κάποια εμπειρικά τεστ σε μαθητές να διαπιστωθεί η πρόοδός τους και να γίνει κάποια τροποποίηση ή όχι στην αλληλεπίδραση μεταξύ ρομπότ και μαθητή [76].

Μία άλλη διπλωματική που αναφέρεται στην συγκεκριμένη δημοσίευση, έχει τίτλο «Fuzzy automatic guitar tuner». Σε αυτήν την εργασία, δημιουργείτε ένα αυτόματο κουρδιστήρι κιθάρας κάνοντας χρήση ενός ασαφή ελεγκτή. Ουσιαστικά, για την δημιουργία του ασαφή ελεγκτή γίνεται χρήση του Simulink που παρέχει το Matlab, ενώ για τον υπολογισμό της θεμελιακής συχνότητας και των αρμονικών καθώς και για την σύγκριση με το επιθυμητό μοτίβο χρησιμοποιώντας FFT (Fast Fourier Transform) γίνεται χρήση του xPC real-time kernel (Simulink real-time). Η διαφορά που προκύπτει από την σύγκριση, εισάγεται στον ασαφή ελεγκτή ο οποίος παράγει στην έξοδό του κατάλληλο σήμα για την ρύθμιση της τάσης της χορδής. Μετά από δοκιμές διαπιστώθηκε ότι υπάρχει ένα μικρό περιθώριο σφάλματος μιας και ο τόνος ξεφεύγει ελαφρά χωρίς να ακούγεται άσχημα η νότα [76].

Στην συγκεκριμένη δημοσίευση, αναφέρεται μία πολύ ενδιαφέρον διπλωματική εργασία η οποία έχει τίτλο «Optimiztion of intelligent tuning system for stringed instruments based on wireless sensors network». Πιο συγκεκριμένα, σε αυτή την εργασία, προτείνεται ένα σύστημα αυτομάτου κουρδίσματος βασισμένο σε ασύρματα δίκτυα αισθητήρων. Το υλισμικό (Hardware) του συστήματος, αποτελείται από την μονάδα λήψης ήχου, την μονάδα φωνητικής εκπομπής, την μονάδα επιλογής και ανάλυσης επικοινωνίας των δικτύων καθώς και του κυκλώματος οδήγησης της μονάδας που ελέγχει το κούρδισμα. Για το λογισμικό (Software) του συστήματος είναι απαραίτητη η σχεδίαση του ενσωματωμένου λογισμικού, του λογισμικού του κινητού, των ροών του συστήματος και της λειτουργίας του αυτόματου κουρδιστηριού. Επιπλέον, για μεγαλύτερη ακρίβεια γίνεται χρήση της βελτιωμένης βραχύχρονης μεθόδου εντοπισμού του μέσου πλάτους της κορυφής (improved short-time average amplitude endpoint detection method). Ωστόσο, για την αντιμετώπιση της ακρίβειας του κουρδίσματος σε όλο το φάσμα των συχνοτήτων το οποίο προκύπτει από την μέθοδο ανίχνευσης του τόνου, γίνεται χρήση της προσέγγισης της αρμονικής κορυφής. Ουσιαστικά, το σύστημα αυτό παρέχει αυτόματο κούρδισμα, περιέλιξη, φωνητική μετάδοση, ασύρματο έλεγχο, οθόνη και μετρονόμο για καλύτερη αλληλεπίδραση μεταξύ χρήστη και συστήματος [76].

Μια άλλη διπλωματική εργασία που αναφέρεται εντός της συγκεκριμένης δημοσίευσης έχει τίτλο «An automatic multi-string musical instrument tuner using one-to-many micro actuating mechanism». Πιο συγκεκριμένα, επινοήθηκε ένας μηχανισμός ο οποίος αποτελείται από μικροενεργοποιητές για το κούρδισμα τριών πρίμων χορδών ενός Ταϊλανδέζικου οργάνου. Ακόμα, ο μηχανισμός περιλαμβάνει έναν μαγνητικό αισθητήρα ο οποίος προσδιορίζει και μετρά το πλάτος του σήματος όταν μία χορδή διεγερθεί. Στην συνέχεια, το σήμα επεξεργάζεται από τον αλγόριθμο Gortzel ο οποίος υπολογίζει την συχνότητα των χορδών. Μετά τον υπολογισμό της συχνότητας οι μονάδες των μικροενεργοποιητών ρυθμίζουν την τάση των χορδών και η διαδικασία επαναλαμβάνεται έως ότου οι χορδές φτάσουν την επιθυμητή συχνότητα. Ο μαγνητικός αισθητήρας που χρησιμοποιείται είναι τόσο μικρός ο οποίος μπορεί να διακρίνει παλμούς ακόμα και όταν οι χορδές απέχουν μεταξύ τους 3 mm. Ωστόσο μπορεί να μετρά ταυτόχρονα ή μεμονωμένα τα σήματα όλων των χορδών χωρίς να επηρεάζεται από τον θόρυβο. Επίσης, οι τρεις χορδές του οργάνου μπορούν να κουρδιστούν ταυτόχρονα ή ξεχωριστά στην ίδια συχνότητα χρησιμοποιώντας κατάλληλα του κινητήρες [76].

Εν συνεχεία, στην συγκεκριμένη δημοσίευση αναφέρεται μία άλλη διπλωματική εργασία με τίτλο «Arduino based automatic guitar tuning system». Ουσιαστικά, σε αυτή την εργασία κατασκευάστηκε ένας μηχανισμός βασισμένος στο Arduino για το κούρδισμα

μιας κιθάρας. Αρχικά, ο ήχος ηχογραφείτε από ένα μικρόφωνο και υπολογίζεται η συχνότητα του σήματος. Έπειτα, το Arduino παράγει κατάλληλα σήματα ώστε να ελέγξει την περιστροφή ενός βηματικού κινητήρα ο οποίος θα περιστρέψει τα κλειδιά. Για να μπορέσει να γίνει το σύστημα αυτό πιο πρακτικό θα πρέπει να βελτιστοποιηθεί η στερέωση της μπαταρίας και του βηματικού κινητήρα [76].

Μια άλλη διπλωματική εργασία που αναφέρεται στην συγκεκριμένη δημοσίευση έχει τίτλο «Development of a triple input musical instrument tuner using Yin algorithm». Ουσιαστικά, σε αυτή την εργασία δημιουργείται ένα χαμηλού κόστους σύστημα με βέλτιστη ανίχνευση τόνου το οποίο τρέχει σε ένα smartphone με Android λειτουργικό σύστημα. Πιο συγκεκριμένα, το σύστημα αυτό εντοπίζει και επεξεργάζεται σήματα από διάφορα μουσικά όργανα αρκεί οι συχνότητές τους να είναι από 27.5Hz έως 4186Hz. Επιπλέον, περιγράφονται διάφορες μέθοδοι οι οποίες συνδυάζονται ώστε να ληφθεί ο ήχος από διάφορα μουσικά όργανα σε μία μόνο συσκευή. Αυτή η συσκευή επωφελή ένα γκρουπ στο οποίο υπάρχουν διαφορετικά όργανα διότι μπορούν να τα κουρδίσουν χρησιμοποιώντας την ίδια συσκευή. Μία μελλοντική βελτίωση θα ήταν η χρήση μιας εσωτερικής μπαταρίας στο σύστημα [76].

Μία εξίσου ενδιαφέρον διπλωματική εργασία που αναφέρεται στην συγκεκριμένη δημοσίευση έχει τίτλο «Comparative study of digital signal processing techniques for tuning an acoustic guitar». Πιο συγκεκριμένα, στην εργασία αυτή συγκρίνονται δύο διαφορετικές τεχνικές για το κούρδισμα μίας ακουστικής κιθάρας, ο μετασχηματισμός Fourier (Fast Fourier Transform) και η εκτίμηση πυκνότητας φάσματος (Spectral Density Estimation). Αρχικά, ο ήχος της κάθε χορδής της κιθάρας ηχογραφείτε και αποθηκεύεται σε αρχεία WAV τα οποία εισέρχονται στον αλγόριθμο ο οποίος είναι γραμμένος σε Python και στο Matlab. Στην συγκεκριμένη εργασία γίνεται εκτίμηση πέντε θεμελιακών συχνοτήτων και γίνεται ανάλυση της επεξεργασίας του κουρδίσματος για κάθε μία από τις δυο τεχνικές συγκρίνοντας την ακρίβεια της κάθε μίας. Σύμφωνα με τα αποτελέσματα, η τεχνική της φασματικής πυκνότητας ισχύος (Power Spectral Density) είναι πιο γρήγορη και ακριβής σε σχέση με τον μετασχηματισμό Fourier (Fast Fourier Transform) [76].

Επιπροσθέτως, στην συγκεκριμένη δημοσίευση, αναφέρεται μια διπλωματική εργασία η οποία έχει έναν εντελώς διαφορετικό τρόπο προσέγγισης για την κατασκευή ενός αυτόματου κουρδιστηριού και έχει τίτλο «Real time implementation of a tuning device using a digital signal processor». Ουσιαστικά, σχεδιάζεται ένα κουρδιστήρι βασισμένο στην ψηφιακή επεξεργασία σήματος (DSP) για την επίτευξη καλύτερων αποτελεσμάτων. Πιο συγκεκριμένα, το πρόγραμμα αναπτύσσεται σε γλώσσα προγραμματισμού C στο TMS320C5402 DSP Starter Kit κάνοντας χρήση DSPρουτίνων συναρμολόγησης της Texas Instruments. Επίσης, στο περιβάλλον ανάπτυξης (IDE) Code Composer Studio (CCS) που παρέχει η Texas Instruments εκτελέστηκε το λογισμικό του κουρδιστηριού [76].

Μια άλλη εξίσου ενδιαφέρον διπλωματική εργασία που αναφέρεται μέσα σε αυτήν τη δημοσίευση έχει τίτλο «Comparison of fundamental frequency detection methods and introducing simple self-repairing algorithm for musical applications». Ουσιαστικά, στην συγκεκριμένη εργασία γίνεται σύγκριση πέντε προσεγγίσεων ανίχνευσης της θεμελιώδης συχνότητας για σήματα φωνητικών και μελωδικών οργάνων, ενώ η αποτελεσματικότητα της κάθε μίας προσέγγισης ελέγχεται μέσα από μία σειρά νοτών ενός μπάσου κλαρινέτου. Τα αποτελέσματα της σύγκρισης, έδειξαν ότι η συνάρτηση αυτόματης συσχέτισης (AutoCorrelation ACF) και τροποποιημένη συνάρτηση αυτόματης συσχέτισης (Modified

AutoCorrelation MACF) έχουν την μέγιστη απόδοση για την ανίχνευση της θεμελιώδης συχνότητας. Οι δύο αυτές προσεγγίσεις, απέτυχαν να εντοπίσουν την θεμελιώδη συχνότητα μόνο σε ένα μικρό τμήμα, πράγμα που σημαίνει ότι μπορούν να χρησιμοποιηθούν σε πραγματικό χρόνο. Από την άλλη, η φασματική προσέγγιση έδωσε τα χειρότερα αποτελέσματα διότι είναι κατάλληλη μόνο για μεγάλες δειγματοληψίες. Ένα σημαντικό εργαλείο είναι ο αλγόριθμος αυτοεπιδιόρθωσης θεμελιωδών συχνοτήτων ο οποίος σε συνδυασμό με την συνάρτηση αυτόματης συσχέτισης (ACF) και την τροποποιημένη συνάρτηση αυτόματης συσχέτισης (MACF) καθώς και με μία συνάρτηση μετατόπισης της συχνότητας δημιουργείτε ένα εργαλείο χαμηλής υπολογιστικής απόδοσης σχεδόν πραγματικού χρόνου τόσο για κούρδισμα, όσο και για άλλες μουσικές εφαρμογές. Η τεχνική της αυτοεπιδιόρθωσης αποσκοπεί στην μείωση του λόγου σφάλματος της πραγματική θεμελιακής συχνότητας και θα μπορούσε να τροποποιηθεί μελλοντικά για μεγαλύτερη ακρίβεια. Ο συνδυασμός της μεθόδου αυτόματης συσχέτισης (ACF) ή της τροποποιημένης μεθόδου αυτόματης συσχέτισης (MACF) αποτελούν ισχυρά εργαλεία για την ανάλυση τόσο της φωνής και των μουσικών οργάνων, όσο και των συναισθημάτων μέσα από την ομιλία [76].

Μια άλλη διπλωματική εργασία που αναφέρεται στην συγκεκριμένη δημοσίευση για αυτόματα κουρδιστήρια, έχει τον τίτλο «Pitch detection algorithms based on zero-cross rate and autocorrelation function for musical notes». Ουσιαστικά, σε αυτή την διπλωματική εργασία συγκρίνονται δύο αλγόριθμοι για τον εντοπισμό της τονικότητας. Πιο συγκεκριμένα, στον έναν αλγόριθμο γίνεται χρήση της συνάρτησης zero-cross rate (ZCR), ενώ στον άλλον αλγόριθμο γίνεται χρήση της συνάρτησης αυτοσυσχέτισης. Επιπλέον, αναφέρονται τα οφέλη, τα μειονεκτήματα και οι ενισχύσεις για την βελτίωση της απόδοσής τους. Οι δύο αυτοί αλγόριθμου δοκιμάστηκαν με μουσικές νότες από μία ακουστική κιθάρα και τα αποτελέσματά τους συγκρίθηκαν με κάποιες τιμές αναφοράς. Έπειτα, εφαρμόστηκαν μεθοδολογίες για την βελτίωση της απόδοσης των αλγορίθμων και συγκρίνοντας τα αποτελέσματα που προέκυψε σημαντική μείωση στις ανακρίβειες [76].

Η τελευταία διπλωματική εργασία που αναφέρεται σε αυτή την δημοσίευση έχει τίτλο «Frequency analysis of acoustic signal using the Fast Fourier Transformation in Matlab». Πιο συγκεκριμένα σε αυτή την διπλωματική εργασία γίνεται καταγραφή ακουστικών σημάτων σε ψηφιακά αρχεία, των οποίων τα δεδομένα μετασχηματίζονται κατά Fourier (Fast Fourier Transform) ώστε να γίνει ανάλυση των μηχανικών δονήσεων. Από τα πειράματα που έγιναν, προέκυψε μια χρονικά εξαρτώμενη σειρά διαφορετικών φυσικών χαρακτηριστικών που χρησιμοποιούνται για την διερεύνηση των μηχανικών ιδιοτήτων του συστήματος [76].

1.2.6 **Automatic guitar tuner**

Σε αυτή την διπλωματική εργασία, ερευνάται η επιστήμη και η τεχνολογία πίσω από τον ήχο. Πιο συγκεκριμένα, η καλή και η σωστή ακουστική του ήχου ξεκινά από το σωστό κούρδισμα του οργάνου. Γι’ αυτό τον λόγο, στο συγκεκριμένο πρότζεκτ γίνεται χρήση της τεχνολογίας για την ανάλυση του ήχου και ώστε να επιτευχθεί το κούρδισμα των χορδών ενός μπάσου με ακρίβεια και με ελάχιστη προσπάθεια [77].

Αρχικά, το σύστημα που δημιουργήθηκε για την αυτό-κουρδιζόμενη κιθάρα αποτελείται από πέντε υποσυστήματα τα οποία είναι αλληλένδετα. Πιο συγκεκριμένα, αποτελείται από του μαγνήτες οι οποίοι αποσκοπούν στην λήψη του σήματος, τον

μικροελεγκτή, την τροφοδοσία και την διεπαφή μεταξύ συστήματος και χρήστη. Ουσιαστικά, με την χρήση ενός smartphone, ο χρήστης θα συνδέεται με τον μικροελεγκτή μέσω Bluetooth και θα επιλέγει τις ρυθμίσεις και τις ιδιότητες του συστήματος. Η λογική πίσω από την λειτουργία του συστήματος ξεκινάει με τους μαγνήτες οι οποίοι παράγουν ένα σήμα από την χορδή που διεγείρεται και η πληροφορία του σήματος αναλύεται από τον μικροελεγκτή. Στην συνέχεια, ο μικροελεγκτής ανάλογα με τις προτιμήσεις που έχουν εισαχθεί στέλνει το κατάλληλο σήμα στους κινητήρες που χρησιμοποιούνται για τον έλεγχο της τάσης των χορδών. Ουσιαστικά, οι κινητήρες σφίγγουν ή ξεσφίγγουν το κλειδί της αντίστοιχης χορδής αλλάζοντας την συχνότητα ταλάντωσής της και κατά συνέπεια την νότα που σχετίζεται με αυτήν. Κατά την διάρκεια της επεξεργασίας, οι ενισχυτές συνεχίζουν να συλλέγουν την συχνότητα της χορδής στέλνοντας την πληροφορία στον μικροελεγκτή, ο οποίος σταματάει την περιστροφή των κινητήρων όταν επιτευχθεί η επιθυμητή συχνότητα. Η διαδικασία αυτή επαναλαμβάνεται για όλες τις χορδές [77].

Οι έμπειροι κιθαρίστες μπορούν να κουρδίσουν μία κιθάρα μέσα σε λίγα λεπτά, επομένως το κούρδισμα με την αυτόματη συσκευή θα πρέπει να είναι πιο γρήγορο. Θα πρέπει περίπου για το κούρδισμα της κάθε χορδής να γίνεται εντός 30 sec ώστε να μην χρειαστεί να κουρδιστεί η κιθάρα χειροκίνητα. Από την άλλη, για έναν αρχάριο το κούρδισμα παίρνει πάρα πολύ ώρα, συνεπώς με την συσκευή αυτή οι αρχάριοι δεν θα σπαταλούν την ώρα τους προσπαθώντας να κουρδίσουν το όργανο. Ακόμα, η συσκευή θα πρέπει να παρέχει ακρίβεια στο κούρδισμα, δηλαδή το ποσοστό διαφοράς από την επιθυμητή συχνότητα πρέπει να είναι μικρότερο από ±5cents που είναι η ελάχιστη διαφορά που μπορεί να αντιληφθεί το ανθρώπινο αυτί. Επίσης, το σύστημα θα πρέπει να είναι μικρό και βολικό πάνω στο όργανο ώστε να μην επηρεάζει τον χρήστη κατά το παίξιμο του οργάνου. Επιπλέον, τα εξαρτήματα θα πρέπει να είναι οικονομικά, προσιτά και να ταιριάζουν με το σύστημα, γι αυτό τον λόγο γίνεται έρευνα για την εύρεση των καλύτερων εξαρτημάτων [77].

Σε κάθε χορδή του οργάνου αντιστοιχεί μια συγκεκριμένη νότα και θα πρέπει να κουρδίζεται στην αντίστοιχη συχνότητα. Ουσιαστικά, ο χρήστης μέσα από το smartphone έχει την δυνατότητα να επιλέξει ένα από τα πέντε διαφορετικά κουρδίσματα της κιθάρας (Standard, Drop D, Drop B, Semitone flat standard & Semitone flat Drop D). Για την σωστή λειτουργία του συστήματος, θα πρέπει η θερμοκρασία να είναι γύρω στους 21^{o}C με εύρος ±10^{o}C και να βρίσκεται σε ξυρό περιβάλλον χωρίς υψηλή υγρασία. Επίσης, η τροφοδοσία του συστήματος γίνεται με επαναφορτιζόμενες μπαταρίες, οι οποίες θα πρέπει με μία φόρτιση να από 50-100 χρήσεις του συστήματος [77].

Η συσκευή, περιλαμβάνει ένα custom PCB (Printed Circuit Board) το οποίο περιλαμβάνει τον μικροεπεξεργαστή με εισόδους για την λήψη ρου σήματος, την μονάδα σύνδεσης Bluetooth και μία γέφυρα τύπου «Η» για τον έλεγχο των κυκλωμάτων των κινητήρων. Η γέφυρα τύπου «Η» θα πρέπει να είναι απομονωμένη από τον μικροεπεξεργαστή, διότι μπορεί να προκαλέσει ανεπιθύμητο θόρυβο και απότομες αυξήσεις της τάσης [77].

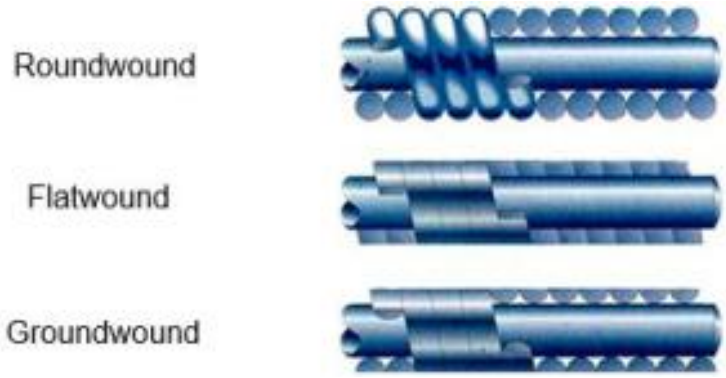
Ένα αντίστοιχο σύστημα αυτόματου κουρδίσματος είναι τοποθετημένο στο πίσω μέρος της κεφαλής σε κάποιες κιθάρες της η εταιρία Gibson. Πιο συγκεκριμένα, με αυτό το σύστημα το κούρδισμα ολοκληρώνεται μέσα σε λίγα δευτερόλεπτα είτε διεγείροντας τις χορδές όλες μαζί είτε μία-μία ξεχωριστά. Επιπλέον, περιλαμβάνει κάποιες LED ενδείξεις οι οποίες δείχνουν την διαδικασία κουρδίσματος δηλαδή ποιες χορδές είναι κουρδισμένες και ποιες πρέπει να διεγερθούν ξανά. Ακόμα, περιλαμβάνει κάποια κουμπιά τα οποία

αποτελούν την διεπαφή μεταξύ χρήστη και συστήματος. Ουσιαστικά, με τα κουμπιά αυτά ο χρήστης μπορεί να ενεργοποιήσει ή να απενεργοποιήσει το σύστημα και να επιλέξει το επιθυμητό κούρδισμα [77].

Για την διεκπεραίωση της συγκεκριμένης διπλωματικής μελετήθηκαν κάποιες εργασίες για το πως μπορεί κανείς να σχεδιάσει και να κατασκευάσει ένα τέτοιο σύστημα. Μία από αυτές τις εργασίες έχει τίτλο «The Mechanix Guitar Tuner» και πραγματοποιήθηκε στο Cornell University. Ουσιαστικά, στην εργασία αυτή κατασκευάστηκε ένα κουρδιστήρι για εξάχορδες κιθάρες είτε ακουστικές είτε ηλεκτρικές με συγκεκριμένες διαστάσεις της γέφυρας. Μια άλλη εργασία από το Clavin College έχει τίτλο «Squad SMARTune» και παρουσιάζει ένα σύστημα το οποίο κουρδίζει ένα τετράχορδο μπάσο. Το σύστημα αυτό ενσωματώνεται στην κεφαλή του μπάσου και αποτελείται από πέντε υποσυστήματα ενώ μπορεί να ολοκληρώσει το κούρδισμα εντός 15sec ή και λιγότερο. Ακόμα, δίνει την δυνατότητα στον χρήστη να επιλέξει το κούρδισμα που επιθυμεί μέσα από ένα εύρος έτοιμων κουρδισμάτων [77].

Ωστόσο, για την δημιουργία ενός συστήματος αυτόματου κουρδίσματος θα πρέπει κανείς να γνωρίζει τις νότες και τα διαστήματα μεταξύ των νοτών. Πιο συγκεκριμένα, στην μουσική ένα διάστημα ονομάζεται η διαφορά μεταξύ δύο συχνοτήτων. Οι διαφορές μεταξύ των συχνοτήτων είναι αντιληπτές από το ανθρώπινο αυτί και καθορίζονται από μία αναλογία. Η μονάδα μέτρησης των διαστημάτων αυτών είναι το Cent. Πιο συγκεκριμένα, ένα ημίτονο (Semitone) αποτελείται από 100Cents, ενώ μια οκτάβα από 1200Cents. Η μικρότερη διαφορά που μπορεί να αντιληφθεί το ανθρώπινο αυτή είναι τα 5Cents [77].

Επιπροσθέτως, στα έγχορδα όργανα, τα κατασκευαστικά χαρακτηριστικά των χορδών παίζουν σημαντικό ρόλο στον τόνο που παράγεται. Πιο συγκεκριμένα, η διάμετρος και το βάρος της χορδής είναι οι σημαντικότεροι παράγοντες που επηρεάζουν τον τόνο. Οι μπάσες χορδές στις κιθάρες (Μι μπάσο, Λα & Ρε) καθώς και οι χορδές του μπάσου αποτελούνται από έναν ατσάλινο πυρήνα περιελιγμένο είτε από ατσάλι είτε από νίκελο. Υπάρχουν τρεις τύποι περιέλιξης του ατσάλινου πυρήνα, ο Roundwound, ο Flatwound και ο Groundwound (Εικόνα 1.55). Ο Roundwound τύπος περιτύλιξης προσφέρει έναν λαμπερό και δυνατό ήχο, αλλά φθείρει τα τάστα με την πάροδο του χρόνου. Η περιέλιξη τύπου Flatwound παράγει έναν πιο απαλό, γλυκό και στρογγυλό ήχο, ενώ φθείρει λιγότερο τα τάστα με την πάροδο του χρόνου. Από την άλλη η περιέλιξη τύπου Groundwound είναι μια υβριδική μορφή που συνδέει του τύπους Roundwound και Flatwound. Με αυτό τον τρόπο παρέχεται λαμπερός ήχος εξαιτίας της περιέλιξης Roundwound ενώ τα τάστα φθείρονται λιγότερο εξαιτίας της περιέλιξης Flatwound [77].



Εικόνα 1.55: Οι τρεις τύποι περιέλιξης των χορδών Roundwound, Flatwound & Groundwound [102].

Ένα ακόμα σημείο που πρέπει να ληφθεί υπ' όψιν για την σχεδίαση του αυτόματου κουρδιστηριού είναι ο τύπος κινητήρων που θα χρησιμοποιηθεί για την περιστροφή των κλειδιών. Ένα είδος κινητήρων είναι οι σερβοκινητήρες, οι οποίοι κατατάσσονται στους περιστροφικούς ενεργοποιητές δίνοντας την δυνατότητα ελέγχου της γωνίας περιστροφής, της ταχύτητας και της επιτάχυνσης με ακρίβεια. Πιο συγκεκριμένα, οι σερβοκινητήρες αποτελούνται από έναν σερβομηχανισμό σε σύστημα κλειστού βρόγχου όπου στην είσοδό του λαμβάνει την επιθυμητή γωνία περιστροφής και με έναν αισθητήρα θέσεως ελέγχει την τρέχουσα περιστροφή του κινητήρα. Το σήμα εισόδου μπορεί να είναι είτε αναλογικό είτε ψηφιακό και καθορίζει την επιθυμητή τιμή που θα πρέπει να έχει στην έξοδό του το σύστημα κλειστού βρόγχου. Η διαφορά μεταξύ της επιθυμητής γωνίας περιστροφής και της τρέχουσας θέσης περιστροφής αποτελεί το σήμα σφάλματος και υπολογίζεται από έναν μικροελεγκτή. Οι σερβοκινητήρες που παρέχουν ανατροφοδότηση της θέσης και της ταχύτητας περιέχουν έναν κωδικοποιητή (encoder). Για την συγκεκριμένη εφαρμογή, αρκούν οι απλοί σερβοκινητήρες που ελέγχουν μόνο την θέση. Ουσιαστικά, συγκρίνεται η επιθυμητή τιμή εξόδου με την πραγματική τιμή και η διαφορά αποτελεί το σήμα σφάλματος το οποίο ο κινητήρας προσπαθεί να το εξαλείψει. Στην συγκεκριμένη εφαρμογή, μόλις ο κωδικοποιητής καταγράψει ότι το σήμα σφάλματος είναι μηδέν, τότε η κιθάρα είναι κουρδισμένη. Ωστόσο, υπάρχουν σερβοκινητήρες οι οποίοι χρησιμοποιούν την ανατροφοδότηση και της θέσης και της ταχύτητας του κινητήρα αυξάνοντας την ακρίβεια αλλά και το κόστος [77].

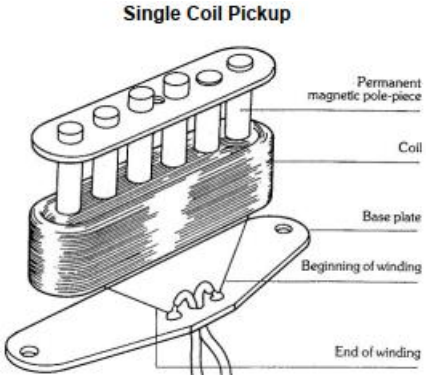
Ένας άλλος τύπος κινητήρων είναι οι βηματικοί κινητήρες οι οποίοι είναι DC κινητήρες χωρίς ψήκτρες και χωρίζουν την περιστροφή σε ένα συγκεκριμένο αριθμό ίσων βημάτων. Ουσιαστικά, το μέγεθος των βημάτων ποικίλοι σε μοίρες και υπάρχει ένα μεγάλο εύρος επιλογών. Το πόσο θα περιστραφεί ο άξονας καθορίζεται από ένα πλήθος αριθμών τετραγωνικών παλμών, δηλαδή κάθε παλμός κινεί τον άξονα αναλόγως. Πιο συγκεκριμένα, οι βηματικοί κινητήρες ελέγχονται από οδοντοτούς ηλεκτρομαγνήτες οι οποίοι τοποθετούνται γύρω από ένα γρανάζι. Κάθε φορά που δίνεται ενέργεια σε έναν ηλεκτρομαγνήτη για μια μικρή περιστροφή, ο προηγούμενος απενεργοποιείται. Αυτή η διαδικασία αντιπροσωπεύει ένα βήμα από το σύνολο των βημάτων για την περιστροφή του άξονα 360^{o}. Οι βηματικοί κινητήρες χωρίζονται σε δύο βασικές κατηγορίες ανάλογα με το είδος περιέλιξης. Η μία κατηγορία είναι οι μονοπολικοί κινητήρες οι οποίοι περιέχουν μονή περιέλιξη με κεντρική λήψη και η μία από τις δύο πλευρές έχει τη δική του μαγνητική κατεύθυνση. Ουσιαστικά, στους μονοπολικούς κινητήρες δεν χρειάζεται να αλλάξει η φορά του ρεύματος ώστε να αλλάξει φορά περιστροφής ο άξονας, αλλά αυτό καθορίζεται από έναν μικροελεγκτή. Η άλλη κατηγορία είναι οι διπολικοί κινητήρες οι οποίοι έχουν μονή περιέλιξη ανά φάση χωρίς κεντρική λήψη. Στους διπολικούς κινητήρες για να αλλάξει φορά περιστροφής ο άξονας, το ρεύμα στην περιέλιξη θα πρέπει να έχει αντίθετη φορά ώστε να αντιστραφεί ο μαγνητικός πόλος. Αυτό έχει ως αποτέλεσμα να απαιτείται ένα περίπλοκο κύκλωμα οδήγησης. Για την συγκεκριμένη εφαρμογή, αρκεί ο μονοπολικός κινητήρας [77].

Ένας πιο συνηθισμένος τύπος κινητήρων είναι οι DC κινητήρες οι οποίοι περιλαμβάνουν όμοιους μαγνήτες οι οποίοι απωθούνται και ανόμοιους μαγνήτες οι οποίοι έλκονται. Πιο συγκεκριμένα, όταν το ρεύμα διαρρέει το πηνίο τότε στο κέντρο του δημιουργείται ένα περιστρεφόμενο μαγνητικό πεδίο. Το ρεύμα με το μαγνητικό πεδίο είναι σύγχρονα, δηλαδή με θετική κατεύθυνση του ρεύματος, το μαγνητικό πεδίο ενεργοποιείτε προς την μία κατεύθυνση, ενώ με αρνητική φορά ρεύματος υο μαγνητικό πεδίο

ενεργοποιείτε προς την αντίθετη κατεύθυνση. Οι DC κινητήρες στην απλούστερή τους μορφή αποτελούνται από δύο ή περισσότερα τυλίγματα στον οπλισμό και από δύο σταθερούς μαγνήτες στον στάτορα. Το πηνίο τυλίγεται πάνω σε μονωμένες υποδοχές γύρω από σιδερένιους πόλους, ενώ οι άκρες των καλωδίων προσκολλώνται στον συλλέκτη. Επιπλέον, στον οπλισμό τοποθετούνται ρουλεμάν τα οποία κρατάνε κεντραρισμένο τον κινητήρα, τον άξονα και τις συνδέσεις στον συλλέκτη. Η τροφοδοσία του πηνίου καθορίζει την κατεύθυνση του ηλεκτρομαγνητικού πεδίου η οποία αλληλεπιδρά με το ηλεκτρομαγνητικό πεδίο των σταθερών μαγνητών του στάτη κάνοντας έτσι τον οπλισμό να περιστραφεί. Ωστόσο, το μέγεθος του σύρματος του πηνίου στον οπλισμό και η ένταση του ρεύματος που το διαρρέει, είναι δύο βασικοί παράγοντες που επηρεάζουν την ένταση του ηλεκτρομαγνητικού πεδίου. Ακόμα, χρησιμοποιούνται δύο καρβουνάκια ή αλλιώς ψήκτρες οι οποίες δημιουργούν μια κινούμενη επαφή με τον συλλέκτη, με σκοπό την ενεργοποίηση του πηνίου. Οι πιο νέες γενιές αυτών των κινητήρων χρησιμοποιούν ειδικά ηλεκτρονικά που ενεργοποιούν και απενεργοποιούν το ρεύμα στο πηνίο αποφεύγοντας έτσι τους σπινθηρισμούς. Η ταχύτητα περιστροφής του άξονα καθώς και η ροπή που παράγεται εξαρτάται από τα ενεργά πεδία στον στάτορα και στον οπλισμό, καθώς και την μέθοδο σύνδεσης που χρησιμοποιείτε. Πιο συγκεκριμένα, η ταχύτητα είναι ανάλογη της τάσης που εφαρμόζεται στον οπλισμό και μπορεί να ελεγχθεί με την χρήση ενός ποτενσιόμετρου. Όσον αφορά την συνδεσμολογία, όταν η πηγή συνδέεται σε σειρά με τον οπλισμό και τα τυλίγματα, ο κινητήρας παράγει μεγάλη ροπή εκκίνησης, ενώ στην παράλληλη σύνδεση της πηγής με τον οπλισμό και τα τυλίγματα ο κινητήρας μπορεί να ελεγχθεί με μεγαλύτερη ακρίβεια ως προς την ταχύτητα, αλλά δεν παρέχει μεγάλη ροπή εκκίνησης. Για την συγκεκριμένη εφαρμογή οι DC κινητήρες με σύνδεση σε σειρά παράγουν τόσο μεγάλη ροπή η οποία μπορεί να βλάψει την κιθάρα, ενώ στην παράλληλη σύνδεση η ροπή είναι τόσο μικρή που δεν αρκεί για να περιστρέψει τα κλειδιά της κιθάρας. Ωστόσο, για την συγκεκριμένη εφαρμογή θα ταίριαζαν οι σύνθετοι κινητήρες οι οποίοι συνδέουν τον οπλισμό και τα τυλίγματα και σε σειρά αλλά και παράλληλα δίνοντας έτσι υψηλή ροπή και υψηλό έλεγχο της ταχύτητας [77].

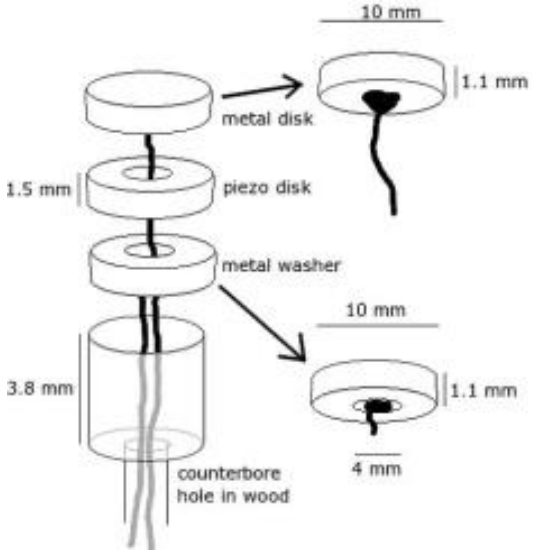
Ένα σημαντικό κομμάτι στις ηλεκτρικές κιθάρες, είναι οι μαγνήτες οι οποίοι έχουν σκοπό την «σύλληψη» της μηχανικής δόνησης των χορδών και την μετατροπή της σε ηλεκτρικό σήμα. Πιο συγκεκριμένα, ο μαγνήτης της κιθάρας αποτελείται από έναν μόνιμο μαγνήτη ο οποίος είναι τυλιγμένος χιλιάδες φορές από ένα χάλκινο σύρμα. Επομένως, όταν η χορδή διεγείρεται, τότε το μαγνητικό πεδίο διαταράσσεται αλλάζοντας την μαγνητική ροή προκαλώντας έτσι επαγωγή στο πηνίο [77].

Η μία κατηγορία μαγνητών είναι οι λεγόμενη μονού πηνίου (Εικόνα 1.56), οι οποίοι είναι πιο επιρρεπής στον θόρυβο που δημιουργείται από την ισχύς των καλωδίων, των μετασχηματιστών και των πηγών εναλλασσόμενου ρεύματος. Ουσιαστικά, οι μαγνήτες μονού πηνίου είναι πιο ευαίσθητοι στα ηλεκτρομαγνητικά πεδία που βρίσκονται κοντά. Όσο πιο ισχυροί οι μαγνήτες, τόσο μεγαλύτερη είναι η μαγνητική ροή με αποτέλεσμα υψηλότερη έξοδο. Οι παθητικοί μαγνήτες μονού πηνίου είναι ευρέως γνωστή, αποτελούνται από ένα χάλκινο σύρμα τυλιγμένο γύρω από έναν μόνιμο μαγνήτη, δεν απαιτούν εξωτερική τροφοδοσία και παράγουν ένα σήμα χαμηλού δυναμικού. Από την άλλοι οι πολυφωνικοί μαγνήτες αποτελούνται από ένα ξεχωριστό μαγνήτη και τύλιγμα για κάθε χορδή παράγοντας έτσι ένα ξεχωριστό σήμα για κάθε χορδή. Ωστόσο, εξαιτίας της μικρής απόστασης των μαγνητών υπάρχει μια αλληλεπίδραση όταν διεγείρεται η διπλανή χορδή από αυτή που αντιστοιχεί στον συγκεκριμένο μαγνήτη [77].



Εικόνα 1.56: Μαγνήτης μονού πηνίου [103].

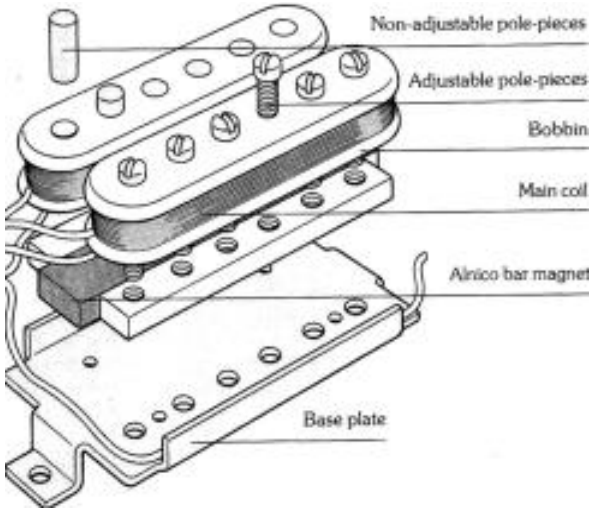
Μια άλλη κατηγορία μαγνητών ονομάζεται πιεζοηλεκτρικοί και κατατάσσονται στην κατηγορία των ενεργών μαγνητών (Εικόνα 1.57). Η κατηγορία αυτή χρειάζεται εξωτερική τροφοδοσία μέσω μπαταριών και ένα κύκλωμα προενίσχυσης, φίλτρων και Equalizer για να λειτουργήσουν. Οι πιεζοηλεκτρικοί μαγνήτες χρησιμοποιούνται κυρίως ηλεκτρό-ακουστικές και ακουστικές κιθάρες καθώς και σε μερικά μπάσα. Η τοποθέτηση των πιεζοηλεκτρικών μαγνητών γίνεται κοντά στον καβαλάρη και το πλεονέκτημά τους είναι ότι δεν επηρεάζονται από άλλο μαγνητικό πεδίο. Αυτό έχει ως αποτέλεσμα, να παράγεται ένας καθαρός και διαυγές ήχος. Επίσης, οι πιεζοηλεκτρική μαγνήτες έχουν απόκριση στο εύρος συχνοτήτων 10-100kHz, ενώ οι απλοί μαγνήτες δίνουν έμφαση σε πιο κεντρικές συχνότητες [77].



Εικόνα 1.57: Πιεζοηλεκτρικός μαγνήτης [104]

Από τις πιο ευρέως γνωστές κατηγορίες μαγνητών είναι οι Humbucker, οι οποίοι αποτελούνται από δύο πηνία (Εικόνα 1.58). Πιο συγκεκριμένα, η περιέλιξη του ενός πηνίου στους μαγνητικούς πόλους γίνεται ανάποδα από το άλλο και αυτό έχει ως αποτέλεσμα οι μαγνήτες να έχουν αντίθετη πολικότητα. Με αυτόν τον τρόπο, το βουητό από το περιβάλλον φτάνει στο κάθε πηνίο και εξαιτίας των αντίστροφων τυλιγμάτων τα ημιτονοειδή σήματα ηλεκτρομαγνητικής παρεμβολής είναι ίσα και αντίθετα με αποτέλεσμα να αλληλοαναιρούνται. Ουσιαστικά, οι μαγνήτες και τα τυλίγματα των πηνίων είναι εκτός φάσης και το σήμα που λαμβάνεται από το ένα και από το άλλο τύλιγμα είναι συμφασικό. Σε περίπτωση που τα πηνία είναι συνδεδεμένα σε σειρά τότε η

τάση του σήματος διπλασιάζεται εξαιτίας της αύξησης της επαγωγής των μαγνητών. Όμως η σύνδεση των πηνίων σε σειρά εξασθενεί τις υψηλότερες συχνότητες δίνοντας έτσι στον ήχο λιγότερα πρίμα. Σε περίπτωση όμως που η σύνδεση των πηνίων είναι παράλληλα, τότε το σήμα από τις χορδές αθροίζεται. Αυτή η συνδεσμολογία συναντάτε σε τζαζ κιθάρες όπου ο ήχος είναι πιο καθαρός και δεν γίνεται εξασθένηση των υψηλότερων συχνοτήτων [77].



Εικόνα 1.58: Μαγνήτης διπλού πηνίου (Humbucker) [105].

Εν συνεχεία, ένας μικροελεγκτής που θα μπορούσε να χρησιμοποιηθεί στην συγκεκριμένη εργασία είναι ο Arduino Due, ο οποίος περιλαμβάνει έναν 32 bit επεξεργαστή της Atmel βασισμένος στον Cortex M3 της ARM και τρέχει στα 84MHz. Ένας άλλος πιθανός επεξεργαστής είναι ο Beaglebone Black ο οποίος αποτελείται από τον επεξεργαστή AM335X της Texas Instruments ο οποίος βασίζεται στον Cortex A8 της ARM και τρέχει στο 1GHz. Ακόμα, αναλύθηκε ο μικροελεγκτής MSP430 της Texas Instruments ο οποίος είναι 16bit και τρέχει στα 25MHz επιτυγχάνοντας χαμηλή κατανάλωση. Ένας άλλος τύπος μικροελεγκτών είναι οι PIC microcontrollers (Periferal Interface Controller) οι οποίοι έχουν Harvard αρχιτεκτονική και κατασκευάζονται από την Microchip Technology. Ο μικροελεγκτής που επιλέχθηκε τελικά για την συγκεκριμένη εργασία είναι ο Atmega 328p ο οποίος χρησιμοποιείται στο Arduino UNO και τρέχει με ταχύτητες από 25MHz έως 32MHz. Επίσης, είναι μικρός σε μέγεθος και μπορεί να τοποθετηθεί πάνω στην πλακέτα [77].

Για την συγκεκριμένη εργασία εξετάζονται κάποιες τεχνικές επεξεργασίας σήματος οι οποίες θα πρέπει να δειγματοληπτούν γρήγορα και με ακρίβεια και να διορθώνει τις συχνότητες που δίνουν οι μαγνήτες. Η μία πιθανή τεχνική είναι ο γρήγορος μετασχηματισμός Fourier (FFT) ο οποίος μετασχηματίζει το σήμα από το πεδίο του χρόνου στο πεδίο της συχνότητας. Πιο συγκεκριμένα, ο FFT χρειάζεται ένα πλαίσιο δειγμάτων το οποίο αποτελεί μία περίοδο του σήματος και ο ήχος είναι ένα σήμα το οποίο επαναλαμβάνεται τακτικά. Από την άλλη ο μετασχηματισμός κύματος (Wavlet Transform) είναι μια τεχνική η οποία χρησιμεύει σε μεταβαλλόμενα κύματα χαμηλής συχνότητας. Ουσιαστικά, η τεχνική αυτή αλλάζει την επέκταση του χρόνου και όχι την μορφή της κυματομορφής. Μια άλλη τεχνική επεξεργασίας σήματος, βασίζεται στον αλγόριθμο Goertzel ο οποίος αξιολογεί μεμονωμένους όρους του διακριτού μετασχηματισμού Fourier (DFT). Η συγκεκριμένη εφαρμογή απαιτεί μεγάλο αριθμό δειγμάτων για την ύπαρξη μεγαλύτερης ακρίβειας πράγμα που κάνει την συγκεκριμένη τεχνική ακατάλληλη για την

εργασία. Μια έξυπνη τεχνική που θα μπορούσε να χρησιμοποιηθεί είναι η προσπέλαση κατωφλιού (Threshold Crossing), η οποία υπολογίζει την περίοδο του σήματος και κατά συνέπεια μπορεί να καθοριστεί και η συχνότητα. Πιο συγκεκριμένα, όταν το σήμα αυξάνεται και ξεπερνά ένα καθορισμένο σημείο, τότε ξεκινά η χρονομέτρηση μέχρι το σήμα να περάσει από το ίδιο σημείο με θετική κλίση. Έτσι υπολογίζεται η περίοδος του σήματος και εν συνεχεία η συχνότητα του σήματος καθορίζοντας έτσι αν η χορδή είναι κουρδισμένη ή όχι. Μια τελευταία τεχνική περιλαμβάνει τον αναλογικό-ψηφιακό μετατροπέα (ADC) ο οποίος μπορεί να βρεθεί ως περιφερειακό σε πολλούς μικροελεγκτές. Πιο συγκεκριμένα, λαμβάνει ένα σήμα συνεχούς χρόνου και το μετατρέπει σε διακριτές ψηφιακές τιμές. Επειδή η συγκεκριμένη διαδικασία περιλαμβάνει κβαντισμό θα υπάρχει πάντα ένα μικρό σφάλμα. Επίσης. η δειγματοληψία θα πρέπει να γίνεται σε πολλές διαφορετικές περιόδους και όχι μία μετατροπή την φορά [77].

Ένα σημαντικό κομμάτι της συγκεκριμένης εργασίας είναι η τροφοδοσία του όλου συστήματος. Η πηγή τροφοδοσία θα πρέπει να είναι φορητή, ελαφριά και να φτάνει τις 50-100 χρήσης ανά φόρτιση. Η μία λύση είναι η χρήση μπαταριών οι οποίες χωρίζονται σε πρωτεύουσες ή αλλιώς μίας χρήσης και σε δευτερεύουσες ή αλλιώς επαναφορτιζόμενες. Οι μπαταρίες γενικός αποτελούνται από δύο πόλους έναν θετικό και έναν αρνητικό οι οποίοι χαρακτηρίζονται ως άνοδος και κάθοδος αντίστοιχα. Οι δευτερεύουσες μπαταρίες που χρησιμοποιούνται συνήθως είναι λιθίου και βγαίνουν σε διάφορα μεγέθη και σχήματα. Οι κυψελίδες των μπαταριών χωρίζονται σε διάφορους τύπους όπως γαλβανικές κυψελίδες, ηλεκτρολυτικές κυψελίδες, κυψελίδες τύπου fuel, κυψελίδες τύπου wet και κυψελίδες τύπου dry. Πιο συγκεκριμένα, οι μπαταρίες με κυψελίδες τύπου dry χρησιμοποιούν έναν ηλεκτρολύτη σε μορφή τζελ και αυτό τους δίνει το πλεονέκτημα ότι μπορούν να τοποθετηθούν με οποιονδήποτε προσανατολισμό πάνω στο σύστημα. Πάραυτα, οι μπαταρίες με κυψελίδες τύπου dry παρέχουν πάρα πολύ μικρή τάση και θα πρέπει να τοποθετηθούν αρκετές από αυτές σε σειρά για να επιτευχθεί η επιθυμητή τάση εξόδου. Οι παράγοντες που παίζουν βασικό ρόλο στην απόδοση των μπαταριών είναι η ποσότητα του φορτίου, η θερμοκρασία και η χωρητικότητα της μπαταρίας. Ουσιαστικά, η χωρητικότητα της μπαταρίας είναι η ποσότητα του ηλεκτρικού φορτίου που μπορεί να αποδώσει στην επονομαζόμενη τάση. Όσα περισσότερα ηλεκτρόδια περιέχονται στην κάθε κυψέλη, τόσο μεγαλύτερη θα είναι και η χωρητικότητα της μπαταρίας. Επιπλέον, οι μονάδες μέτρηση της χωρητικότητας της μπαταρίας είναι τα αμπερώρια (A·h). Από την άλλη, αν οι μπαταρίες δεν χρησιμοποιηθούν για μεγάλο χρονικό διάστημα, τότε δημιουργείτε ένα φαινόμενο αυτό-εκφόρτισης. Επίσης, μετά από πολλές χρήσεις, οι μπαταρίες χρειάζονται αντικατάσταση. Τέλος, οι μπαταρίες αποδίδουν καλύτερα όταν έχουν χαμηλούς ρυθμούς εκφόρτισης [77].

Από την άλλη πλευρά, μια λύση αποτελεί και η χρήση τροφοδοτικού για την τροφοδοσία του συστήματος. Πιο συγκεκριμένα, τα τροφοδοτικά μετατρέπουν την μία μορφή ηλεκτρικής ενέργειας σε μία άλλη με σκοπό την τροφοδοσία και είναι ικανά να διατηρούν την ίδια τάση ή το ίδιο ρεύμα εξόδου ανεξάρτητα από τις αλλαγές στο ρεύμα φορτίου ή της τάσης εισόδου. Επίσης, τα τροφοδοτικά πρέπει να δίνουν την απαραίτητη τάση που χρειάζεται το φορτίο συμπεριλαμβανομένου της καταναλισκόμενης τάσης που χρειάζεται το κάθε τροφοδοτικό. Για την συγκεκριμένη εφαρμογή η χρήση τροφοδοτικού δεν είναι η κατάλληλη, διότι είναι ογκώδη και απαιτούν από τον χρήστη να βρίσκεται κοντά σε κάποια πρίζα [77].

Η διεπαφή μεταξύ χρήστη και συστήματος μπορεί να γίνει με την δημιουργία μίας εφαρμογής η οποία θα βρίσκεται στο smartphone του χρήστη. Ένα ευρέως γνωστό λειτουργικό σύστημα στα Apple smartphone είναι το Apple iOS το οποίο αποτελείται από τέσσερα επίπεδα. Στο επίπεδο Core OS layer υπάρχουν υπηρεσίες χαμηλού (Hardware & network) και υψηλού επιπέδου (Gatekeeper), καθώς επίσης περιλαμβάνεται και το πλαίσιο Core Bluetooth το οποίο επιτρέπει την επικοινωνία μέσω Bluetooth. Εν συνεχεία, το επίπεδο core Service παρέχει υπηρεσίες σε εφαρμογές χωρίς την άμεση παρουσία του χρήστη. Ακόμα, το επίπεδο Media layer περιλαμβάνει τεχνολογίες γραφικών βίντεο και ήχου. Το τέταρτο επίπεδο είναι το Cocoa touch το οποίο περιλαμβάνει πλαίσια τα οποία είναι υπεύθυνα για τις λειτουργίες των πλήκτρων και κάποιων υπηρεσιών υψηλού επιπέδου. Οι εφαρμογές της Apple παρέχονται στους χρήστες μέσω του App Store και διανέμονται μέσω του πλαισίου Gatekeeper του επιπέδου OS core. Η ανάπτυξη εφαρμογών γίνεται στο ολοκληρωμένο περιβάλλον προγραμματισμού (IDE) Apple Xcode το οποίο είναι δωρεάν για τους εγγεγραμμένους Apple προγραμματιστές. Η γλώσσα προγραμματισμού που χρησιμοποιείται στο Apple Xcode είναι η Objective-C. Επίσης, για να μπορέσει ο δημιουργός να ανεβάσει τις εφαρμογές του στο App Store θα πρέπει να εγγραφεί στην Apple και να αποκτήσει πιστοποίηση [77].

Από την άλλη η εφαρμογή θα μπορούσε να δημιουργηθεί σε smartphone με λειτουργικό σύστημα Android. Πιο συγκεκριμένα, το λειτουργικό σύστημα Android βασίζεται στον Linux Kernel που είναι open source και δωρεάν. Η τελευταία έκδοση του Android ονομάζεται Android Lollipop και έχει σχεδιαστεί με διεπαφή χρήστη κάνοντάς την κατάλληλη για τις συσκευές με οθόνη αφής. Οι Android εφαρμογές γράφονται μέσω του Android Software Development Kit σε γλώσσα προγραμματισμού Java. Από την άλλη, μπορεί να χρησιμοποιηθεί το ολοκληρωμένο περιβάλλον προγραμματισμού (IDE) Eclipse το οποίο κάνει χρήση των Android Development Tools (ADT). Επίσης, το Google App Inventor περιλαμβάνει εργαλεία προγραμματισμού για αρχάριους προγραμματιστές. Ωστόσο, το App Invertor για Android σχεδιάστηκε ώστε να βοηθήσει προγραμματιστές όλων των επιπέδων να δημιουργήσουν λειτουργικές εφαρμογές. Οι Android εφαρμογές που δημιουργούνται αποθηκεύονται στο Android Application Package (APK) όπου υπάρχει ένας φάκελος σε μορφή Zip με αρχεία που χρειάζονται για την εγκατάσταση του λογισμικού. Επιπλέον, το λειτουργικό σύστημα Android κάνει χρήση πολλών εξαρτημάτων και αισθητήρων για την υποστήριξη Android χαρακτηριστικών και λειτουργιών (οθόνη αφής , GPS, γυροσκόπιο, επιταχυσνιόμετρο, αισθητήρες πίεσης, αισθητήρες εγγύτητας θερμόμετρα και μονάδες Bluetooth). Η διεπαφή μεταξύ χρήστη και συστήματος για την συγκεκριμένη εργασία δημιουργήθηκε με την χρήση του open source περιβάλλοντος Android OS [77].

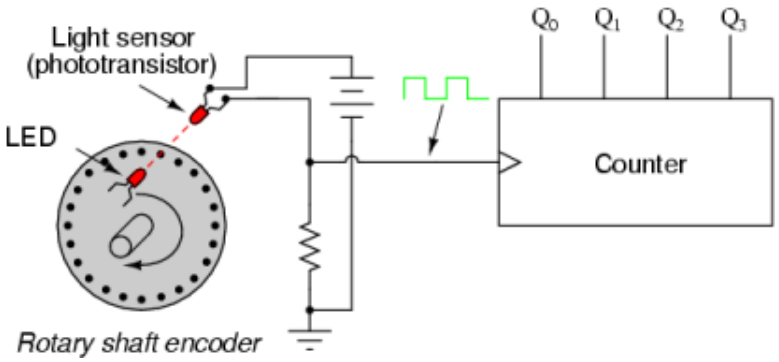
Επιπροσθέτως, στην συγκεκριμένη εργασία χρησιμοποιείτε μονάδα Bluetooth ώστε ο χρήστης να αλληλεπιδρά με το σύστημα. Πιο συγκεκριμένα, το Bluetooth είναι έναν ασύρματο πρωτόκολλο επικοινωνίας το οποίο επιτρέπει την επικοινωνία μεταξύ δύο συσκευών σε μικρή απόσταση και σχεδιάστηκε με σκοπό την αντικατάσταση των καλωδίων μεταξύ των συσκευών. Η τεχνολογία που χρησιμοποιεί το Bluetooth ονομάζεται φάσμα διασκορπισμού με άλματα συχνότητας Frequency-Hopping Spread Spectrum) και είναι χαμηλή σε κόστος και σε κατανάλωση ενέργειας. Για την σωστή λειτουργία του Bluetooth, θα πρέπει και οι δύο συσκευές να έχουν Bluetooth και θα πρέπει να ακολουθηθεί μια διαδικασία που καλείται σύζευξη ώστε οι δύο συσκευές να βρουν η μία την άλλη. Κατά την σύζευξη των δύο συσκευών είναι σημαντικό να βρίσκονται εντός των

καθορισμένων ορίων απόστασης που έχει το Bluetooth. Κατά την ασύρματη σύνδεση μέσω Bluetooth, γίνεται χρήση ενός πρωτοκόλλου το οποίο μεταδίδει τα δεδομένα σε μορφή πακέτων. Επιπλέον, το Bluetooth δημιουργεί ένα δικό του δίκτυο το οποίο ονομάζεται piconet και μπορεί να ελέγχει ταυτόχρονα μέχρι επτά συσκευές όπου η κάθε μία μπορεί να είναι συνδεδεμένη σε πολλά άλλα piconets ταυτόχρονα. Ακόμα η τεχνολογία Bluetooth κάνει χρήση της δομής Master-Slave ώστε να μπορέσει να ελέγξει παραπάνω από μία συσκευές. Ωστόσο, υπάρχουν δύο ειδών υλοποιήσεις για την σχεδίαση του ολοκληρωμένου κυκλώματος του Bluetooth. Η μία υλοποίηση ονομάζεται Single-Mode στην οποία χρησιμοποιείτε μια στοίβα πρωτοκόλλου χαμηλής κατανάλωσης η οποία χαρακτηρίζεται από ένα ελαφρύ Link Layer. Ουσιαστικά με το Link Layer, επιτυγχάνεται η λειτουργία αδρανείας με εξαιρετικά χαμηλή κατανάλωση, απλή εύρεση συσκευής και αξιόπιστη μεταφορά δεδομένων με κρυπτογραφημένες συνδέσεις. Από την άλλη, στην υλοποίηση Dual-Mode οι λειτουργίες του Bluetooth βρίσκονται πάνω στον Bluetooth ελεγκτή ο οποίος περιέχει μια σειρά από παλιά πρωτόκολλα. Ουσιαστικά, όταν δύο συσκευές είναι συζευγμένες, τότε δημιουργείτε ένα κλειδί το οποίο δημοσιεύεται και αποθηκεύεται και από τις δύο συσκευές ώστε να μπορέσει η μία να αυθεντικοποιήσει την άλλη. Με αυτό τον τρόπο τα δεδομένα που ανταλλάσσονται μεταξύ των συσκευών είναι κρυπτογραφημένα ώστε να υπάρχει προστασία από κατασκοπία και πειρατεία. Πριν προλάβουν οι δύο συσκευές να ξεκινήσουν να αλληλεπιδρούν μεταξύ τους θα πρέπει πρώτα να γίνει αυθεντικοποίηση και σύζευξη. Επιπλέον. Με την εισαγωγή της ασφαλής απλής σύζευξης (Secure Simple Pairing) οι μηχανισμοί σύζευξης αλλάζουν. Πιο συγκεκριμένα, χρησιμοποιείται μια μορφή κρυπτογραφίας μέσω του δημοσίου κλειδιού όπου ο χρήστης καλείται να κάνει μια αριθμητική σύγκριση και να επιβεβαιώσει την σύζευξη. Εν συνεχεία, το κατώτερο επίπεδο στην στοίβα πρωτοκόλλου του Bluetooth είναι το επίπεδο ραδιοκύματος το οποίο είναι υπεύθυνο για την διαμόρφωση και την απο-διαμόρφωση των δεδομένων σε σήματα RF. Επιπλέον, το επίπεδο ραδιοκύματος περιλαμβάνει το επίπεδο ευαισθησίας, τα χαρακτηριστικά διαμόρφωσης και την ανοχή των ραδιοσυχνοτήτων. Επίσης, κάτω από το επίπεδο ραδιοκύματος βρίσκεται το επίπεδο της βασικής ζώνης το οποίο είναι υπεύθυνο για την διαμόρφωση των δεδομένων που μεταδίδονται από και προς το επίπεδο ραδιοκύματος. Επίσης, στο ίδιο επίπεδο βρίσκεται ο ελεγκτής ζεύξης ο οποίος είναι υπεύθυνος για την εκτέλεση εντολών αλλά και για την δημιουργία και την διαχείριση των συνδέσεων από τον διαχειριστή συνδέσμων. Το αμέσως επόμενο επίπεδο είναι αυτό της διαχείρισης συνδέσμων το οποίο είναι υπεύθυνο για την δημιουργία και την διαμόρφωση συνδέσμων καθώς και για την διαχείριση αιτημάτων αλλαγής ισχύος. Ωστόσο, υπάρχουν δύο είδη συνδέσμων, η σύγχρονη προσανατολισμένη σύνδεση, η οποία περιλαμβάνει δεσμευμένο εύρος ζώνης καναλιών για την επικοινωνία δύο συσκευών Mastr-Slave και διαχειρίζεται την ανταλλαγή δεδομένων χωρίς την αναμετάδοση πακέτων. Ο άλλος τύπος ονομάζεται ασύγχρονη σύνδεση και προσφέρει μεγαλύτερη ασφάλεια μετάδοσης και σύνδεσης μιας και κωδικοποιεί την πληροφορία σε πακέτα. Το αμέσως επόμενο επίπεδο είναι το HCI (Host Controller Interface) το οποίο είναι υπεύθυνο για τον διαχωρισμό των υψηλότερων επιπέδων από τα χαμηλότερα επίπεδα. Ουσιαστικά, το επίπεδο αυτό υποστηρίζεται σε συστήματα Bluetooth που υλοποιούνται από δύο επεξεργαστές, όπου ο ένας αποσκοπεί στην υλοποίηση κατώτερων επιπέδων, ενώ ο άλλος για την υλοποίηση υψηλότερων επιπέδων. Στην συνέχεια, κάτω από το επίπεδο HCI βρίσκεται το επίπεδο L2CAP 9Logical Link Control and Adaption Protocol) το οποίο είναι υπεύθυνο για την δημιουργία συνδέσεων σε ήδη εγκατεστημένους συνδέσμους ή μη, για την πολυπλεξία μεταξύ των πρωτοκόλλων

υψηλότερων επιπέδων καθώς και για το επαναπακετάρισμα των πακέτων δεδομένων από υψηλότερα επίπεδα ώστε να γίνουν αντιληπτά από χαμηλότερα επίπεδα. Τα αμέσως επόμενα επίπεδα περιλαμβάνουν τα πρωτόκολλα εύρεσης υπηρεσίας SDP (Service Discovery Protocol) και το RFCOMM. Πιο συγκεκριμένα, το SDP καθορίζει ενέργειες τόσο για SDP διακομιστές όσο και για SDP πελάτες. Ενώ το RFCOMM εξομοιώνει τις ρυθμίσεις και την κατάσταση της σειριακής καλωδιακής γραμμής μιας σειριακής θύρας RS-232, καθώς επίσης συνδέεται τόσο με τα χαμηλότερα όσο και με τα υψηλότερα επίπεδα. Το τελικό επίπεδο είναι το επίπεδο πρωτοκόλλου μεταφοράς που ονομάζεται ανταλλαγή αντικειμένων (OBEX Object Exchange) και ορίζει αντικείμενα δεδομένων που ανταλλάσσονται από δύο συσκευές με την χρήση ενός πρωτοκόλλου επικοινωνίας [77].

Επιπροσθέτως, γίνεται μια αναφορά στον τρόπο λειτουργίας των κωδικοποιητών (Encoders) οι οποίοι έχουν σκοπό την μετατροπή ενός κομματιού πληροφορίας σε ένα άλλο πεδίο. Πιο συγκεκριμένα, το δειγματοληπτιμένο σήμα το λαμβάνει ο κωδικοποιητής και το μετατρέπει σε μία ψηφιακή αναπαράσταση η οποία θα είναι απαραίτητη για την ρύθμιση των κλειδιών της κιθάρας. Στην συγκεκριμένη εργασία, ο κωδικοποιητής που χρησιμοποιείται είναι τύπου μορφομετατροπής (Transducer). Ουσιαστικά, οι μορφομετατροποίς αποτελούνται από αισθητήρες κίνησης για την ανίχνευση είτε της θέσης είτε του προσανατολισμού με σκοπό την ανάδραση του συστήματος. Οι μορφομετατροποίς ελέγχουν την θέση είτε γραμμικά είτε περιστροφικά. Στο συγκεκριμένο σύστημα, η περιστροφική θέση των κλειδιών της κιθάρας είναι η μεταβλητή η οποία ελέγχεται διότι περιστροφικά επιτυγχάνεται το κούρδισμά της. Ο περιστροφικός κωδικοποιητής (Εικόνα 1.59), λαμβάνει το σήμα σφάλματος και εξάγει την ποσότητα κατά την οποία θα πρέπει να περιστραφεί ο αντίστοιχος κινητήρας για να κουρδιστεί η χορδή. Βέβαια, η χρήση κωδικοποιητών είναι περιττή μιας και οι κινητήρες δεν περιστρέφονται γρήγορα, καθώς επίσης γίνεται συνέχεια δειγματοληψία του σήματος της χορδής και εντοπίζεται από εκεί το σφάλμα [77].

Μια ακόμα αναφορά γίνεται στις γέφυρες τύπου «Η» οι οποίες χρησιμοποιούνται με DC κινητήρες με σκοπό τον έλεγχο της ταχύτητας και της φοράς περιστροφής. Πιο συγκεκριμένα, οι γέφυρες τύπου «Η» μπορούν να κατασκευαστούν είτε με την χρήση ηλεκτρονόμων (Relay) είτε με την χρήση ημιαγωγών. Για την συγκεκριμένη εργασία, γίνεται χρήση γέφυρας τύπου «Η» με ημιαγωγούς διότι είναι μικρή σε μέγεθος και προσφέρει μεγαλύτερη ταχύτητα [77].



Εικόνα 1.59: Παράδειγμα κωδικοποιητή περιστροφικού άξονα [106].

Εν συνεχεία, εξετάζεται η σχεδίαση του Hardware και του Software του συστήματος. Αρχικά, η αρχιτεκτονική του συστήματος παρουσιάζεται μέσα από μπλοκ διαγράμματα, ξεκινώντας από τον χρήστη ο οποίος αλληλεπιδρά με το σύστημα μέσω της

εφαρμογής που περιέχει το smartphone. Ο μικροελεγκτής είναι συνδεδεμένος μέσω Bluetooth με το smartphone και αφού γίνουν οι επιθυμητές επιλογές, ο μικροελεγκτής αναμένει το αναλογικό σήμα από τους μαγνήτες το οποίο για να παραχθεί θα πρέπει ο χρήστης να διεγείρει τις χορδές. Έπειτα, μέσω των αναλογικό-ψηφιακών μετατροπέων (ADC) γίνεται δειγματοληψία του αναλογικού σήματος και το αποτέλεσμα συγκρίνεται με την επιθυμητή τιμή του σήματος. Η διαφορά που προκύπτει (σφάλμα) ενεργοποιεί τους κινητήρες κατάλληλα με σκοπό την περιστροφή του κλειδιού ώστε να ρυθμιστή η ταλάντωση της χορδής στην επιθυμητή συχνότητα. Επιπλέον, η διανομή ισχύος του συστήματος περιλαμβάνει έναν ρυθμιστή τάσης ώστε η τάση να φτάσει να έχει την κατάλληλη τιμή τόσο για τον μικροελεγκτή όσο και για του κινητήρες συνεχούς ρεύματος [77].

Όσον αφορά την διαδικασία σύνδεσης μέσω Bluetooth, ο χρήστης θα εκκινήσει την εφαρμογή στο smartphone και αν το Bluetooth είναι ενεργοποιημένο θα πρέπει να επιλέξει να γίνει σύζευξη με την μονάδα Bluetooth του συστήματος. Σε περίπτωση όμως που δεν είναι ενεργοποιημένο το Bluetooth, η εφαρμογή ζητά από τον χρήστη να το ενεργοποιήσει και στην συνέχεια θα πρέπει να γίνει σύνδεση με την μονάδα Bluetooth του συστήματος. Επίσης, η εφαρμογή εμφανίζει ένα μήνυμα σε περίπτωση επιτυχημένης ή αποτυχημένης σύνδεσης του Bluetooth. Αφού επιτευχθεί η σύνδεση μέσω Bluetooth του smartphone και του συστήματος, τότε ο χρήστης θα πρέπει να επιλέξει το επιθυμητό κούρδισμα και να πατήσει το κουμπί Start. Μόλις πατηθεί το κουμπί Start, τότε ξεκινά η διαδικασία κουρδίσματος και ο μικροελεγκτής αναμένει το σήμα από την διέγερση των χορδών. Στην συνέχεια, η πραγματική τιμή του σήματος συγκρίνεται με την επιθυμητή τιμή παράγοντας το σήμα σφάλματος, το οποίο με την σειρά του ενεργοποιεί τον αντίστοιχο κινητήρα με σκοπό την περιστροφή του κλειδιού της συγκεκριμένης χορδής. Μόλις τελειώσει το κούρδισμα της πρώτης χορδής, τότε το σύστημα ρωτά τον χρήστη αν υπάρχουν άλλες χορδές για κούρδισμα ώστε να ακολουθηθεί η ίδια διαδικασία. Η σειρά των διαδικασιών στο εσωτερικό του μικροελεγκτή ξεκινούν με την λήψη του αναλογικού σήματος των μαγνητών με σκοπό την μετατροπή του σε ψηφιακό από τον αναλογικό-ψηφιακό μετατροπέα. Στην συνέχεια, τρέχει ο αλγόριθμος επεξεργασίας του ψηφιακού σήματος ώστε να κουρδιστεί η χορδή που διεγέρθηκε περιστρέφοντας το αντίστοιχο κλειδί. Ακόμα, ο μικροελεγκτής ελέγχει την γέφυρα τύπου «Η» ώστε να καθορίσει την ταχύτητα και την φορά περιστροφής του κινητήρα και κατά συνέπεια του κλειδιού της αντίστοιχης χορδής. Πάραυτα, το σύστημα δίνει την δυνατότητα στον χρήστη να κουρδίσει την κιθάρα χειροκίνητα [77].

Εν συνεχεία, δίνονται κάποιες επιλογές για την σχεδίαση του Hardware με βάση τον προϋπολογισμό. Πιο συγκεκριμένα, στην εργασία αυτή γίνεται χρήση των εξαφωνικών μαγνητών από την κιθάρα Power Gig Guitar διότι είναι φθηνότεροι σε σχέση με τους τετραφωνικούς μαγνήτες Ronald GK-3 της εταιρίας Uberta οι οποίοι προσφέρουν ξεχωριστή ανάλυση της κάθε χορδής. Για την επιλογή του κατάλληλου κινητήρα θα πρέπει να ληφθεί υπ’ όψιν η ροπή που πρέπει να ασκηθεί στο κλειδί της χορδής Μι καντίνι διότι σε αυτήν ασκείτε η μεγαλύτερη τάση. Επίσης, σημαντικό ρόλο παίζει και ο όγκος του κινητήρα διότι θα πρέπει να προσαρμοστεί επάνω στην κεφαλή της κιθάρας. Η ροπή που πρέπει να παράγει ο κινητήρας πρέπει να είναι τουλάχιστον από 30-40lbs-in. Επομένως, για την συγκεκριμένη εργασία επιλέχθηκε ο κινητήρας ROB-12472 Standard Gearmotor διότι είναι προσιτός, οικονομικός και ικανοποιεί τις απαιτήσεις τόσο ως προς το μέγεθος όσο και ως προς την ροπή. Το μειονέκτημα του συγκεκριμένου κινητήρα είναι ότι για μία

πλήρης περιστροφή χρειάζεται 10sec με αποτέλεσμα να μην ικανοποιεί τον επιθυμητό χρόνο κουρδίσματος του οργάνου [77].

Στην συνέχεια, γίνεται μια ανάλυση πάνω στις γέφυρες τύπου «Η» οι οποίες ελέγχουν μέσω Software την αμφίδρομη κίνηση και την ακινητοποίηση των DC κινητήρων. Επιπλέον, για την υλοποίηση μιας γέφυρας τύπου «Η» μπορεί να γίνει χρήση ηλεκτρονικών και πιο συγκεκριμένα διπολικών τρανζίστορ (BJT). Πιο συγκεκριμένα, τα διπολικά τρανζίστορ (BJT), μπορούν να κάνουν γρήγορες εναλλαγές παράγοντας λιγότερο θόρυβο και χρειάζονται ένα μικρό ρεύμα στην βάση για να ελέγξουν την διέλευση ρεύματος ή όχι μεταξύ συλλέκτη και εκπομπού. Ένας άλλος τρόπος υλοποίησης μιας γέφυρας τύπου «Η» είναι με την χρήση τρανζίστορ τύπου FET. Ουσιαστικά, τα τρανζίστορ τύπου FET παράγουν λίγο θόρυβο καθώς επίσης περιέχουν μια εσωτερική αντίσταση η οποία καθορίζει το μέγιστο ρεύμα που θα περάσει από το εσωτερικό του τρανζίστορ. Επιπλέον, μπορούν να ελέγχουν ψηφιακά σήματα καθώς και μεγάλα ρεύματα. Ουσιαστικά, τα τρανζίστορ τύπου FET ελέγχουν την διέλευση ρεύματος μεταξύ πηγής και εκπομπού μέσω μιας τάσης στην πύλη-θύρα (Gate) του τρανζίστορ. Από την άλλη πλευρά, η υλοποίηση μιας γέφυρας τύπου «Η» θα μπορούσε να γίνει με την χρήση ηλεκτρονόμων ή αλλιώς ηλεκτρομηχανικών ρελέ τα οποία μπορούν να ελέγξουν αρκετά μεγάλα ρεύματα. Όμως, όσον αφορά την χρονική απόκριση του συστήματος οι ηλεκτρονόμοι είναι αρκετά αργοί και είναι ογκώδης ως προς το μέγεθος. Ένας άλλος τρόπος κατασκευής μιας γέφυρας τύπου «Η» είναι με την χρήση Solid State Relay (SSR) τα οποία δεν περιέχουν μηχανικά μέρη και είναι πιο γρήγορα και αξιόπιστα. Όμως και τα SSR είναι ογκώδης και αρκετά ακριβά για την συγκεκριμένη εφαρμογή. Από την άλλη ο πιο εύκολος και αξιόπιστος τρόπος είναι η χρήση ολοκληρωμένων κυκλωμάτων ως γέφυρες τύπου «Η». Πιο συγκεκριμένα, τα ολοκληρωμένα κυκλώματα L293 DUAL-H bridge της Texas Instruments περιέχουν δύο γέφυρες τύπου «Η» και είναι αρκετά φθηνά. Επίσης, με την χρήση της τεχνικής διαμόρφωσης πλάτους παλμού μπορεί να ελεγχθεί η ταχύτητα περιστροφής καθώς και η φορά περιστροφής του κινητήρα [77].

Ακόμα, γίνεται ανάλυση για την επιλογή κατάλληλου κωδικοποιητή ο οποίος χρησιμοποιείτε από το σύστημα ώστε να αναγνωρίζει την περιστροφή που διέγραψε ο κινητήρας καθώς επίσης να ελέγχει και την ποσότητα στροφών που θα πραγματοποιήσει. Οι κωδικοποιητές με φωτοαντιστάσεις είναι μια πιθανή επιλογή για το συγκεκριμένο σύστημα. Πιο συγκεκριμένα, για να λειτουργήσει ο κωδικοποιητής με τις φωτοαντιστάσεις θα πρέπει η έξοδός του να είναι στραμμένη σε ένα LED. Μεταξύ της εξόδου και του LED τοποθετείται ένας μικρός περιστρεφόμενος δίσκος με δύο τρύπες από τις οποίες επιτρέπει την διέλευση του φωτός. Αυτό έχει ως αποτέλεσμα το δισκάκι να κάνει περιστροφές ενός τετάρτου και να δίνεται σήμα High όταν η φωτοαντίσταση λαμβάνει φως και σήμα Low όταν η φωτοαντίσταση δεν λαμβάνει φως. Παρόλο που αυτός ο τύπος κωδικοποιητή είναι φθηνός έχει μεγάλο χρόνο απόκρισης. Ένας άλλος τύπος είναι οι κωδικοποιητές με φωτοδίοδο οι οποίοι λειτουργούν με την ίδια λογική που λειτουργούν οι κωδικοποιητές με φωτοαντίσταση, μόνο που αντί το φώς να μετατρέπεται σε αντίσταση, μετατρέπεται σε ρεύμα. Από την άλλη υπάρχουν κωδικοποιητές με αισθητήρα αντανάκλασης οι οποίοι περιλαμβάνουν ένα LED υπερύθρων, ένα φωτοτρανζίστορ ώστε να προσμετράτε η ποσότητα αντανάκλασης και έναν περιστροφικό δίσκο ο οποίος αντί για τρύπες έχει διαφορετικά χρώματα. Πιο συγκεκριμένα, ο κωδικοποιητής παράγει στην έξοδό του υψηλό δυναμικό (High) όταν ο αισθητήρας δει σκούρο χρώμα (μαύρο), ενώ δίνει χαμηλό δυναμικό όταν γίνει αντανάκλαση από ανοιχτό χρώμα (άσπρο). Μια άλλη κατηγορία είναι

οι σύγχρονοι κωδικοποιητές οι οποίοι χρησιμοποιούν ένα αγώγιμο κολάρο, έναν μονωτήρα και ένα καλώδιο σήματος. Πιο συγκεκριμένα, το αγώγιμο κολάρο τοποθετείται γύρω από τον άξονα του κλειδιού κουρδίσματος, ενώ ο μονωτήρας εφαρμόζεται στην επιφάνεια του κολάρου με εναλλασσόμενο τρόπο. Κατά την περιστροφή λοιπόν του κολάρου, το καλώδιο σήματος, ακουμπά στον αγώγιμο αγωγό δίνοντας υψηλό δυναμικό (High) ενώ όταν ακουμπά στον μονωτή δίνει χαμηλό δυναμικό (Low). Για την συγκεκριμένη εφαρμογή, ο σύγχρονος κωδικοποιητής απορρίπτεται μια και είναι δύσκολος ως προς την υλοποίηση. Η τελευταία κατηγορία που αναλύεται είναι οι κωδικοποιητές με σωληνωτό διακόπτη ο οποίος έχει παρόμοια λειτουργία με τον αισθητήρα ανάκλασης. Πιο συγκεκριμένα, αντί να διαβάζει μια αντανάκλαση, διαβάζει αν υπάρχει κάποιος μαγνήτης μπροστά στην έξοδο. Τελικά για την συγκεκριμένη εργασία θα γίνει χρήση του κωδικοποιητή με αισθητήρα αντανάκλασης με τύπο QTR-1RC Reflector Sensor της εταιρίας Pololu [77].

Εν συνεχεία, γίνεται ανάλυση της σχεδίασης μικροελεγκτών. Ένα από τα βασικότερα χαρακτηριστικά το οποίο πρέπει να ληφθεί υπ’ όψιν είναι η πηγή τροφοδοσίας του μικροελεγκτή. Επίσης, μια καλή πρακτική είναι η ύπαρξη ενός ενδεικτικού LED το οποίο θα ενεργοποιείται όταν η πλακέτα διαρρέεται από ρεύμα. Όσον αφορά την ακίδα επαναφοράς (reset) του μικροελεγκτή, συνήθως βρίσκεται σε υψηλό δυναμικό και όταν μεταβεί σε χαμηλό δυναμικό τότε γίνεται επαναφορά (reset) του μικροελεγκτή. Επιπλέον, υπάρχει ένας ταλαντωτής στο εσωτερικό του μικροελεγκτή ο οποίος αποσκοπεί στον συγχρονισμό του ρογιού του μικροεπεξεργαστή καθώς και όλων των υπόλοιπων υποσυστημάτων. Συνήθως, οι ταλαντωτές κρυστάλλου είναι πιο ακριβοί σε σχέση με τους ταλαντωτές RC. Για τον έλεγχο DC κινητήρων, γίνεται προσθήκη μιας γέφυρας τύπου «Η». Η σχεδίαση της γέφυρας τύπου «Η» πρέπει να γίνει με ιδιαίτερη προσοχή ώστε να αποφευχθεί ο θόρυβος που θα επηρεάσει τον μικροελεγκτή και οι απότομες αυξήσεις οι οποίες μπορούν να καταστρέψουν τον μικροελεγκτή. Επίσης, η μονάδα Bluetooth που επιλέχθηκε για την συγκεκριμένη εργασία είναι η HC-05 η οποία είναι συμβατή με τον μικροελεγκτή MSP430 και παρέχει την δυνατότητα Master-Slave [77].

Εν συνεχεία, γίνεται μια αναφορά στην σχεδίαση της διάτρητης πλακέτας (PCB) πάνω στην οποία θα τοποθετηθεί το ολοκληρωμένο κύκλωμα (IC) του μικροελεγκτή και των υπόλοιπων ηλεκτρονικών εξαρτημάτων καθώς και οι συνδέσεις μεταξύ τους. Η σχεδίαση της πλακέτας έγινε με την χρήση του προγράμματος Eagle Light Edition το οποίο περιέχει περιορισμούς στο διαθέσιμο χώρο της πλακέτας. Τα αρχεία που δημιουργούνται από το συγκεκριμένο πρόγραμμα περιλαμβάνουν τα σχέδια, τις διαστάσεις των υλικών, τα σχήματα, το BOM, τον φάκελο επιπέδων (layers), τον φάκελο με τις τοποθετήσεις των εξαρτημάτων, την σύνθεση των σχεδίων και των εντολών, καθώς και το Geber file set. Το Gerber file set περιλαμβάνει αρχεία τα οποία είναι χρήσιμα για τον κατασκευαστή όπως το περίγραμμα FAB, καθώς και ειδικά χαρακτηριστικά όπως εγκοπές, «κοψίματα», λοξοτομές κτλ. Ο σχεδιασμός της πλακέτας παίζει σημαντικό ρόλο όσον αφορά την ακρίβεια και την πληρότητα. Πιο συγκεκριμένα, περιέχει σημαντικές πληροφορίες για την λειτουργία της πλακέτας, περιλαμβάνει διάφορες σχεδιαστικές λεπτομέρειες, ενσωματώνει τον αριθμό παρτίδας κατασκευής για τον προσδιορισμό τόσο της τιμής όσο και των προδιαγραφών καθώς επίσης περιέχει και τις προδιαγραφές συσκευασίας. Εξίσου σημαντικό ρόλο παίζει η τοποθέτηση των ηλεκτρονικών εξαρτημάτων διότι με αυτό τον τρόπο καθορίζεται η θερμική διαχείριση, η λειτουργία και ο θόρυβος. Η θερμότητα που παράγεται από το ολοκληρωμένο κύκλωμα (IC) και από τα

υπόλοιπα ηλεκτρονικά εξαρτήματα μεταδίδεται στα στρώματα χαλκού της πλακέτας. Το ιδανικό είναι να βρίσκεται όλη η πλακέτα στην ίδια θερμοκρασία. Για την αντιμετώπιση της θερμοκρασία θα πρέπει να δημιουργηθούν περισσότερα στρώματα γείωσης ή ισχύος που να συνδέονται με πηγές θερμότητας. Ακόμα, με την δημιουργία διαδρόμων θερμότητας και υψηλού ρεύματος βελτιώνεται η αγωγή της θερμότητας. Επιπλέον, με την μεγιστοποίηση της περιοχής που μεταφέρει την θερμότητα στο περιβάλλον, και με την χρήση θερμικά αγώγιμων πεδίων η θερμοκρασία μειώνεται δραματικά. Επίσης, η τοποθέτηση των ηλεκτρονικών εξαρτημάτων πρέπει να γίνει με τέτοιον τρόπο που θα διευκολύνει την σχεδίαση των διαδρομών επάνω στην πλακέτα παρέχοντας την βέλτιστη απόδοση [77].

Επιπροσθέτως, είναι πολύ σημαντική η σειρά με την οποία θα γίνει η τοποθέτηση των εξαρτημάτων επάνω στην πλακέτα. Πιο συγκεκριμένα, πρώτα τοποθετούνται οι επαφές, μετά τα κυκλώματα ισχύος, έπειτα τα ευαίσθητα και ακριβείας κυκλώματα, εν συνεχεία τα κρίσιμα κυκλώματα και τέλος όλα τα υπόλοιπα. Όσον αφορά την σχεδίαση της διαδρομής, επιλέγεται με βάση τα επίπεδα ισχύος, την ευαισθησία θορύβου και την ικανότητα δημιουργίας της διαδρομής. Τα γρήγορα μεταβαλλόμενα σήματα ή αλλιώς τα σήματα υψηλής συχνότητας πρέπει να εξετάζονται με μεγάλη προσοχή όταν τοποθετούνται κόμβοι υψηλής αντίστασης. Επίσης η τελική διάταξη πρέπει πάντα να αναθεωρείται τόσο ως προς την διαδρομές όσο και ως προς την τοποθέτηση των εξαρτημάτων ώστε να ικανοποιούνται όλοι οι περιορισμοί σχεδιασμού. Ουσιαστικά, γίνεται επαλήθευση ότι η ευαίσθητοι κόμβοι και τα κυκλώματα είναι προστατευμένα από τις πηγές θορύβου, ότι υπάρχει μάσκα συγκόλλησης μεταξύ των pin και των αγωγών καθώς και ότι το Silkscreen είναι καθαρό. Επιπλέον, η προσθήκη των επιπέδων γίνεται ανά ζεύγη μιας και η προσθήκη χαλκού γίνεται ανά ζεύγη. Πιο συγκεκριμένα, κατά την στοίβαξη των επιπέδων θα πρέπει πάντα το πρώτο επίπεδο κάτω από τα εξαρτήματα να είναι αυτό της γείωσης ενώ όλα τα υπόλοιπα να είναι τα στρώματα ισχύος. Ωστόσο, υπάρχει και ο έλεγχος των κανονισμών σχεδιασμού (DRC) το οποίο περιλαμβάνεται στο περιβάλλον σχεδίασης του ηλεκτρονικού σχεδίου και καθορίζει αν το φυσικό επίπεδο σε μία διάταξη ενός ολοκληρωμένου κυκλώματος ικανοποιεί τις απαιτούμενες παραμέτρους. Ωστόσο, ο έλεγχος των κανονισμών σχεδιασμού αντιλαμβάνεται μόνο τα σφάλματα για τα οποία έχει προγραμματιστεί να παρακολουθεί, ενώ το σετ κανόνων συχνά αλλάζει ανάλογα με τα μεμονωμένα σχέδια. Πιο συγκεκριμένα, με τον έλεγχο αυτόν καλύπτονται τα κενά μεταξύ των συσκευών, τα ασύνδετα δίκτυα, τα βραχυκυκλώματα και οι παραβιάσεις κενού αέρα. Επίσης, ελέγχεται αν οι αγωγοί είναι πολύ κοντά μεταξύ τους ή κοντά σε συγκολλήσεις. Σημαντικό ρόλο πίσω από τον σχεδιασμό μιας πλακέτας παίζει το κόστος. Στην συγκεκριμένη εργασία, οι τελική πλακέτα σχεδιάστηκε μέσω του EAGLE Cad και κατασκευάστηκε από την OSH Park [77].

Το τελικό κομμάτι Hardware που χρησιμοποιήθηκε στην συγκεκριμένη εργασία είναι η συσκευή smartphone. Πιο συγκεκριμένα χρησιμοποιήθηκε ένα Samsung Galaxy S4 το οποίο έχει Bluetooth για την ασύρματη σύνδεση με τον μικροελεγκτή καθώς και λειτουργικό σύστημα Android που χρειάζεται η εφαρμογή [77].

Εν συνεχεία, σημαντικό ρόλο παίζει το λογισμικό ως προς το γρήγορο, ακριβή και αποτελεσματικό κούρδισμα του οργάνου. Πιο συγκεκριμένα, η εφαρμογή που έχει εγκατασταθεί στο Smartphone επικοινωνεί μέσω του Bluetooth με τον μικροελεγκτή δίνοντας την δυνατότητα στον χρήστη να επιλέξει ένα από τα πέντε προκαθορισμένα κουρδίσματα. Έπειτα, θα πρέπει να πατήσει το κουμπί Start ώστε να ξεκινήσει η

διαδικασία κουρδίσματος, όπου ο μικροελεγκτής περιμένει την διέγερση της πρώτης χορδής. Μετά το κούρδισμα της πρώτης χορδής το σύστημα προτρέπει τον χρήστη να διεγείρει την επόμενη χορδή. Μόλις κουρδιστούν όλες οι χορδές, τότε η οθόνη του Smartphone επιστρέφει στο αρχικό μενού όπου εμφανίζονται τα πέντε προκαθορισμένα κουρδίσματα και ένα κουμπί εξόδου ώστε ο χρήστης να μπορεί να βγει και να κλείσει την εφαρμογή. Όσον αφορά τον αλγόριθμο επεξεργασίας του σήματος, θα πρέπει να αναλύει το σήμα γρήγορα και με ακρίβεια. Ουσιαστικά, η ταχύτητα και η ακρίβεια της ανάλυσης του σήματος εξαρτάται από την τεχνική που εφαρμόζεται, η οποία στην προκειμένη περίπτωση είναι ο Fast Fourier Transform (FFT). Η τεχνική ανάλυσης πρέπει να εφαρμοστεί στα δείγματα των δεδομένων όσο πιο γρήγορα γίνεται ώστε ο μικροελεγκτής να δώσει την ανάλογη εντολή για το πόσο θα περιστραφεί ο αντίστοιχος κινητήρας, ενώ ταυτόχρονα ο αλγόριθμος θα πρέπει να ανανεώνει την πληροφορία της συχνότητας δίνοντας έτσι την απαραίτητη ανατροφοδότηση στο σύστημα. Η διαδικασία αυτή πρέπει να επαναλαμβάνεται για όλες τις χορδές. Η κάθε χορδή έχει μία προκαθορισμένη συχνότητα ανάλογα με το κούρδισμα που έχει επιλεχθεί από τον χρήστη. Για την εισαγωγή των συχνοτήτων αυτών στον μικροελεγκτή, το μπάσο κουρδίστηκε χειροκίνητα και οι κατάλληλες συχνότητες καταγράφηκα στην μνήμη του μικροελεγκτή. Επίσης, η ποιότητα της ανάλυσης εξαρτάται κυρίως από την ποσότητα και την ποιότητα των μαγνητών [77].

Όσον αφορά την σύνδεση μεταξύ του Hardware και του Software, θα πρέπει ο κώδικας της Android εφαρμογής να λαμβάνει τις εισόδους του χρήστη και μέσω του πρωτοκόλλου επικοινωνίας να στέλνονται στον μικροελεγκτή. Επίσης, η εφαρμογή θα πρέπει να παρέχει την δυνατότητα στον χρήστη να αναζητά και να συνδέεται με άλλες ενεργές συσκευές Bluetooth. Αυτό επιτυγχάνεται μέσω του μενού «Ρυθμίσεις Συσκευής» σε όλες τις συσκευές Android όπου ο χρήστης μπορεί να αναζητά και να συνδέεται σε απομακρυσμένες συσκευές βλέποντας ονόματα και MAC διευθύνσεις. Επιπλέον, η Android εφαρμογή γράφεται σε γλώσσα προγραμματισμού JAVA και η λογική ελέγχου που ακολουθεί είναι η ερμηνεία της εισόδου του χρήστη, η υποστήριξη της γραφικής διεπαφής, η διαχείριση Bluetooth, η μετάδοση δεδομένων και ο έλεγχος ολοκλήρωσης της διαδικασίας. Ακόμα, ελέγχει αν το Bluetooth είναι ενεργοποιημένο, αιτώντας την άδεια από τον χρήστη σε περίπτωση που είναι απενεργοποιημένο και επιτρέπει στον χρήστη την έξοδο από την εφαρμογή τερματίζοντας την διεργασία. Υψίστης σημασίας είναι η συνεχόμενη σύνδεση του μικροελεγκτή με το Smartphone μέσω του Bluetooth ώστε να παρακολουθούνται οι είσοδοι του χρήστη και τα δεδομένα να στέλνονται στον μικροελεγκτή όταν πατηθεί το κουμπί Start. Επιπροσθέτως, ο μικροελεγκτή λαμβάνει ως είσοδο το αναλογικό σήμα από τους μαγνήτες και το μετατρέπει σε ψηφιακό μέσω του αναλογικό-ψηφιακού μετατροπέα. Στην συνέχεια το συγκρίνει με μία επιθυμητή συχνότητα κουρδίσματος και το σήμα σφάλματος οδηγεί κατάλληλα τον αντίστοιχο κινητήρα ώστε να ρυθμιστή η τάση ταλάντωσης της χορδής. Για την εδραίωση της επικοινωνίας μεταξύ της μονάδας Bluetooth και του μικροελεγκτή, θα πρέπει να γίνει διαμόρφωση της μονάδας με την χρήση της ασύγχρονης σειριακής διεπαφής [77].

Στην συγκεκριμένη εφαρμογή είναι επίσης σημαντικός ο σχεδιασμός της τροφοδοσίας διότι η πηγή θα πρέπει να διαρκεί για μεγάλο χρονικό διάστημα και να παρέχει την απαιτούμενη ποσότητα ενέργειας που χρειάζεται το σύστημα. Ουσιαστικά, με την σχεδίαση μίας πηγής ισχύος για όλα τα εξαρτήματα επιτυγχάνεται η εξοικονόμηση χρημάτων, ο απλός σχεδιασμό και δεν προστίθεται επιπλέον βάρος στο όργανο. Πάραυτα, θα πρέπει να γίνει χρήση κατάλληλων ρυθμιστών τάσης ώστε να ρίξουν την τάση στην

απαιτούμενη τιμή που χρειάζεται το κάθε υποσύστημα. Στην συγκεκριμένη εφαρμογή χρησιμοποιήθηκε μια μπαταριοθήκη 6xAA ως πηγή τροφοδοσίας του συστήματος διότι ικανοποιούνται οι απαιτήσεις, το κόστος και η φορητότητα του συστήματος [77].

Εν συνεχεία, η διαδικασία της δημιουργίας του πρωτότυπου συστήματος ξεκινάει με την απόκτηση των εξαρτημάτων, έπειτα με την επαλήθευση της λειτουργίας τους και μετά με την αντιμετώπιση των προβλημάτων που έχουν προκύψει στο τελικό σύστημα. Όσον αφορά του κινητήρες, αρχικά, αγοράστηκε ένας ώστε να εξεταστεί αν μπορεί να περιστρέψει το κλειδί της πιο παχιάς χορδής του οργάνου. Μετά τους κινητήρες αγοράστηκε μία γέφυρα τύπου «Η» η οποία εξετάστηκε αν είναι συμβατή με δύο κινητήρες. Έπειτα, για την κατασκευή του κωδικοποιητή εξετάζεται ένα υλικό το οποίο είναι ιδιαίτερα αντανακλαστικό και ένα λευκό ώστε να ελεγχθεί σε πιο θα δώσει ο αισθητήρας χαμηλό δυναμικό (LOW). Στην συνέχεια, χρωματίζονται με μαύρο χρώμα κάποιες περιοχές και ελέγχεται αν θα δώσει ο αισθητήρας υψηλό δυναμικό (HIGH). Επιπλέον, η τάση που πρέπει να παρέχεται στην μονάδα Bluetooth και στον μικροελεγκτή είναι 3.3Volt, ενώ ο κάθε κινητήρας χρειάζεται από 3Volt-9Volt για να λειτουργήσει. Για να μπορέσει το σύστημα να έχει μικρή κατανάλωση, θα πρέπει οι κινητήρες να λειτουργούν με την ελάχιστη τάση η οποία θα είναι αρκετή για να περιστραφούν τα κλειδιά. Όμως, επειδή οι κινητήρες χρειάζονται διαφορετική τάση για να λειτουργήσουν σε σχέση με τον μικροελεγκτή και την μονάδα Bluetooth θα πρέπει να γίνει χρήση ρυθμιστών τάσης. Αφού τροφοδοτηθούν όλα τα εξαρτήματα από μία πηγή τότε ξεκινά η τοποθέτησή τους πάνω στο σύστημα. Ωστόσο, μια καλή πρακτική είναι η δοκιμή όλου του συστήματος στο Breadboard πριν την τοποθέτησή του πάνω στην πλακέτα. Επιπροσθέτως, κατά την δημιουργία της πλακέτας θα πρέπει τα pin τροφοδοσίας να συνδεθούν τελευταία ώστε να αποφευχθεί οποιοδήποτε βραχυκύκλωμα. Επίσης, η δημιουργία της εφαρμογής Android έγινε με την χρήση του Eclipse IDE από το Android SDK toolkit. Κατά την διάρκεια της κατασκευής του κώδικα, θα πρέπει το Smartphone να είναι συνδεδεμένο μέσω USB με τον υπολογιστή ώστε να γίνεται η μεταφορά των αρχείων, η αποσφαλμάτωση του κώδικα και ο έλεγχος της γραφικής διεπαφής του χρήστη [77].

Όσον αφορά την κατασκευή των PCB, έγινε έρευνα πριν επιλεχθεί κάποια εταιρία. Πιο συγκεκριμένα, μία από αυτές τις εταιρίες είναι η OSH Park η οποία επιτρέπει τα σχέδια που έχουν δημιουργηθεί μέσω του Eagle CAD, αλλά έχει συγκεκριμένες απαιτήσεις σχεδιασμού για την κατασκευή των PCB. Από την άλλη, η εταιρία 4PCB κάνει ειδικές προσφορές σε φοιτητές, δεν απαιτεί ελάχιστη ποσότητα PCB για να γίνει η παραγγελία και παρέχει το Advanced Circuits λογισμικό για την σχεδίαση των PCB. Επιπλέον, η 4PCB ελέγχει τα PCB αρχεία και στέλνει ενημέρωση μέσω e-mail για τυχόν προβλήματα σχεδιασμού. Με αυτό τον τρόπο διασφαλίζονται οι απαιτούμενες προδιαγραφές της κατασκευής. Πάραυτα, υπάρχουν κι άλλοι κατασκευαστές πλακετών οι οποίοι είτε παρέχουν δικό τους λογισμικό σχεδίασης PCB είτε δέχονται και άλλα προγράμματα σχεδίασης όπως το Eagle CAD [77].

Εν συνεχεία, γίνεται ανάλυση του κώδικα της Android εφαρμογής και του μικροελεγκτή. Πιο συγκεκριμένα, η Android εφαρμογή δημιουργήθηκε μέσω του Android SDK το οποίο παρέχει API βιβλιοθήκες καθώς και εργαλεία για το στήσιμο, την αποσφαλμάτωση και τον έλεγχο της εφαρμογής. Επίσης, η εφαρμογή γράφτηκε στο ολοκληρωμένο πρόγραμμα ανάπτυξης Eclipse το οποίο χρησιμοποιεί Android Development Tools Plugin με την χρήση της γλώσσα προγραμματισμού JAVA. Επιπλέον, είναι σημαντικό το τηλέφωνο να είναι συνδεδεμένο κατά την συγγραφή του κώδικα για το

Bluetooth ώστε να είναι φανερό πως αλληλεπιδρά ο κώδικας με την συσκευή. Επιπλέον, μέσω του Android Manifest Permissions ενεργοποιούνται και αποθηκεύονται οι άδειες χρήστη. Ακόμα, η εφαρμογή πρέπει να εμφανίζει μηνύματα σχετικά με το αν έχει ενεργοποιηθεί το Bluetooth και στην περίπτωση που δεν έχει ενεργοποιηθεί θα πρέπει να ζητά άδεια από τον χρήστη για να το ενεργοποιήσει. Ωστόσο, δίνεται η δυνατότητα στον χρήστη να ακυρώσει την ενεργοποίηση του Bluetooth. Επίσης, μόλις γίνει η ενεργοποίηση του Bluetooth, για να μπορέσει να κάνει αναζήτηση θα πρέπει να δημιουργηθεί μία κλάση δέκτη Broadcast και μία κλάση φίλτρου. Μόλις ληφθεί ένα σήμα από μία νέα συσκευή, τότε αυτή αποθηκεύεται και ξεκινά ο κώδικας για το αυτόματο κούρδισμα του οργάνου. Επίσης, θα πρέπει η συσκευή να ανακαλεί προηγούμενες συζεύξεις χωρίς να πρέπει να δοθεί άδεια εκ νέου. Ουσιαστικά, η κινητή συσκευή αποτελεί τον πελάτη (Client), ενώ η μονάδα Bluetooth του συστήματος αποτελεί τον διακομιστή (Server) [77].

Κατά την κατασκευή και την δοκιμή του πρωτότυπου συστήματος είναι σημαντική η ασφάλεια για να μην υπάρξουν κίνδυνοι και τραυματισμοί. Πιο συγκεκριμένα, τα ηλεκτρικά εξαρτήματα θα πρέπει να αποφορτιστούν από τον στατικό ηλεκτρισμό ώστε να μην προκληθεί ζημία σε κάποιο άλλο εξάρτημα. Επίσης, θα πρέπει να αποφευχθούν τα βραχυκυκλώματα σιγουρεύοντας ότι όλες οι γειώσεις καταλήγουν στο ίδιο σημείο. Ωστόσο κατά την συναρμολόγηση και κατά την αποσυναρμολόγηση του συστήματος θα πρέπει να γίνει πρώτα αποσύνδεση της τροφοδοσίας ώστε να αποφευχθεί η πιθανότητα ηλεκτροπληξίας. Ένα εξίσου σημαντικό κομμάτι ασφαλείας είναι τα επίπεδα ακτινοβολίας των συσκευών Bluetooth τα οποία εξαρτώνται από την κλάση του Bluetooth. Ουσιαστικά, η κλάση 1 είναι η πιο ισχυρή διότι αγγίζει σε εύρος τα 100m και την ισχύς μετάδοσης στα 100mW. Από την άλλη η κλάση 2 είναι λιγότερο ισχυρή φτάνοντας σε εύρος τα 10m και ισχύς μετάδοσης στα 2.5mW. Ενώ, η κλάση 3 είναι η λιγότερο ισχυρή απ' όλες φτάνοντας σε εύρος μικρότερο από τα 10m και ισχύς μετάδοσης 1mW. Γι' αυτό τον λόγο οι κλάσης 2 & 3 είναι καταλληλότερες από την κλάση No1 διότι έτσι αποφεύγονται οποιαδήποτε συμπτώματα ηλεκτρικής ευαισθησίας [77].

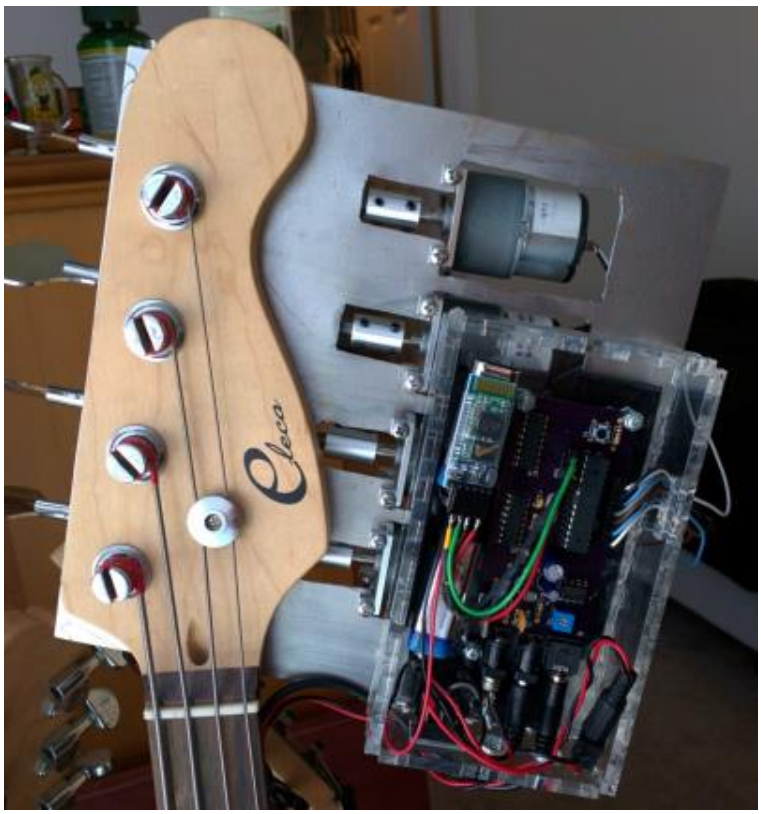
Εν συνεχεία, θα πρέπει να γίνουν δοκιμές στο σύστημα για να διασφαλιστεί ότι δουλεύει ορθά και χωρίς σφάλματα. Μία καλή πρακτική είναι το σύστημα να δοκιμαστεί πρώτα σε επιμέρους υποσυστήματα ώστε να εντοπιστούν πιο εύκολα τα πιθανά σφάλματα. Το ένα υποσύστημα που δοκιμάστηκε είναι οι κινητήρες ώστε να διασφαλιστεί ότι μπορούν να περιστρέψουν τα κλειδιά και ως προς τις δύο κατευθύνσεις. Ένα άλλο υποσύστημα που πρέπει να εξεταστεί είναι η πηγή τροφοδοσίας, τόσο ως προς τις απαραίτητες τάσης όσο και ως προς το ρεύμα στον μικροελεγκτή και τους κινητήρες. Ωστόσο, η πηγή τροφοδοσίας θα πρέπει να δοκιμαστεί με τους κινητήρες υπό φορτίο ώστε να ελεγχθεί ότι παρέχεται η απαραίτητη ποσότητα ρεύματος. Στην συνέχεια, γίνεται δοκιμή του μικροελεγκτή ώστε να διασφαλιστεί ότι γίνεται σωστή εκτέλεση του προγράμματος, ότι επικοινωνεί με όλα τα υποσυστήματα αλλά και με την μονάδα Bluetooth από την οποία λαμβάνει τα δεδομένα που στέλνονται από το Smartphone του χρήστη. Επιπλέον, θα πρέπει να δοκιμαστεί και η μονάδα Bluetooth ώστε να ελεγχθεί ότι επικοινωνεί σωστά τόσο με τον μικροελεγκτή όσο και με την συσκευή smartphone. Ακόμα, η εφαρμογή στο Smartphone είναι μία ακόμη παράμετρος που πρέπει να εξεταστεί για τυχόν σφάλματα. Θα πρέπει επίσης η εφαρμογή να δοκιμαστεί όταν το Smartphone έχει συζευχθεί με την μονάδα Bluetooth του συστήματος ώστε να διασφαλιστεί ότι ο μικροελεγκτής λαμβάνει τις εντολές. Επιπροσθέτως, θα πρέπει να δοκιμαστούν και οι μαγνήτες ώστε να διασφαλιστεί ότι έχουν συνδεθεί σωστά και ότι το σήμα που παράγεται

είναι χωρίς θόρυβο. Σε περίπτωση που το σήμα καταλαμβάνεται από υπερβολικό θόρυβο, θα πρέπει οι μαγνήτες ξανασυνδεθούν και να επανακολληθούν. Αφού ελεγχθούν όλα τα υποσυστήματα τότε ξεκινά ο έλεγχος ολόκληρου του συστήματος. Πιο συγκεκριμένα, αφού ενεργοποιηθεί το σύστημα θα πρέπει να γίνει σύζευξη του Smartphone με την μονάδα Bluetooth του συστήματος και να επιλεχθεί το επιθυμητό κούρδισμα μέσω της εφαρμογής. Έπειτα, με την διέγερση των χορδών το σύστημα θα πρέπει να κουρδίσει τις χορδές στις επιθυμητές συχνότητες. Η επαλήθευση του κουρδίσματος γίνεται με την χρήση ενός εξωτερικού κουρδιστηριού. Η ίδια διαδικασία επαναλαμβάνεται και το σύστημα ελέγχεται ξανά και για τα υπόλοιπα κουρδίσματα. Ουσιαστικά, πρέπει να ελεγχθεί ότι η συσκευή ενεργοποιήθηκε σωστά, ότι τα υποσυστήματα συνδέθηκαν και αλληλεπιδρούν σωστά, ότι η εφαρμογή του Smartphone λειτουργεί κανονικά και το βασικότερο ότι οι χορδές έχουν κουρδιστεί στην επιθυμητή συχνότητα. Επίσης, θα πρέπει να ελεγχθεί και η πλακέτα PCB, ώστε να εξασφαλιστεί ότι δεν υπάρχει θόρυβος ή παρεμβολές μεταξύ των εξαρτημάτων αλλά ούτε και βραχυκυκλώματα. Ωστόσο, μετά την τοποθέτηση των εξαρτημάτων επάνω στην πλακέτα, θα πρέπει να διασφαλιστεί ότι δεν υπάρχουν βραχυκυκλώματα, ότι τα εξαρτήματα βρίσκονται στην σωστή θέση με την κατάλληλη τιμή και ότι ο μικροελεγκτής έχει τοποθετηθεί με τον κατάλληλο προσανατολισμό [77].

Έπειτα, αναλύονται κάποιες μικρές τροποποιήσεις καθώς και το πως αυτές επηρεάζουν το σύστημα. Πιο συγκεκριμένα, η τοποθέτηση ενός πολυφωνικού μαγνήτη κάτω από την κάθε χορδή έχει ως αποτέλεσμα την αλλοίωση του ήχου εξαιτίας της αφαίρεσης μερικών κομματιών ξύλου. Μια άλλη τροποποίηση, αφορά του κωδικοποιητές οι οποίοι δεν είναι απαραίτητη μιας και οι κινητήρες περιστρέφονται αρκετά αργά και θα σταματήσουν στην επιθυμητή συχνότητα. Επιπλέον, σχεδιάστηκε μία βάση από λαμαρίνα για την τοποθέτηση των κινητήρων στην κεφαλή της κιθάρας. Ακόμα στο κάτω μέρος των κλειδιών έγινε μια εγκοπή ώστε ο κάθε κινητήρας να μπορέσει να περιστρέψει το αντίστοιχο κλειδί με την χρήση μιας μύτης με ίσια κεφαλή (Εικόνα 1.60). Μια άλλη σημαντική τροποποίηση που έγινε αφορά την μέθοδο επεξεργασίας του σήματος. Πιο συγκεκριμένα, επειδή έγινε χρήση του μικροελεγκτή ATmega328p, δεν χρησιμοποιήθηκε ο γρήγορος μετασχηματισμός Fourier (FFT), αλλά μια μέθοδος προσπέλασης κατωφλιού με ακρίβεια 0.1Hz. Μια άλλη τροποποίηση που έγινε αφορά την Android εφαρμογή, η οποία αντί να εμφανίζει πολλαπλές οθόνες εμφανίζει μία ώστε να είναι πιο φιλική προς τον χρήστη. Επίσης, η MAC διεύθυνση της μονάδας Bluetooth κωδικοποιήθηκε εντός του κώδικα της εφαρμογής ώστε να γίνεται σύζευξη μόνο με αυτή την συσκευή [77].

Εικόνα 1.60: Η βάση από λαμαρίνα με τους κινητήρες συνδεδεμένους πάνω στα κλειδιά του μπάσου [107].

Ο τελικός σχεδιασμός της οθόνης περιλαμβάνει μία αρχική οθόνη με πέντε κουμπιά όπου το κάθε ένα αντιστοιχεί σε ένα κούρδισμα. Όταν πατηθεί κάποιο από αυτά, τότε στο κάτω μέρος της οθόνης εμφανίζεται ένα μήνυμα το οποίο υποδεικνύει το κούρδισμα που έχει επιλεχθεί και προτρέπει τον χρήστη να διεγείρει την πρώτη χορδή. Τέλος, η πλακέτα (PCB) αντί να τοποθετηθεί στο σώμα της κιθάρας τοποθετήθηκε στην ίδια βάση με τους κινητήρες ώστε να αποφευχθούν τα πολλά καλώδια. Ωστόσο, δημιουργήθηκε μια θήκη για την πλακέτα (PCB) ώστε να παρέχεται μεγαλύτερη προστασία (Εικόνα 1.61) [77].



Εικόνα 1.61: Το τελικό σχέδιο του αυτόματου κουρδιστηριού κιθάρας (Automatic Guitar Tuner) [108]

1.2.7 Final Project ECE 470 Microcontroller Based Guitar Tuner

Σε αυτή την διπλωματική εργασία σχεδιάζεται και υλοποιείται ένα κουρδιστήρι ακριβείας για το κούρδισμα των χορδών μιας κιθάρας. Ουσιαστικά, για την υλοποίηση του κουρδιστηριού έγινε σύγκριση του χρόνου εισόδου και εξόδου, χρησιμοποιήθηκε η μέθοδος των διακοπών (Interrupts), LED, αναλογικό-ψηφιακός μετατροπέας και μία LCD οθόνη [78].

Αρχικά, λαμβάνεται το σήμα από τους μαγνήτες το οποίο παράγεται όταν διεγερθούν οι χορδές και στην συνέχεια ενισχύεται. Έπειτα, κατά την επεξεργασία του σήματος υπολογίζεται η απόσταση μεταξύ των κορυφών του σήματος και γι αυτό θα πρέπει να γίνει χρήση της μη αναστρέψιμης θύρας ενός ενισχυτή. Με αυτόν τον τρόπο επιτυγχάνεται η ενίσχυση μόνο του θετικού μισού του σήματος. Επίσης, για να μείνει μόνο η θεμελιώδης συχνότητα στο σήμα θα πρέπει το πλάτος των υψηλότερων συχνοτήτων να μειωθεί. Αυτό επιτυγχάνεται με την χρήση ενός βαθυπερατού φίλτρου διότι η μέγιστη συχνότητα που μπορεί να φτάσει η κιθάρα με την διέγερση των ανοιχτών χορδών είναι 330Hz. Η πρωταρχική ιδέα ήταν η χρήση ενός αναλογικού-ψηφιακού μετατροπέα ώστε να γίνει δειγματοληψία της κυματομορφής και με την χρήση ενός αλγόριθμου να βρεθεί η τιμή της κορυφής. Αν η τιμή που θα προέκυπτε από τον αλγόριθμο ήταν μεγαλύτερη από την ακατέργαστη κυματομορφή, τότε θα εμφανιζόταν λογικό 1 σε κάποια γενική θύρα εισόδου εξόδου. Με αυτόν τον τρόπο θα κατασκευαζόταν ένας τετραγωνικός παλμός με περίοδο ίδια με αυτή της θεμελιώδης συχνότητας του σήματος. Όμως αντί γι' αυτήν την ιδέα έγινε χρήση ενός ανιχνευτή κορυφής (peak detector) και ενός συγκριτή ώστε να απλοποιηθεί ο κώδικας. Η κυματομορφή που προκύπτει εισέρχεται στον χρονοδιακόπτη ο οποίος είτε με την χρήση συσσώρευσης παλμών είτε συγκρίνοντας την κυματομορφή με την είσοδο θα εντοπίσει την συχνότητα. Ωστόσο, για την αύξηση της ακρίβειας γίνονται αρκετές μετρήσεις και υπολογίζεται ο μέσος όρος [78].

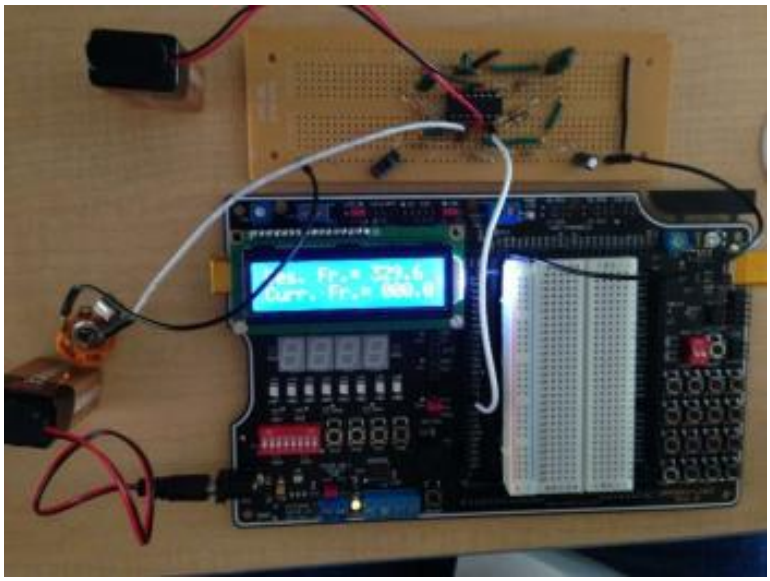
Τον ρόλο της διεπαφής μεταξύ χρήστη και συστήματος παίζει μία οθόνη LCD, η οποία μετά τον εντοπισμό της πραγματικής συχνότητας και αφού συγκριθεί με την επιθυμητή συχνότητα της αντίστοιχης χορδής εμφανίζει την νότα που κουρδίζεται. Ωστόσο, γίνεται χρήση του κώδικα ASM ώστε να επιτευχθεί η μετατροπή των δεκαδικών αριθμών από το δυαδικό σύστημα σε κώδικα ASCII. Με αυτό τον τρόπο οι δεκαδικοί αριθμοί μπορούν να εμφανιστούν στην LCD οθόνη [78].

Για την διεκπεραίωση του πειράματος έγινε χρήση του code warrior για την πλακέτα HCS12 και ένας παλμογράφος με τον οποίο επαληθεύτηκαν τα σήματα που λαμβάνοντας από την κιθάρα αλλά και για τον έλεγχο της σωστής λειτουργίας του κυκλώματος. Για την δοκιμή του κυκλώματος έγινε χρήση του τελεστικού ενισχυτή LM324, μιας γεννήτριας συχνοτήτων και μίας κιθάρας από την οποία το σήμα λήφθηκε μέσω ενός καλωδίου Jack. Για την ανίχνευση της συχνότητας έγινε χρήση της μεθόδου των διακοπών (Interrupts). Ωστόσο, η υλοποίηση της συγκεκριμένης εργασίας είχε επιτυχία μιας και κατόρθωσε να εμφανιστούν όλες οι συχνότητες των χορδών της κιθάρας στην LCD οθόνη [78].

Έπειτα, γίνεται ανάλυση των κυματομορφών μέσα από τον μικροελεγκτή και το κύκλωμα. Πιο συγκεκριμένα, η κιθάρα συνδέθηκε την πρώτη φορά στον παλμογράφο ώστε να εμφανιστεί το σήμα που πρέπει να επεξεργαστεί και έπειτα στην μη

αναστρέφουσα είσοδο του ενισχυτή. Η έξοδος του ενισχυτή συνδέεται στον παλμογράφο ώστε να εμφανιστεί το ενισχυμένο σήμα της κιθάρας. Εν συνεχεία, η έξοδος του ενισχυτή συνδέεται με το βαθυπερατό φίλτρο και η έξοδος αυτό στον παλμογράφο. Το σήμα που εμφανίζεται από την έξοδο του φίλτρου στον παλμογράφο είναι πιο καθαρό και πιο ευκρινές. Μετά, το σήμα διέρχεται από τον ανιχνευτή κορυφής όπου εμφανίζεται στον παλμογράφο πιο απλό, ενώ όταν περάσει και από τον συγκριτή τότε στον παλμογράφο εμφανίζονται τετραγωνικοί παλμοί [78].

Με την συγκεκριμένη εργασία αποδείχτηκε ότι μπορεί να επιτευχθεί ο συντονισμός της κάθε χορδής στην επιθυμητή συχνότητα (Εικόνα 1.62). Όσον αφορά τις βελτιώσεις του συστήματος θα μπορούσε να γίνει προσθήκη μιας εξωτερικής οθόνης η οποία θα εμφανίζει την συχνότητα. Το μόνο πράγμα που δεν επιλύει η εργασία αυτή είναι ο τρόπος με τον θα κουρδιστούν οι χορδές. Ωστόσο, η αρχική ιδέα ήταν η ανίχνευση της κορυφής και η σύγκριση του σήματος να γίνουν μέσω αλγορίθμου, αλλά αυτό έγινε με την χρήση υλικού (Hardware). Επίσης, το κουρδιστήρι λειτουργεί πολύ καλά για τρείς από τις έξι χορδές. Αυτό οφείλεται στον περιορισμό του υλικού (Hardware) και μπορεί να αντιμετωπισθεί με την προσθήκη ειδικών φίλτρων για τις άλλες τρεις χορδές. Τέλος, η ιδέα περί εμφάνισης της τρέχουσας συχνότητας στην LCD οθόνη και η χρήση LED ως κατευθυντήρια ένδειξη για το κούρδισμα της αντίστοιχης χορδής, αντικαταστάθηκε με την εμφάνιση της επιθυμητής συχνότητας και της διαφοράς με την τρέχουσα συχνότητα στην LCD οθόνη [78].



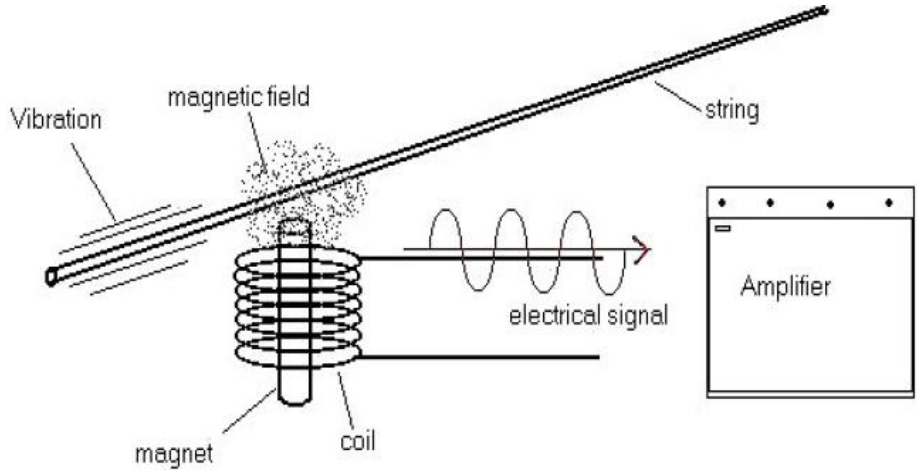
Εικόνα 1.62: Το τελικό κύκλωμα για την ανίχνευση της συχνότητας των χορδών μιας κιθάρας [109].

1.2.8 **Design of an Electric Guitar Tuner**

Στην συγκεκριμένη διπλωματική εργασία, σχεδιάζεται ένα ηλεκτρικό κουρδιστήρι κιθάρας καθώς επίσης διερευνάται και μία νέα μέθοδος εντοπισμού της συχνότητας η οποία είναι πιο αποτελεσματική σε θορυβώδης περιβάλλον. Για την κατασκευή του συστήματος γίνεται χρήση ενός Arduino UNO το οποίο λαμβάνει το ενισχυμένο σήμα από την κιθάρα, εντοπίζει την συχνότητα και μέσω μιας LCD οθόνης καθοδηγεί τον χρήστη για το κούρδισμα της αντίστοιχης χορδής. Η μέθοδος της αυτοσυσχέτισης (autocorrelation) μέσα από την συγκεκριμένη εργασία αποδείχτηκε αποτελεσματική και εύκολη στην εφαρμογή της με το Arduino UNO [79].

Αρχικά, κάθε μία νότα της κιθάρας αποτελείται από μία θεμελιώδη συχνότητα και από πολλές αρμονικές. Ουσιαστικά, οι αρμονικές είναι αυτές που κάνουν την ίδια νότα να ακούγεται με διαφορετικό τόνο στα διάφορα έγχορδα όργανα. Ο τόνος αποτελεί την αναγνώριση του ήχου από τον άνθρωπο ως υψηλότερος ή χαμηλότερος. Η κάθε μία νότα που παίζεται αντιστοιχεί σε μία μοναδική συχνότητα ήχου. Στην πραγματικότητα, η τονικότητα του ήχου δεν έχει ούτε γραμμική ούτε λογαριθμική σχέση με την συχνότητα της νότας. Όμως για την απλότητα της συγκεκριμένης εργασίας ο τόνος και η θεμελιώδης συχνότητα του ήχου θεωρούνται ισοδύναμα. Τα κουρδιστήρια κιθάρας που υπάρχουν στο εμπόριο, ουσιαστικά λαμβάνουν και αναλύουν την τονικότητα της νότας δίνοντας μια οπτική ανατροφοδότηση στον χρήστη για το πόσο απέχει η νότα που παράγεται αυτή την στιγμή από την επιθυμητή [79].

Η συγκεκριμένη εργασία αποσκοπεί στην ανίχνευση του τόνου του ήχου που παράγεται από τους μαγνήτες μιας ηλεκτρικής κιθάρας (Εικόνα 1.63). Πιο συγκεκριμένα, το ταλαντευόμενο σήμα μπορεί να ανιχνευτεί είτε στο πεδίο του χρόνου είτε στο πεδίο της συχνότητας είτε και στα δύο. Η συχνότητα με την οποία ταλαντώνεται το σήμα αποτελεί το ανάστροφο της χρονικής περιόδου. Στην συγκεκριμένη εργασία γίνεται χρήση αλγόριθμου ο οποίος ενεργεί στο πεδίο του χρόνου. Τέτοιου τύπου αλγόριθμοι είναι ο Zero Crossing και η μέθοδος αυτοσυσχέτισης (Autocorrelation). Ουσιαστικά, ο αλγόριθμος Zero Crossing εντοπίζει το σημείο στο οποίο η τάση του σήματος είναι μηδέν. Δηλαδή, ένας κύκλος επανάληψης δημιουργείτε όταν η τάση του σήματος μηδενιστεί για δεύτερη φορά. Με αυτόν τον τρόπο υπολογίζεται η περίοδος του σήματος και κατά συνέπεια η θεμελιώδης συχνότητα της νότας. Η μέθοδος Zero Crossing μπορεί να είναι γρήγορη και φθηνή, αλλά είναι αναποτελεσματική και δεν παρέχει ακρίβεια σε θορυβώδη περιβάλλον. Ωστόσο, για την αύξηση της ακρίβειας της συγκεκριμένης μεθόδου θα πρέπει να γίνει χρήση φίλτρων για την προεπεξεργασία του σήματος. Μία ακόμη χρήση της μεθόδου Zero Crossing, είναι στην επεξεργασία της ομιλίας [79].

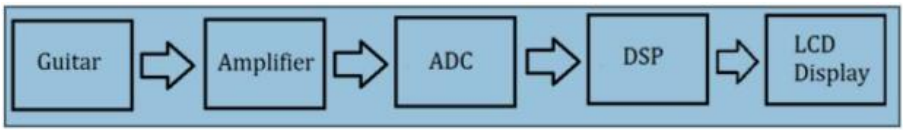


Εικόνα 1.63: Η λειτουργία ενός μαγνήτη κιθάρας [110].

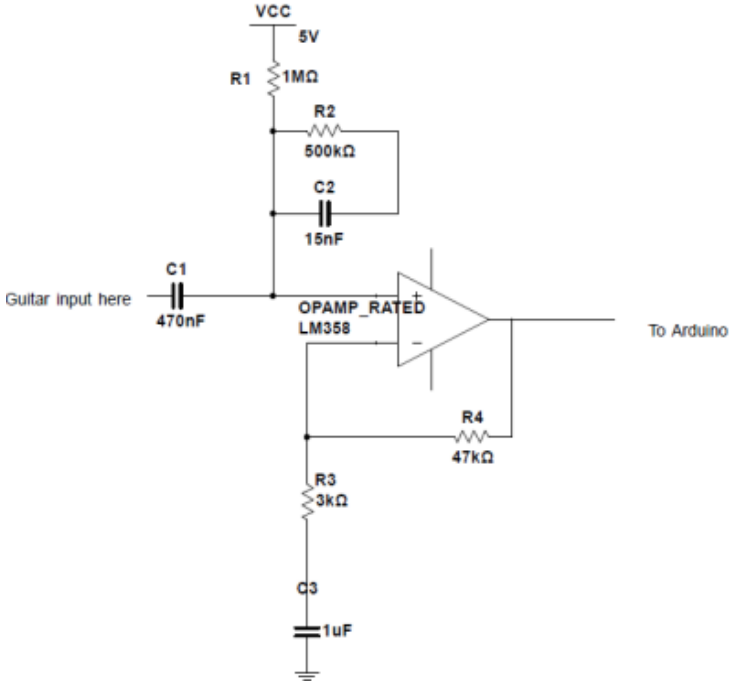
Από την άλλη, η μέθοδος της συσχέτησης αποτελεί μία στατική σύζευξη-συνένωση (Static Association) η οποία βρίσκει πόσο κοντά είναι μία μεταβλητή σε μία άλλη έχοντας γραμμική σχέση. Η μέθοδος της αυτοσυσχέτισης (Autocorrelation) στην επεξεργασία σήματος, συσχετίζει ένα σήμα με το καθυστεριμένο αντίγραφο του εαυτού του ως μία συνάρτηση καθυστέρησης. Πιο απλά, εντοπίζει τις μεταξύ τους ομοιότητες συναρτήσει της μεταξύ τους υστέρησης. Από την συγκεκριμένη εργασία προέκυψε ότι ο αλγόριθμος της αυτοσυσχέτισης εντοπίζει καλύτερα τον τόνο της κιθάρας καθώς μπορεί να εντοπίσει

και τα επαναλαμβανόμενα μοτίβα στα σήματα τα οποία παρεμβάλλονται από θόρυβο ή από αρμονικές [79].

Επιπροσθέτως, το συγκεκριμένο σύστημα αποτελείτε από ένα μπλοκ διάγραμμα το οποίο ορίζει την σειρά με την οποία εκτελούνται οι διεργασίες. Το μπλοκ διάγραμμα έχει ως εξής: Κιθάρα → Ενισχυτής → Αναλογικό-ψηφιακός μετατροπέας → Ψηφιακή επεξεργασία σήματος → LCD οθόνη (Εικόνα 1.64). Πιο συγκεκριμένα, η κιθάρα παράγει το σήμα εσόδου, ενώ η LCD οθόνη παράγει το σήμα εξόδου το οποίο έχει σκοπό την καθοδήγηση του χρήστη για το κούρδισμα της αντίστοιχης χορδής. Ο ενισχυτής που χρησιμοποιήθηκε στην συγκεκριμένη εργασία είναι ο τελεστικός ενισχυτής LM358 ο οποίος έχει σκοπό την ενίσχυση του ασθενούς σήματος που παράγει η κιθάρα ώστε να μπορέσει ο αλγόριθμος να το επεξεργαστεί. Επίσης, ο συγκεκριμένος ενισχυτής παράγει ελάχιστο θόρυβο και χαμηλή παραμόρφωση αρμονικών. Ωστόσο, στην είσοδο του ενισχυτή τοποθετείται ένας πυκνωτής ο οποίος λειτουργεί ως χαμηλοπερατό φίλτρο. Ακόμα για να μπορέσει το σήμα να επεξεργαστεί θα πρέπει να ενισχυθεί με μία DC offset συνιστώσα η οποία θα δημιουργήσει μια μετατόπιση του σήματος (Εικόνα 1.65). Εν συνεχεία, ο αναλογικό-ψηφιακός μετατροπέας αποσκοπεί στην ψηφιακή μετατροπή του αναλογικού σήματος διότι γίνεται πιο εύκολη η επεξεργασία του σήματος δίνοντας πιο ακριβή αποτελέσματα. Όμως θα πρέπει να καθοριστεί η κατάλληλη ανάλυση καθώς και η συχνότητα δειγματοληψίας. Επιπλέον, κατά την ψηφιακή επεξεργασία σήματος ουσιαστικά τρέχει ο αλγόριθμος εντοπισμού της συχνότητας του σήματος που παράγει η κιθάρα. Τόσο η αναλογικό-ψηφιακή μετατροπή όσο και η επεξεργασία του σήματος γίνονται εντός του μικροελεγκτή. Επίσης, η LCD οθόνη παράγει την έξοδο του συστήματος η οποία έχει σκοπό την καθοδήγηση του χρήστη για το κούρδισμα της κιθάρας [79].



<u>Εικόνα 1.64:</u> Το μπλοκ διάγραμμα του συστήματος [111].



Εικόνα 1.65: Το κύκλωμα ενίσχυσης του σήματος της κιθάρας [112].

Σύμφωνα με τα στοιχεία της συγκεκριμένης εργασίας, ο αλγόριθμος της αυτοσυσχέτισης υπολογίζει με μεγάλη ακρίβεια την συχνότητα του σήματος ακόμα και σε θορυβώδες περιβάλλον, αρκεί όμως η θεμελιώδης συχνότητα να είναι αυτή με την μεγαλύτερη ενέργεια. Πάραυτα, το σύστημα δεν λειτούργησε όπως αναμενόταν και αυτό οφείλεται σε πρόβλημα στο κύκλωμα ενίσχυσης. Παρά την αντικατάσταση του τελεστικού ενισχυτή LM358 με τον TL082 το πρόβλημα παρέμεινε. Μία βελτίωση του συγκεκριμένου συστήματος θα μπορούσε να περιλαμβάνει την χρήση σερβοκινητήρων για την περιστροφή των κλειδιών της κιθάρας. Εν συνεχεία, θα μπορούσε να δημιουργηθεί μια εφαρμογή για smartphone η οποία θα τρέχει τον αλγόριθμο μέσω ασύρματης επικοινωνίας Bluetooth και θα ελέγχει τους σερβοκινητήρες με τέτοιον τρόπο ώστε να επιτευχθεί το κούρδισμα της κιθάρας [79].

1.2.9 Design and development of a low cost automatic stringed instrument tuner

Σε αυτή την διπλωματική εργασία, κατασκευάζεται μια συσκευή η οποία αναλύει τις δονήσεις του οργάνου μέσω ενός αλγορίθμου εντοπισμού συχνοτήτων. Η διαδικασία ανάλυσης των δονήσεων του οργάνου προσφέρει μεγάλη ακρίβεια.

Αρχικά, οι μουσικοί αντιλαμβάνονται τον ήχο του μουσικού οργάνου με το οποίο ασχολούνται ξέρουν την διαδικασία κουρδίσματός του. Για την εύρεση του τόνου της κάθε χορδής του οργάνου, γίνεται χρήση αλγορίθμων εντοπισμού της θεμελιώδης συχνότητας. Πιο συγκεκριμένα, υπάρχουν αλγόριθμοι οι οποίοι βασίζονται στο πεδίο του χρόνου και χρησιμοποιούν την μέθοδο της αυτοσυσχέτισης. Δηλαδή, συγκρίνουν το λαμβανόμενο σήμα με ένα χρονικά καθυστερημένο αντίγραφο του εαυτού του. Από την άλλη, υπάρχουν αλγόριθμοι οι οποίοι βασίζονται στο πεδίο της συχνότητας και κάνουν χρήση της μεθόδου Cetrum. Ουσιαστικά, στην μέθοδο Ceptrum εξετάζεται το πλάτος της περιόδου του σήματος κάνοντας χρήση του γρήγορου μετασχηματισμού Fourier (FFT). Στην συγκεκριμένη εργασία αναπτύχθηκε ένα βελτιωμένο και χαμηλού κόστους πρωτότυπο σύστημα κλειστού βρόγχου το οποίο λαμβάνει το ήχο μέσω των δονήσεων του οργάνου [80].

Εν συνεχεία, για την κατασκευή του συστήματος χρησιμοποιήθηκαν αναπτυξιακές μονάδες οι οποίες εξοικονομούν χρόνο ανάπτυξης, κατασκευής και κόστους. Ουσιαστικά, το σύστημα λαμβάνει και επεξεργάζεται το ήχο μέσω των δονήσεων και επενεργεί μέσω του DC κινητήρα στα κλειδιά του οργάνου. Για την λήψη των δονήσεων από το όργανο, χρησιμοποιείτε ένας πιεζοηλεκτρικός μετατροπέας ο οποίος τοποθετείται στην κεφαλή της κιθάρας. Το σήμα που παράγεται από τον πιεζοηλεκτρικό μετατροπέα, προ-επεξεργάζεται από μία πλακέτα ήχου και εν συνεχεία με την χρήση του αλγόριθμου YIN εντοπίζεται η θεμελιώδης συχνότητα του σήματος. Επιπλέον, για να είναι εφικτός ο έλεγχος της περιστροφής του DC κινητήρα, γίνεται χρήση ενός ελεγκτή PID και ενός περιστροφικού κωδικοποιητή τύπου RGB. Επίσης, το σύστημα περιλαμβάνει μία γραφική διεπαφή χρήστη η οποία αποτελείται από μία οθόνη γραφικών και έναν κινητήρα δονήσεων. Ακόμα χρησιμοποιείτε μία Micro SD ως βάση δεδομένων στου συστήματος. Επιπροσθέτως, το Hardware του συστήματος συνδέει όλα τα περιφερειακά πάνω σε μία custome πλακέτα

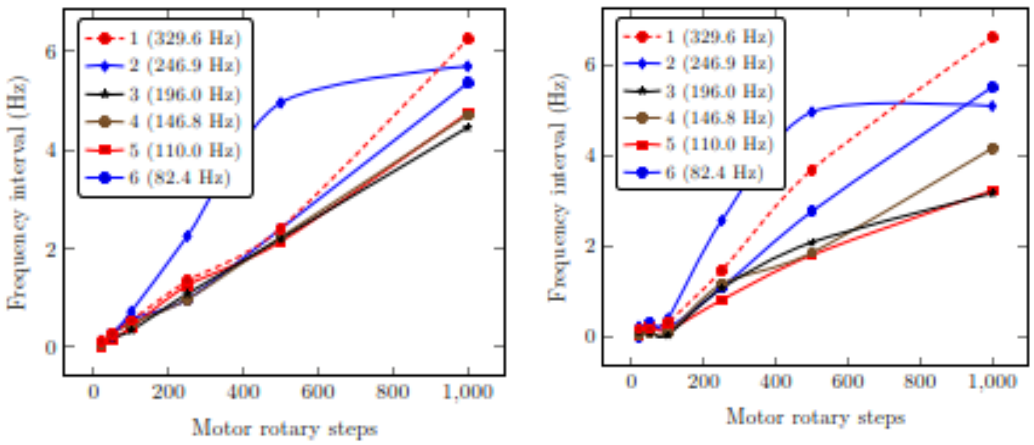
(PCB), η οποία τοποθετείται μέσα σε ένα περίβλημα που κατασκευάστηκε με την χρήση τρισδιάστατου εκτυπωτή. Στην συγκεκριμένη εργασία, χρησιμοποιήθηκε μια ακουστική κιθάρα Fender T-Bucket 300CE TBK η οποία έχει ενσωματωμένη προ-ενίσχυση μέσω των μαγνητών Fishman Isys και περιλαμβάνει τις χορδές Fender Dura-tone (Εικόνα 1.66). Για την ομαλοποίηση του κουρδίσματος χρησιμοποιήθηκε το κουρδιστήρι της Boss TU-03 το οποίο παρέχει ακρίβεια ±1cent και εύρος συχνοτήτων από 27.5Hz (A0) μέχρι 4186Hz (C8). Στην αρχή η κιθάρα κουρδίζεται με το κουρδιστήρι και λαμβάνονται κάποιες εκτιμήσεις της συχνότητας με σκοπό τον υπολογισμό του μέσου όρου. Στην συνέχεια ο μέσος όρος μετατρέπεται σε cents και εμφανίζεται στην οθόνη ώστε να ενημερωθεί ο χρήστης. Έπειτα η συσκευή αναμένει από τον χρήστη να πιέσει τον κωδικοποιητή για να περιστραφεί ο κινητήρας με έναν προκαθορισμένο αριθμό βημάτων. Μόλις ο κινητήρας σταματήσει, τότε εκτελείτε ξανά η ίδια τεχνική για την απόκτηση της συχνότητας. Μετά από αυτή την διάγνωση υπολογίζεται πάλι το διάστημα των συχνοτήτων σε cents και ενημερώνεται ο χρήστης [80].

Εικόνα 1.66: Η ακουστική κιθάρα Fender T-Bucket 300CE TBK [113].

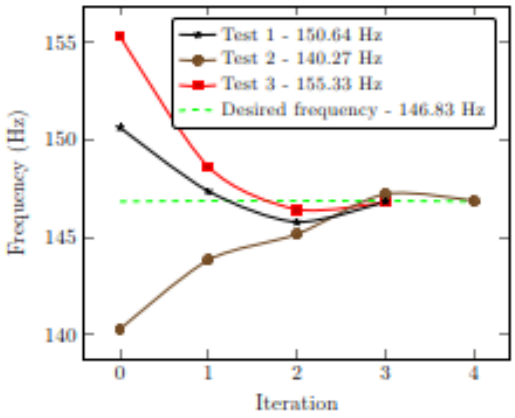
Στο πειραματικό μοντέλο, χρησιμοποιείτε το κουρδιστήρι και μία σειριακή επικοινωνία για την εξαγωγή δεδομένων που λαμβάνονται κατά την διαδικασία. Έπειτα, δοκιμάζεται να κουρδιστεί η χορδή PE (D3) με το πρωτότυπο σύστημα το οποίο συνδέεται και αυτό σειριακά με τον υπολογιστή για την εξαγωγή των δεδομένων που παράγονται (Εικόνα 1.67). Ωστόσο, γίνεται χρήση και του κουρδιστηριού της Boss για να συγκριθούν τα αποτελέσματα. Ο κλειστός βρόγχος που έχει δημιουργηθεί δέχεται ως είσοδο την θεμελιώδη συχνότητα της χορδής και παράγει στην έξοδό του το κατάλληλο σήμα για τον έλεγχο του κινητήρα της αντίστοιχης χορδής. Ουσιαστικά, το σήμα ανάδρασης του κλειστού βρόγχου δίνεται από τον μορφομετατροπέα δονήσεων. Ο χρήστης είναι αυτός που επιλέγει την συχνότητα στο σύστημα μέσα από την επιλογή της χορδής που επιθυμεί να κουρδίσει. Ανάλογα με την διαφορά από την επιθυμητή συχνότητα, ο κινητήρας περιστρέφει το κλειδί της αντίστοιχης χορδής προς την αντίστοιχη κατεύθυνση. Μόλις σταματήσει ο κινητήρας τότε επαναλαμβάνεται η διάγνωση του κουρδίσματος της συγκεκριμένης χορδής και στην περίπτωση που είναι ίδια με την επιθυμητή, τότε η διαδικασία ολοκληρώνεται. Στην περίπτωση όμως που η διάγνωση δώσει διαφορετική συχνότητα από την επιθυμητή, τότε η διαδικασία διόρθωσης επαναλαμβάνεται μέχρι να επιτευχθεί η επιθυμητή συχνότητα [80].

Εικόνα 1.67: Χρήση του πρωτότυπου συστήματος για το κούρδισμα της PE (D3) χορδής [114].

Τέλος, σε όλα τα κλειδιά της κιθάρας μελετήθηκε η τάση της χορδής και η τριβή του κάθε κλειδιού και στις δύο κατευθύνσεις περιστροφής. Τα δεδομένα που λήφθηκαν μέσω της σειριακής επικοινωνίας αναπαριστώνται σε διαγράμματα. Πιο συγκεκριμένα, στα διαγράμματα αυτά αναπαριστάτε η μεταβολή της συχνότητας της κάθε χορδής ανάλογα με την αυξομείωση της τάσης της χορδής (Εικόνα 1.68). Δηλαδή αναπαρίσταστε η αντίδραση της συχνότητας όταν ασκείται επίδραση στο αντίστοιχο κλειδί. Παρατηρήθηκε ότι όλες οι χορδές εμφανίζουν παρόμοιες γραφικές. Επίσης, τα αποτελέσματα από τις δοκιμές του αυτόματου κουρδίσματος της κιθάρας τα οποία λαμβάνονται μέσω της σειριακής επικοινωνίας παρουσιάζονται και αυτά σε διάγραμμα. Ουσιαστικά, από το διάγραμμα προκύπτει ότι η χορδή PE (D3) κουρδίζεται επιτυχώς μέσα σε 10 sec με λίγες επαναλήψεις (Εικόνα 1.69). Σύμφωνα με τις αναλύσεις των αποτελεσμάτων, τι πρωτότυπο σύστημα έχει πολύ καλή ακρίβεια και ο εντοπισμός της θεμελιώδης συχνότητας μέσω των δονήσεων είναι ικανοποιητικός. Ωστόσο, το πρωτότυπο σύστημα προσφέρει εργονομία, αισθητική και λειτουργικότητα. Όσον αφορά τις βελτιώσεις του συγκεκριμένου συστήματος, θα μπορούσε να ενσωματωθούν νέες μουσικές κλίμακες και εναλλακτικά κουρδίσματα. Επίσης, μία καλή ιδέα θα ήταν το ίδιο σύστημα να χρησιμοποιηθεί και σε άλλα μουσικά όργανα πιο πολύπλοκα όπως το πιάνο και τα ντραμς [80].



Εικόνα 1.68: Τα διαγράμματα μεταβολής της συχνότητας συναρτήσει της μείωσης της τάσης (αριστερά) και της αύξησης της τάσης (δεξιά) των χορδών [115].



Εικόνα 1.69: Το διάγραμμα της μεταβολής της συχνότητας της χορδής PE (D3) συναρτήσει των επαναλήψεων της διαδικασίας [116].

1.2.10 Automatic Guitar Tuner

Στην συγκεκριμένη εργασία, σχεδιάζεται και αναπτύσσεται ένα σύστημα αυτομάτου κουρδίσματος κιθάρας το οποίο λαμβάνει τον τόνο και περιστρέφει κατάλληλα το κλειδί της αντίστοιχης χορδής. Η διαδικασία κουρδίσματος ξεκινά από την τάση της επιλεγμένης χορδής, την διέγερσή της για την παραγωγή του ήχου, την σύλληψη του σήματος με σκοπό την ψηφιοποίηση και την σύγκριση του με το επιθυμητό σήμα ώστε να ρυθμιστεί η τάση της χορδής κατάλληλα. Αυτή η διαδικασία επαναλαμβάνεται για όλες τις χορδές. Επιπλέον, στην συγκεκριμένη εργασία αναλύεται το φαινόμενο της παραγωγής του ήχου από μία χορδή, ο μηχανισμός που ρυθμίζει την τάση της καθώς και το μαθηματικό μοντέλο το οποίο ταξινομεί τον ήχο ως νότα. Επίσης, γίνεται αναφορά στις μεθόδους με τις οποίες μπορεί να ληφθεί ο ήχος και να μετατραπεί σε ηλεκτρικό σήμα. Ωστόσο, η επιλογή συστήματος για την λήψη και την ενίσχυση του ηχητικού κύματος είναι καίριας σημασίας πριν την ψηφιοποίηση του σήματος. Το σύστημα αυτομάτου κουρδίσματος χρησιμοποιεί τον μικροελεγκτή του Arduino για τον εντοπισμό της συχνότητας και για τον έλεγχο περιστροφής του κινητήρα. Ακόμα, αναλύεται η κατασκευή της βάσης η οποία στηρίζει όλο το σύστημα στην κεφαλή της κιθάρας καθώς και ενός αντάπτορα ο οποίος εφαρμόζει στα κλειδιά της κιθάρας και μεταφέρει την περιστροφή του άξονα του κινητήρα στο κλειδί [81].

Στα έγχορδα όργανα για να ακουστούν σωστά οι νότες θα πρέπει να η να έχει ρυθμιστεί κατάλληλα η τάση των χορδών. Όμως με το πέρασμα του χρόνου οι χορδές φθείρονται με αποτέλεσμα να ξεκουρδίζονται. Υπάρχουν πολλές συσκευές οι οποίες εντοπίζουν τη συχνότητα και καθοδηγούν τον χρήστη για το κούρδισμα των χορδών. Το ηλεκτρομηχανικό σύστημα το οποίο αναπτύσσεται στην συγκεκριμένη εργασία λαμβάνει και συγκρίνει το σήμα της κάθε χορδής με το επιθυμητό και ανάλογα ενεργοποιεί τον κατάλληλο κινητήρα και περιστρέφει το αντίστοιχο κλειδί της κιθάρας. Αυτό θα έχει ως αποτέλεσμα το κούρδισμα του οργάνου να γίνεται εύκολα, γρήγορα και αποτελεσματικά. Ωστόσο, στόχος της συγκεκριμένης εργασίας είναι ο κατάλληλος σχεδιασμός του μηχανικού μέρους ώστε να μην επηρεάζεται η λειτουργικότητα του συστήματος, ο εντοπισμός της καλύτερης λύσης για την καταγραφή του ήχου, η ολοκλήρωση του κουρδίσματος το συντομότερο δυνατόν καθώς και η κατάλληλη τοποθέτηση του συστήματος χωρίς να επηρεάζεται η κιθάρα. Η ευχρηστία και η τοποθέτηση του συστήματος σε οποιαδήποτε κιθάρα είναι ένας ακόμα στόχος της εργασίας [81].

Εν συνεχεία, τα ηχητικά κύματα διαδίδονται με την ταλάντωση της πίεσης του αέρα και μέσω του ανθρώπινου αυτιού γίνονται αντιληπτά στον εγκέφαλο. Επίσης, ο ήχος μπορεί κα αντανακλάται στις διάφορες επιφάνειες με αποτέλεσμα να δημιουργούνται διάφορα εφέ όπως της ηχούς και της παραμόρφωσης. Τα χαρακτηριστικά τα οποία κάνουν ξεχωριστό τον ήχο που παράγεται από ένα μουσικό όργανο είναι το ύψος του ήχο, ο τόνος και η ένταση. Πιο συγκεκριμένα το ύψος του ήχου σχετίζεται με το πόσο συχνό δονείτε το μέσω διάδοσης, δηλαδή σχετίζεται με την συχνότητα. Η ένταση του ήχου σχετίζεται με το ποσότητα ενέργειας που μεταδίδεται σε μια επιφάνεια ανά μονάδα χρόνου, ή αλλιώς με το πόσο δυνατός ή απαλός είναι ο ήχος. Ο τόνος είναι το χαρακτηριστικό το οποίο μπορεί να ξεχωρίσει τον ήχο όταν δύο όργανα παίζουν την ίδια νότα με ίδιο ύψος και ένταση. Ουσιαστικά το χαρακτηριστικό του τόνου δίνεται από τις αρμονικές συχνότητες που βρίσκονται παράπλευρα της θεμελιώδης συχνότητας. Πιο συγκεκριμένα, στα έγχορδα όργανα, όταν διεγείρεται μία χορδή τότε δημιουργείτε ένα στάσιμο κύμα μεταξύ δύο κόμβων και η ελάχιστη συχνότητα με την οποία ταλαντώνεται ονομάζεται θεμελιώδης συχνότητα. Όλοι οι υπόλοιποι τρόποι ταλάντωσης είναι πολλαπλάσια της θεμελιώδης συχνότητας και ονομάζονται αρμονικές συχνότητες. Επομένως, για να κουρδιστεί η χορδή θα πρέπει να περιστραφεί το αντίστοιχο κλειδί μέχρι να επιτευχθεί η επιθυμητή συχνότητα. Με αυτό τον τρόπο προσαρμόζονται η τάση της χορδής και η γραμμική πυκνότητα μάζας τα οποία είναι αντιστρόφως ανάλογα μεγέθη. Πιο συγκεκριμένα, όσο αυξάνεται η τάση, η γραμμική πυκνότητα μάζας μειώνεται με αποτέλεσμα να παράγεται ήχος υψηλότερης συχνότητας. Επίσης, η περιστροφή του κλειδιού επιδρά σε ένα σύστημα γραναζιών το οποίο περιστρέφει την κεφαλή στην οποία τυλίγεται η χορδή. Ανάλογα με την φορά περιστροφής η χορδή σφίγγει ή χαλαρώνει επηρεάζοντας έτσι την τάση της, την γραμμική πυκνότητα μάζας και κατά συνέπεια την συχνότητα ταλάντωσής της [81].

Από έρευνες έχει προκύψει ότι το ανθρώπινο αυτί μπορεί και αντιλαμβάνεται ήχους με συχνότητα από 16Hz έως 20,000Hz. Στην μουσική, υπάρχουν δώδεκα νότες οι οποίες δημιουργούν μία οκτάβα. Επίσης, δύο ίδιες νότες σχηματίζουν μία οκτάβα όταν η συχνότητα της μίας είναι διπλάσια της άλλης. Η διαφορά των συχνοτήτων μεταξύ δύο νοτών είναι λογαριθμική, δηλαδή όσο μεγαλώνει η συχνότητα, τόσο μεγαλώνει η διαφορά των συχνοτήτων μεταξύ δύο νοτών. Πάραυτα, η διαφορά αυτή στον ανθρώπινο αυτί ακούγεται ίδια σε οποιαδήποτε οκτάβα. Το κλασσικό standard κούρδισμα της κιθάρας ξεκινάει από την Μι (E) της δεύτερης οκτάβας και φτάνει μέχρι την Μι (E) της τέταρτης οκτάβας [81].

Για όλα τα μουσικά όργανα, υπάρχουν κουρδιστήρια τα οποία αναγνωρίζουν την νότα που παράγεται και καθορίζουν αν το αντίστοιχο όργανο είναι κουρδισμένο ή όχι. Τα πιο φθηνά κουρδιστήρια χρησιμοποιούν μια LED οθόνη η οποία δείχνει αν ο ήχος που παράγεται από το όργανο είναι υψηλότερος ή χαμηλότερος από την επιθυμητή τιμή. Από την άλλη τα πιο ακριβά κουρδιστήρια δείχνουν την διαφορά μεταξύ της πραγματικής και της επιθυμητής συχνότητας. Όμως, ο χρήστης είναι αυτό που πρέπει να ρυθμίσει την τάση των χορδών σύμφωνα με το κουρδιστήρι. Γι' αυτό τον λόγο στην συγκεκριμένη εργασία, δημιουργείτε ένα σύστημα το οποίο λαμβάνει την συχνότητα που εκπέμπει το όργανο και εκτελεί μία μηχανική διεργασία για το κούρδισμα του οργάνου [81].

Πριν την εύρεση των κουρδιστηριών, υπήρχε το διαπασών (Εικόνα 1.70) για το κούρδισμα των οργάνων. Πιο συγκεκριμένα, το διαπασών διεγείρεται μόλις χτυπηθεί παράγοντας συνήθως την νότα Λα (A) 440Hz χωρίς τις αρμονικές συχνότητες. Ο μουσικός όμως ήταν αυτό που έπρεπε να συγκρίνει τους ήχους από το διαπασών και από το όργανο

ώστε να διακρίνει αν είναι κουρδισμένο ή όχι και να πράξει ανάλογα. Όμως αυτός ο τρόπος κουρδίσματος παίρνει πολύ περισσότερο χρόνο [81].

Εικόνα 1.70: Διαπασών [117]

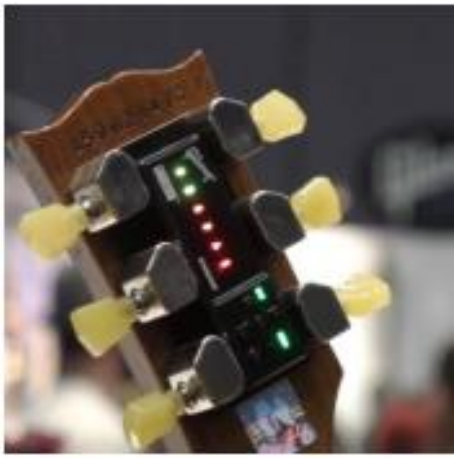
Από την άλλη, τα κουρδιστήρια με βελόνα (Εικόνα 1.71) μπορεί να αντιληφθούν τον ήχο είτε ακουστικά, είτε μέσω καλωδίου Jack αν και εφόσον μπορεί να το υποστηρίξει η κιθάρα. Σε αυτού του τύπου τα κουρδιστήρια, όταν η βελόνα είναι κάθετη στην μέση, τότε η χορδή είναι κουρδισμένη στην επιθυμητή νότα. Στην περίπτωση όμως που η βελόνα έχει απόκλιση προς τα αριστερά ή προς τα δεξιά, τότε η συχνότητα του ήχου που λαμβάνεται είναι χαμηλότερη ή υψηλότερη αντίστοιχα. Ωστόσο, τα κουρδιστήρια αυτά αντί για βελόνα μπορούν να έχουν κάποιο LED. Ο τρόπος λήψης του σήματος στα συγκεκριμένα κουρδιστήρια παίζει σημαντικό ρόλο στην ποιότητα του κουρδίσματος. Ουσιαστικά, αν το σήμα λαμβάνεται με την χρήση μικροφώνου, τότε το σήμα περιλαμβάνει θόρυβο και το κούρδισμα δεν γίνεται με ακρίβεια. Αν όμως χρησιμοποιηθεί η θύρα Jack του κουρδιστηριού τότε ο θόρυβος αγνοείτε και το κούρδισμα γίνεται με μεγαλύτερη ακρίβεια [81].

Εικόνα 1.71: Κουρδιστήρι με βελόνα [118]

Μία άλλη κατηγορία κουρδιστηριών είναι αυτά που τοποθετούνται πάνω στο όργανο με την χρήση ενός σφιγκτήρα και αντιλαμβάνονται τον ήχο μέσα από τις δονήσεις του ξύλου (Εικόνα 1.72). Δηλαδή, αντιλαμβάνονται το μηχανικό κύμα μέσα από το σώμα του οργάνου και από εκεί μεταδίδεται στο κουρδιστήρι. Το κούρδισμα που γίνεται είναι πιο αξιόπιστο μιας και αγνοείτε ο θόρυβος από το περιβάλλον [81].

Εικόνα 1.72: Κουρδιστήρι με σφιγκτήρα [119]

Επιπλέον, υπάρχουν και αυτόματα κουρδιστήρια τα οποία λειτουργούν με μπαταρίες και μέσω ρομποτικών ακροδεκτών κουρδίζουν το όργανο. Γι' αυτήν την κατηγορία κουρδιστηριών υπάρχουν δύο παραλλαγές. Η μία έχει το κουρδιστήρι πάνω στην κεφαλή της κιθάρας και με την χρήση ρομποτικών κλειδιών κουρδίζονται οι χορδές του οργάνου. Ένα τέτοιο σύστημα χρησιμοποιεί η Gibson στις κιθάρες της και ονομάζεται Power Tune και κατασκευάζεται από την εταιρία Tronical (Εικόνα 1.73). Το σύστημα αυτό μπορεί να τοποθετηθεί σχεδόν σε οποιαδήποτε μοντέλο κιθάρας και παρέχει γρήγορο και ακριβές κούρδισμα. Η δεύτερη παραλλαγή, χρησιμοποιεί ένα μικρόφωνο για την λήψη του ήχου και μία ξεχωριστή συσκευή η οποία τοποθετείται με την βοήθεια του χρήστη στο κλειδί το οποίο επιθυμεί να κουρδίσει. Εν συνεχεία ο χρήστης διεγείρει την αντίστοιχη χορδή και η συσκευή περιστρέφει το κλειδί μέχρι η χορδή να αποκτήσει την επιθυμητή συχνότητα. Ένα τέτοιο κουρδιστήρι είναι το automatic crank tuner της εταιρίας Jowoom (Εικόνα 1.74). Όμως, η δεύτερη παραλλαγή δεν παρέχει καλή ακρίβεια κουρδίσματος εξαιτίας του θορύβου από το περιβάλλον [81].



Εικόνα 1.73: Το αυτόματο κουρδιστήρι Power Tune από την Tronical που χρησιμοποιεί η Gibson [120]

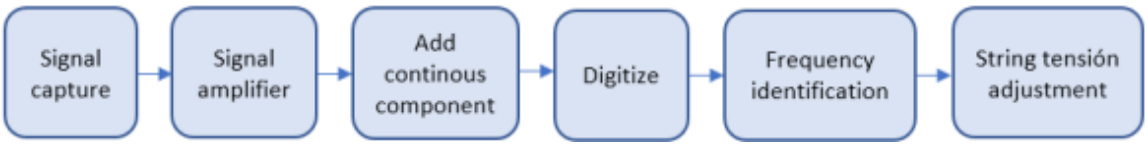
Εικόνα 1.74: Το κουρδιστήρι Automatic crank tuner από την εταιρία Jowoom [121]

Στόχοι της συγκεκριμένης εργασίας είναι η εγκατάσταση του αυτόματου κουρδιστηριού να μην τροποποιεί την κιθάρα, χαμηλό κόστος και το σύστημα σύσφιξης που θα χρησιμοποιηθεί για την τοποθέτησή του να παρέχει αξιόπιστη λήψη του σήματος μέσω των δονήσεων. Επίσης, το σύστημα εφαρμόστηκε σε μία κιθάρα epiphone δεν θα πρέπει ο συνολικός χρόνος συναρμολόγησης να ξεπερνάει τα 120sec. Εν συνεχεία, ο χρόνος κουρδίσματος δεν θα πρέπει να ξεπερνάει τα 60sec. Ωστόσο, η διαδικασία κουρδίσματος του οργάνου θα πρέπει να είναι απλή και διαισθητική χωρίς να απαιτείται η εμπειρία του χρήστη. Επίσης, το μέγεθος του συστήματος δεν θα πρέπει να ξεπερνάει το μέγεθος της κεφαλής του οργάνου και η διαφορά της πραγματικής και της επιθυμητής συχνότητας θα πρέπει να είναι μικρότερη από 5Hz για να ολοκληρωθεί το κούρδισμα [81].

Εν συνεχεία, για τον υπολογισμό της συχνότητας υπάρχουν μέθοδοι τόσο στο πεδίο του χρόνου όσο και στο πεδίο της συχνότητας. Για τον υπολογισμό της συχνότητας στο πεδίο των συχνοτήτων γίνεται χρήση είτε του διακριτού μετασχηματισμού Fourier (DFT) ή του γρήγορου μετασχηματισμού Fourier (FFT). Ουσιαστικά, ο μετασχηματισμός Fourier λαμβάνει ένα πλαίσιο δεδομένων μιας συγκεκριμένης χρονικής στιγμής και ως αποτέλεσμα δίνει το φάσμα των συχνοτήτων των δεδομένων. Ωστόσο, για την εύρεση της θεμελιώδης συχνότητας θα πρέπει να υπολογιστεί το πλάτος της μεγαλύτερης κορυφής στο φάσμα των συχνοτήτων. Όλες οι υπόλοιπες συχνότητες γύρω από την θεμελιώδη συχνότητα είναι οι αρμονικές συχνότητες και δίνουν την χαρακτηριστική χροιά. Σύμφωνα με το θεώρημα Nyquist-Shannon θα πρέπει ο ρυθμός δειγματοληψίας να είναι τουλάχιστον διπλάσιος του εύρους ζώνης ώστε να μπορέσει το σήμα να ανακατασκευαστεί με ακρίβεια. Τα έγχορδα όργανα κατατάσσονται στα δυναμικά συστήματα διότι με το πέρασμα του χρόνου αλλάζουν κατάσταση δηλαδή μεταβάλλεται η συχνότητα των χορδών. Στην εργασία αυτή, ελέγχεται η τάση της χορδής με σκοπό να ρυθμιστεί η συχνότητα της ταλαντευμένης χορδής. Τα συστήματα αυτομάτου ελέγχου περιλαμβάνουν ένα βρόγχο ανάδρασης, μία αναφορά στην είσοδο του συστήματος και έναν ελεγκτή ο οποίος επιχειρεί να μεταφέρει την έξοδο του συστήματος στην επιθυμητή τιμή. Αν η ανάδραση είναι θετική τότε το σύστημα είναι ένας ταλαντωτής, ενώ αν η ανάδραση είναι αρνητική τότε το σύστημα είναι ένας αντισταθμιστής [81].

Για τον σχεδιασμό του πρωτότυπου θα πρέπει να γίνει χρήση μικροελεγκτή ο οποίος θα ελέγχει τις ηλεκτρονικές συσκευές καθώς και άλλες διαδικασίες. Μερικές τεχνολογικές πτυχές που πρέπει να καλύπτει ένας μικροελεγκτής είναι το χαμηλό κόστος, να περιλαμβάνει πολλές εισόδους και εξόδους και τα εργαλεία για την εφαρμογή τους να

είναι απλά. Για να μπορέσει ο μικροελεγκτής να ικανοποιήσει το θεώρημα Nyquist-Shannon θα πρέπει να έχει τουλάχιστον έξι ακροδέκτες εισόδου και ελάχιστη συχνότητα δειγματοληψίας 990Hz. Για την συγκεκριμένη εργασία επιλέχθηκε ο μικροελεγκτής Arduino UNO λόγου του χαμηλού κόστους και του εύκολου προγραμματισμού. Επειδή όμως δεν μπορεί να υποστηρίξει την μέθοδο του μετασχηματισμού Fourier θα πρέπει να γίνει χρήση άλλης μεθόδου για τον εντοπισμό της συχνότητας. Η σειρά με την οποία θα εκτελούνται οι διαδικασίες είναι η εξής: σύλληψη του σήματος, ενίσχυση του σήματος, προσθήκη συνεχών εξαρτημάτων, ψηφιοποίηση, εντοπισμός συχνότητας και ρύθμιση της τάσης της χορδής (Εικόνα 1.75). Κάθε μία διαδικασία θα πρέπει να σχεδιαστεί ξεχωριστά για να επιτευχθεί η διαδικασία κουρδίσματος. Για την διαδικασία λήψης του ακουστικού σήματος από το όργανο θα πρέπει να επιλεχθεί ένα εξάρτημα που θα το μετατρέπει σε ηλεκτρικό σήμα με όσο το δυνατόν λιγότερο θόρυβο ώστε να το επεξεργαστεί ο μικροελεγκτής. Για την συγκεκριμένη εργασία επιλέχθηκε το μικρόφωνο επαφής Korg cm200 το οποίο είναι χαμηλού κόστους, μικρό σε μέγεθος και παρέχει την απαραίτητη ποιότητα ώστε να μπορέσει να γίνει ανίχνευση της συχνότητας (Εικόνα 1.76). Ουσιαστικά, το μικρόφωνο επαφής με την χρήση ενός πιεζοηλεκτρικού υλικού λαμβάνει το μηχανικό κύμα που μεταδίδεται μέσω του ξύλου. Με αυτόν τον τρόπο αντιμετωπίζεται ο θόρυβος από το περιβάλλον. Το σήμα που παράγεται από το μικρόφωνο επαφής είναι ασθενές και γι' αυτό θα πρέπει να ενισχυθεί και να μετατοπιστεί ώστε να μπορέσει ο μικροελεγκτής να το επεξεργαστεί. Το κύκλωμα ενίσχυσης και μετατόπισης της συγκεκριμένης άσκησης έγινε σύμφωνα με την μελέτη της Amanda Ghassaei [81].



Εικόνα 1.75: Η σειρά εκτέλεσης των διεργασιών [122]

Εικόνα 1.76: Το μικρόφωνο επαφής Korg cm200 [123]

Το Arduino UNO θα πρέπει να εντοπίσει την συχνότητα του σήματος που έχει προκύψει από την δειγματοληψία και αφού καθορίσει την τρέχουσα συχνότητα θα πρέπει να ορίσει το πόσο θα πρέπει να περιστραφεί το αντίστοιχο κλειδί. Επειδή το Arduino UNO δεν μπορεί να χρησιμοποιήσει την μέθοδο εντοπισμού της συχνότητας μέσω Fourier γίνεται χρήση του αλγόριθμου που δημιουργήθηκε από την Amanda Ghassaei. Πιο συγκεκριμένα, το σήμα εισόδου είναι μια σύνθετη κυματομορφή η οποία πρέπει να διαχωριστεί σε απλά ημιτονοειδή κύματα. Ωστόσο, το ημιτονοειδές κύμα με το μεγαλύτερο πλάτος καθορίζει την συχνότητα του σύνθετου κύματος. Αυτό έχει ως

αποτέλεσμα η συχνότητα των γεγονότων να συμπίπτει με την θεμελιώδη συχνότητα του σήματος. Επίσης, για την συγκεκριμένη εργασία επιλέχθηκε ένας μονοπολικός βηματικός κινητήρας διότι παρέχει μεγάλη ακρίβεια θέσης και επαναληπτικότητα και δεν απαιτείτε η χρήση κωδικοποιητή. Ο μονοπολικός βηματικός κινητήρας περιέχει τέσσερα πηνία και μπορεί να ελεγχθεί με τρεις διαφορετικούς τρόπους. Ο ένας τρόπος είναι δίνοντας ρεύμα ταυτόχρονα σε δύο πηνία για κάθε βήμα επιτυγχάνοντας μέγιστη ροπή και μέγιστη κατανάλωση. Ο άλλος τρόπος είναι διεγείροντας ένα πηνίο την φορά για ένα βήμα επιτυγχάνοντας έτσι μέτρια κατανάλωση και μέτρια ροπή. Ο τρίτος τρόπος χρησιμοποιεί τους δύο προηγούμενους τρόπους, δηλαδή μία φορά διεγείρει ένα πηνίο και την επόμενη διεγείρει δύο πηνία. Με αυτόν τον τρόπο γίνεται κίνηση μισού βήματος και επιτυγχάνεται μία ομαλή κίνηση με μέσω όρο ροπής και κατανάλωσης από τους προηγούμενους δύο τρόπους. Ωστόσο, ο βηματικός κινητήρας θα μπορούσε να ελεγχθεί χειροκίνητα, δηλαδή το κάθε πηνίο να συνδεθεί σε ένα μία ψηφιακή έξοδο του Arduino και ανάλογα να γραφτεί ο επιθυμητός αλγόριθμος. Επιπλέον, για οποιανδήποτε τρόπο θα πρέπει να γίνει χρήση οδηγού κινητήρων διότι οι ψηφιακοί ακροδέκτες του Arduino δεν παρέχουν τις ιδανικές τιμές τάσης και ρεύματος ώστε να λειτουργήσουν οι κινητήρες. Πιο συγκεκριμένα, οι οδηγοί βηματικών κινητήρων είναι ενισχυτές ρεύματος οι οποίοι λαμβάνουν ένα σήμα μικρού ρεύματος και το μετατρέπουν σε ένα σήμα υψηλού ρεύματος ώστε να καταφέρουν να οδηγήσουν το κινητήρα. Για την διεκπεραίωση της συγκεκριμένης εργασίας, χρησιμοποιείτε η κανονική κίνηση του βηματικού κινητήρα και ο προγραμματισμός του γίνεται χειροκίνητα. Στην συγκεκριμένη εργασία η επιλογή του βηματικού κινητήρα έγινε με βάση την ροπή η οποία πρέπει να είναι μεγαλύτερη από αυτήν που χρειάζεται το κλειδί για να περιστραφεί. Ο βηματικός κινητήρας που επιλέχθηκε είναι ο 28BYJ-48 (Εικόνα 1.77) ο οποίος περιλαμβάνει 64 βήματα και ο τύπος οδηγού κινητήρα που επιλέχθηκε είναι ο ULN2003 (Εικόνα 1.78) [81].

Εικόνα 1.77: Ο βηματικός κινητήρας 28BYJ-48 [124]

Εικόνα 1.78: Ο οδηγός του βηματικού κινητήρα ULN2003 [125]

Ακόμα, για την μετάδοση της περιστροφής του άξονα στο κλειδί της κιθάρας σχεδιάστηκε και κατασκευάστηκε ένας αντάπτορας ο οποίος εφαρμόζει από την μία πλευρά στον άξονα και από την άλλη πλευρά στο κλειδί (Εικόνα 1.79). Ο σχεδιασμός του αντάπτορα έγινε στο CAD Creo και τυπώθηκε από έναν τρισδιάστατο εκτυπωτή. Επίσης, η ταχύτητα και η φορά περιστροφής του κινητήρα και κατά συνέπεια του κλειδιού, εξαρτάται από την διαφορά της πραγματικής συχνότητας με την επιθυμητή. Πιο συγκεκριμένα, όταν η διαφορά της επιθυμητής συχνότητας με την πραγματική είναι 10Hz, τότε ο κινητήρας θα πρέπει να περιστρέψει το κλειδί κατά ένα τέταρτο, ενώ όταν η διαφορά είναι 5Hz, τότε ο κινητήρας θα περιστρέψει το κλειδί κατά ένα όγδοο. Με αυτό τον τρόπο επιτυγχάνεται η μείωση του χρόνου συντονισμού του οργάνου [81].

Εικόνα 1.79: Ο βηματικός κινητήρας με τον αντάπτορα για την περιστροφή του κλειδιού [126]

Επιπροσθέτως, σχεδιάστηκε μία βάση εφαρμόζει επάνω στην κεφαλή της κιθάρας και περιέχει τον κινητήρα μαζί με τον αντάπτορα (Εικόνα 1.80). Η στερέωση του κουρδιστηριού είναι σημαντική διότι πρέπει να περιορίζονται τυχόν κινήσεις κατά την διάρκεια του κουρδίσματος. Ωστόσο, η βάση σχεδιάστηκε με τέτοιον τρόπο ώστε να μην εμποδίζει την περιστροφή του αντάπτορα. Η σχεδίαση έγινε και αυτή με το CAD Creo και τυπώθηκε από τρισδιάστατο εκτυπωτή με τεχνολογία Multifusion από την HP. Ο βηματικός κινητήρας βιδώνεται πάνω στην βάση και ο αντάπτορας κολλήθηκε με κόλλα πάνω στον άξονα του κινητήρα. Για την επίτευξη της μέγιστης ροπής η ακολουθία ενεργοποίησης των πηνίων του βηματικού κινητήρα έγινε χειροκίνητα. Επίσης, κατά την δοκιμή του κινητήρα ελέγχθηκε ότι η δομή κατά την περιστροφή παραμένει σταθερή και τοποθετήθηκε ένα εξωτερικό κουρδιστήρι ώστε να ελεγχθεί πόσο κοντά στην επιθυμητή συχνότητα κουρδίστηκε η χορδή [81].

Εικόνα 1.80: Η βάση για την εφαρμογή του κινητήρα με τον αντάπτορα επάνω στην κεφαλή της κιθάρας [127]

Ακόμα, γίνεται αναφορά στον χρόνο που αφιερώθηκε για κάθε μία διεργασία μέχρι την ολοκλήρωση της εργασίας. Επίσης, για την κατασκευή του συστήματος χρησιμοποιήθηκε πλαστικό για την εκτύπωση της βάσης και κάποια ηλεκτρονικά εξαρτήματα τα οποία αποτελούνται από τοξικά υλικά που μπορούν να βλάψουν την ανθρώπινη υγεία. Επιπλέον, γίνεται μία ανάλυση του συνολικού κόστους για την υλοποίηση του συστήματος [81].

Τέλος, γίνεται ανάλυση των συμπερασμάτων της εργασίας τα οποία ικανοποιούν τις αρχικές απαιτήσεις. Πιο συγκεκριμένα, το σύστημα είναι στιβαρό, λειτουργικό, φθηνό, απλό, αποτελεσματικό στο κούρδισμα των χορδών και ικανοποιητική επαναληψιμότητα (Εικόνα 1.81). Για την εκβιομηχάνιση του συστήματος θα πρέπει να γίνει προσαρμογή σε κάθε μοντέλο κιθάρας. Ωστόσο, το Firmware, τα ηλεκτρονικά και ο αλγόριθμος μπορούν να παραμείνουν ίδια για οποιοδήποτε έγχορδο όργανο. Επίσης, η προσαρμογή του συστήματος, επηρεάζει και την βάση στήριξης του κινητήρα και τον αντάπτορα. Ωστόσο, με τις νέες τεχνολογίες η κατασκευή των μηχανικών μερών θα μπορούσε να γίνει με μικρότερο όγκο και έτσι το σύστημα θα ήταν πιο εύκολο να εμπορευματοποιηθεί. Επιπροσθέτως, το σύστημα αυτομάτου κουρδίσματος δεν απαιτεί μουσική εμπειρία από τον χρήστη, ωστόσο υπάρχουν αμφιβολίες για την αντίληψη της αγοράς. Πιο συγκεκριμένα οι μουσικοί δύσκολα αναθέτουν το κούρδισμα του οργάνου σε άλλος διότι προτιμούν την διαδικασία κουρδίσματος να την εκτελούν οι ίδιοι. Επιπλέον, μία βελτίωση που θα μπορούσε να γίνει είναι η χρήση ενός LED το οποίο θα προειδοποιεί τον χρήστη ότι ξεκινάει καινούργια επανάληψη και ότι θα πρέπει να διεγείρει την αντίστοιχη χορδή. Μια άλλη βελτίωση αφορά τον όγκο ηλεκτρικών PCA ώστε να μπορεί το σύστημα να αποθηκευτεί σε μία συμβατή θήκη. Ακόμα, το κουρδιστήρι θα μπορούσε να τοποθετηθεί στην κόψη της κεφαλής της κιθάρας χωρίς να προεξέχει. Επίσης, τα εξαρτήματα θα μπορούσε να μπαίνουν σε μία εξωτερική θήκη ώστε να προστατεύονται από εξωτερικούς παράγοντες. Θα μπορούσε ωστόσο να βελτιωθεί η αισθητική του συστήματος ώστε να είναι πιο ελκυστικό στην αγορά. Τέλος, μια σημαντική και έξυπνη βελτίωση θα ήταν η σχεδίαση του συστήματος ώστε να κουρδίζει και τις έξι χορδές της κιθάρας την μία μετά την άλλη χωρίς όμως να απαιτείται αποσυναρμολόγηση και ξανά συναρμολόγηση μετά από το κούρδισμα της κάθε χορδής. Αυτό θα είχε ως αποτέλεσμα την εξοικονόμηση του χρόνου κουρδίσματος αλλά η κατασκευή θα είναι πιο ογκώδης εξαιτίας των έξι κινητήρων [81].

Εικόνα 1.81: Το σύστημα Automatic Guitar Tuner [128]

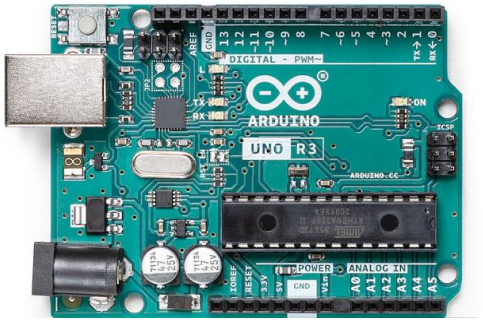
1.3 Η πλατφόρμα ανάπτυξης και το προγραμματιστικό περιβάλλον του Arduino

Στην συγκεκριμένη υποενότητα γίνεται μία αναφορά τόσο στις δυνατότητες του Arduino όσο και στο περιβάλλον προγραμματισμού του. Ακόμα, το Arduino μπορεί να ελέγξει μία αρκετά μεγάλη γκάμα ηλεκτρονικών εξαρτημάτων κάνοντάς το ευέλικτο και εύχρηστο για το στήσιμο διαφόρων εφαρμογών. Επίσης, η συγκεκριμένη διπλωματική εργασία υλοποιήθηκε με Arduino διότι είναι χαμηλό σε κόστος και εύκολο ως προς τον προγραμματισμό του.

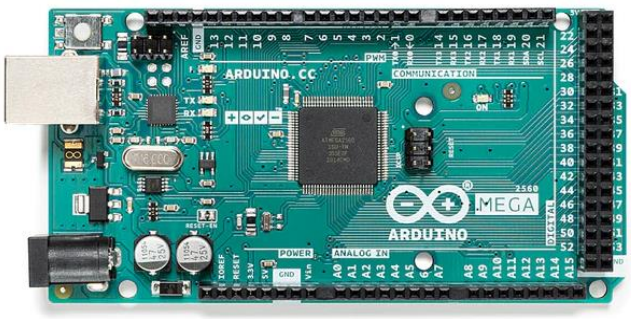
1.3.1 Γενικά

Αρχικά, το Arduino είναι μια πλατφόρμα ανάπτυξης εφαρμογών ανοιχτού κώδικα. Πιο συγκεκριμένα, περιλαμβάνει μια απλή πλακέτα η οποία περιλαμβάνει κυρίως μικροελεγκτές της Atmel AVR. Ωστόσο, οι εκδόσεις των μικροελεγκτών διαφέρουν μεταξύ τους δίνοντας περισσότερες ή λιγότερες δυνατότητες. Ένα από τα βασικότερα πλεονεκτήματα του Arduino είναι το χαμηλό κόστος το οποίο είναι πολύ σημαντικός παράγοντας κατά την υλοποίηση εφαρμογών. Μερικά ακόμα πλεονεκτήματα του Arduino είναι, η συμβατότητά του λογισμικού ανάπτυξης με τα διάφορα λειτουργικά συστήματα, η απλότητα του περιβάλλοντος ανάπτυξης η οποία επωφελή ακόμα και τους αρχάριους χρήστες καθώς και το επεκτάσιμο λογισμικό ανοιχτού κώδικα το οποίο κάποιος μπορεί να το τροποποιήσει σύμφωνα με τις ανάγκες του συστήματός του [82].

Εν συνεχεία, η πιο γνωστή έκδοση του Arduino είναι το Arduino UNO (Εικόνα 1.82) το οποίο περιλαμβάνει τον μικροελεγκτή της Atmel ATmega328 [83]. Μία άλλη έκδοση είναι το Arduino Mega 2560 (Εικόνα 1.83) το οποίο περιλαμβάνει τον μικροελεγκτή ATmega2560 [84].

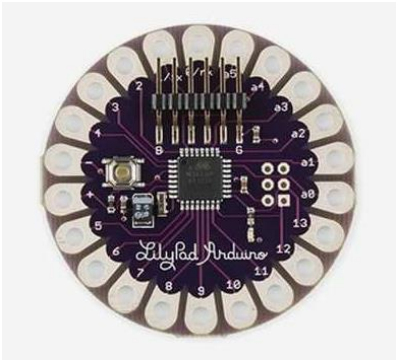


Εικόνα 1.82: Arduino UNO [129].

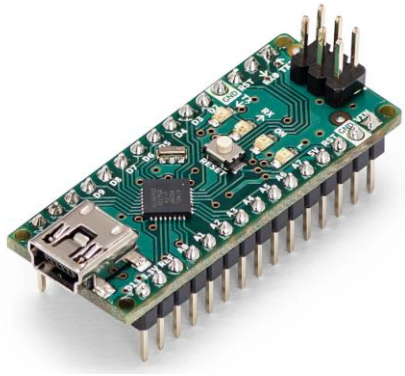


Εικόνα 1.83: Arduino Mega 2560 [130].

Επίσης μια πολύ ενδιαφέρουσα έκδοση του Arduino είναι το Arduino Lilypad (Εικόνα 1.84), το οποίο περιλαμβάνει τον μικροελεγκτή ATmega32u4 και είναι ιδανικό για ειδικές εφαρμογές [82] [85]. Επιπλέον, μία εξίσου γνωστή έκδοση είναι το Arduino Nano (Εικόνα 1.85) το οποίο περιλαμβάνει τον μικροελεγκτή της Atmel ATmega328 και είναι ιδανικό για την υλοποίηση ενσωματωμένων και φορητών συστημάτων [82] [86].

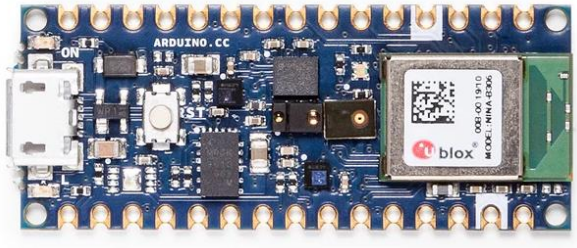


Εικόνα 1.84: Arduino Lilypad [131].

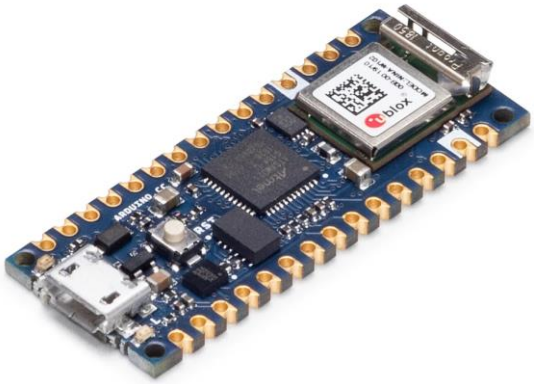


Εικόνα 1.85: Arduino Nano [132].

Ακόμα, μια πιο βελτιωμένη έκδοση του Arduino Nano αποτελεί το Arduino Nano 33BLE SENSE (Εικόνα 1.86), το οποίο περιλαμβάνει τον μικροελεγκτή nRF52840 καθώς και ένα πλήθος αισθητήρων κάνοντάς τον κατάλληλο για διάφορες ενσωματωμένες εφαρμογές [87]. Μία εξίσου εντυπωσιακή έκδοση είναι το Arduino Nano 33 IOT (Εικόνα 1.87) το οποίο περιλαμβάνει τον μικροελεγκτή SAMD21 Cortex®-M0+ 32bit low power ARM MCU και είναι ιδανικό για ανάπτυξη εφαρμογών του διαδικτύου των πραγμάτων μιας και παρέχει την δυνατότητα σύνδεσης με ειδικές υπηρεσίες νέφους (Cloud) [88] . Επίσης, έχει ενσωματωμένους αισθητήρες και υποστηρίζει επικοινωνία μέσω του πρωτοκόλλου Bluetooth [82].

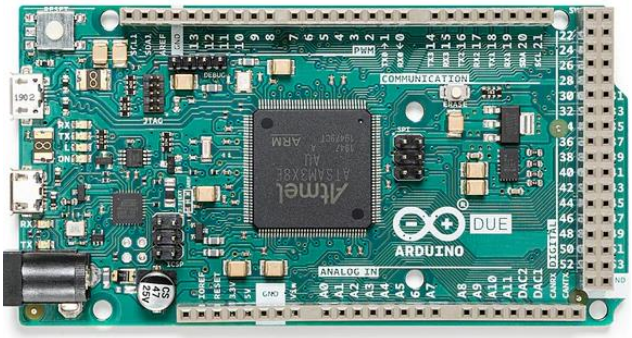


Εικόνα 1.86: Arduino Nano 33BLE SENSE [133].

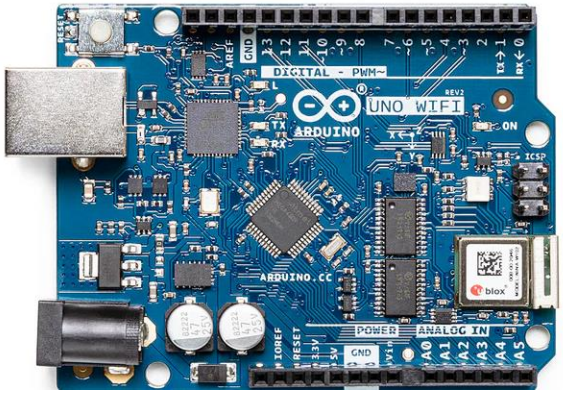


Εικόνα 1.87: Arduino Nano 33IoT [134].

Επιπλέον, μια ενδιαφέρουσα έκδοση αποτελεί το Arduino Due (Εικόνα 1.88) το οποίο περιλαμβάνει τον μικροελεγκτή AT91SAM3X8E και μπορεί να χρησιμοποιηθεί σε ειδικές εφαρμογές όπου απαιτείται σημαντική υπολογιστική ισχύς και υψηλή συχνότητα λειτουργίας [89]. Επιπροσθέτως, μία βελτιωμένη έκδοση αποτελεί το Arduino UNO WiFi REV2 (Εικόνα 1.89) το οποίο περιλαμβάνει τον μικροελεγκτή ATmega4809, αισθητήρα αδρανείας και συνδεσιμότητα WiFi κάνοντάς τον κατάλληλο για εφαρμογές IoT [82] [90].



Εικόνα 1.88: Arduino Due [135].



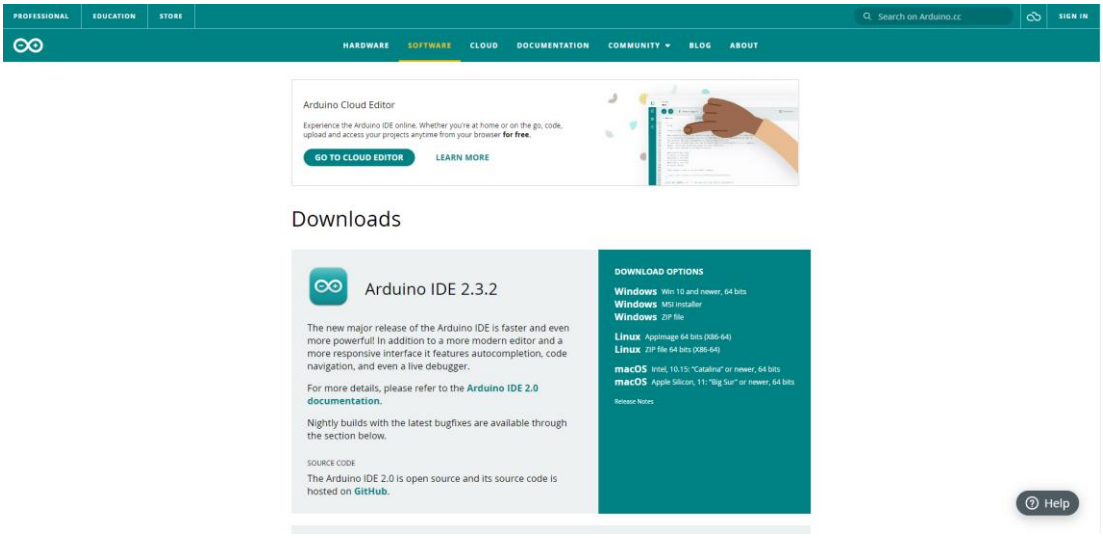
Εικόνα 1.89: Arduino UNO WiFi REV2 [136].

1.3.2 Το προγραμματιστικό περιβάλλον του Arduino

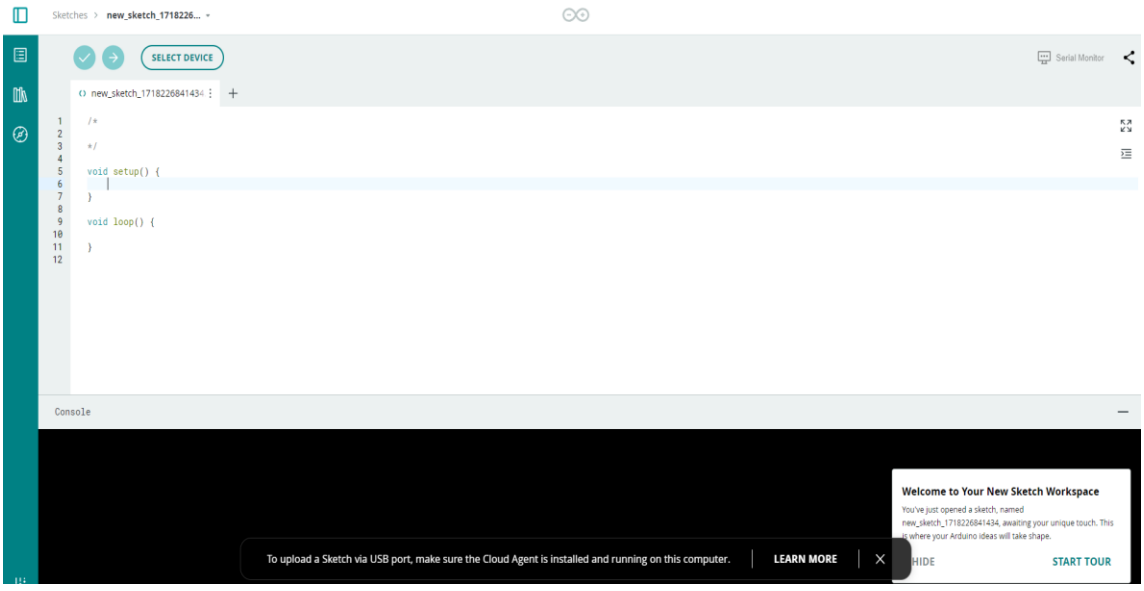
Για τον προγραμματισμό των πλακετών Arduino, χρησιμοποιείται το ολοκληρωμένο περιβάλλον ανάπτυξης Arduino IDE. Η εφαρμογή αυτή δίνει στον χρήστη την δυνατότητα να αναπτύξει προγραμματιστικά τις δικές του εφαρμογές και να τις αποθηκεύσει στον υπολογιστή του καθώς και να τις φορτώσει σε οποιαδήποτε πλακέτα

Arduino. Η εφαρμογή Arduino IDE βρίσκεται στην επίσημη ιστοσελίδα (Εικόνα 1.90) του Arduino και διατίθεται για λειτουργικά συστήματα υπολογιστών Windows, Linux & macOS. Η πιο πρόσφατη έκδοση που μπορεί να εγκατασταθεί μέχρι στιγμής είναι η 2.3.2 [82] [91].

Ένας άλλος τρόπος με τον οποίο θα μπορούσε να γίνει ανάπτυξη του κώδικα για το Arduino είναι μέσω της διαδικτυακής εφαρμογής Arduino Web Editor (Εικόνα 1.91). Ουσιαστικά, είναι παρόμοια εφαρμογή με το Arduino IDE με την μόνη διαφορά ότι απαιτείται μόνιμη σύνδεση στο διαδίκτυο μιας και για την χρήση του χρησιμοποιείται κάποιος Browser. Η ιστοσελίδα του Arduino Web Editor είναι στην ιστόσελίδα του Arduino και απαιτείται δημιουργία λογαριασμού ώστε οι κώδικες που αναπτύσσονται να αποθηκεύονται στο Cloud [92]. Όμως στην περίπτωση που χρειαστεί να φορτωθεί κάποιος από τους κώδικες σε κάποια πλακέτα Arduino, θα πρέπει να έχει γίνει πρώτα εγκατάσταση του Arduino Create Agent το οποίο είναι ένα ειδικό λογισμικό που παρέχει Drivers για την διασφάλιση της ορθής επικοινωνίας μεταξύ του υπολογιστή και της πλακέτας Arduino. Το λογισμικό του Arduino Create Agent υπάρχει σε εκδόσεις αντίστοιχες με το λειτουργικό σύστημα του υπολογιστή και βρίσκεται στην ιστοσελίδα υποστήριξης του Arduino [82][93].

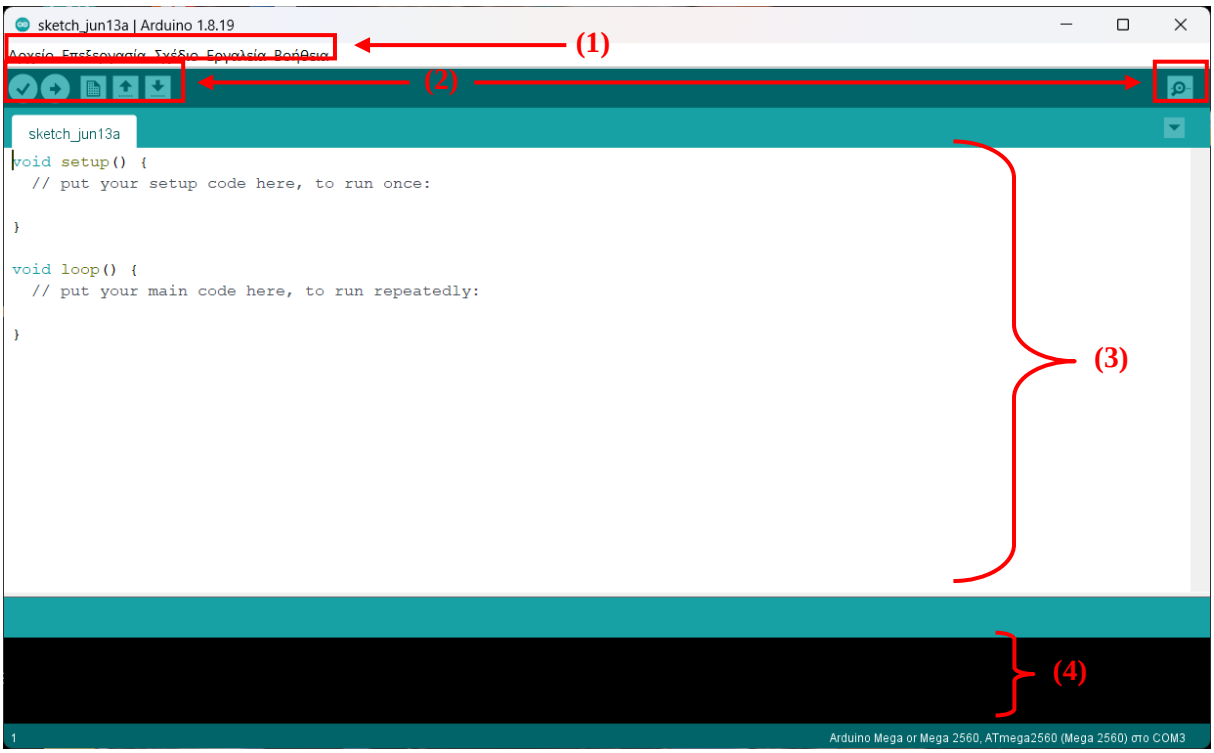


Εικόνα 1.90: Η επίσημη ιστοσελίδα του Arduino για την εγκατάσταση του Arduino IDE [137].



Εικόνα 1.91: Η διαδικτυακή εφαρμογή ανάπτυξης κώδικα Arduino Web Editor [138].

Για την υλοποίηση της συγκεκριμένης διπλωματικής εργασίας, έγινε χρήση του ολοκληρωμένου περιβάλλοντος ανάπτυξης Arduino IDE. Αρχικά, το περιβάλλον του Arduino IDE χωρίζεται σε τέσσερα επιμέρους τμήματα (Εικόνα 1.92). Πιο συγκεκριμένα ξεκινώντας από πάνω προς τα κάτω, υπάρχει η γραμμή μενού (1) η οποία παρέχει πρόσβαση σε λειτουργίες και εργαλεία για το στήσιμο του κώδικα. Στην συνέχεια, υπάρχει μία σειρά με ψηφιακά κουμπιά (2) τα οποία παρέχουν πρόσβαση στις πιο συχνές λειτουργίες. Έπειτα, υπάρχει ο συντάκτης κειμένου (3) μέσα στον οποίο γράφονται οι εντολές του κώδικα. Το τελευταίο τμήμα είναι το μαύρο πλαίσιο (4) στο κάτω μέρος της εφαρμογής στο οποίο εμφανίζονται μηνύματα σχετικά με τον κώδικα [82].



Εικόνα 1.92: Τα τέσσερα βασικά τμήματα του περιβάλλοντος ανάπτυξης Arduino IDE [139].

Ας δούμε τώρα με περισσότερη λεπτομέρεια τα τέσσερα αυτά βασικά τμήματα του Arduno IDE. Πρώτα, η γραμμή μενού (1), περιλαμβάνει το μενού Αρχείο το οποίο παρέχει κάποιες τυπικές λειτουργίες αρχείων, όπως δημιουργία νέου αρχείου, άνοιγμα αρχείου, αποθήκευση αρχείου κτλ. Επιπλέον, στο μενού Αρχείο υπάρχει το υπό-μενού παραδείγματα, το οποίο περιλαμβάνει μικρά κομμάτια κώδικα ως παραδείγματα και το υπό-μενού Προτιμήσεις στο οποίο μπορούν να διαμορφωθούν αρκετές παράμετροι όσον αφορά την εμφάνιση και την λειτουργία του περιβάλλοντος ανάπτυξης. Έπειτα, υπάρχει το μενού Επεξεργασίας το οποίο παρέχει λειτουργίες σχετικά με την σύνταξη του κώδικα, όπως αναίρεση, επανάληψη, αποκοπή, αντιγραφή κτλ. Εν συνεχεία, υπάρχει το μενού Σχέδιο, το οποίο περιλαμβάνει μερικές από τις βασικότερες λειτουργίες του περιβάλλοντος ανάπτυξης Arduino IDE. Πιο συγκεκριμένα, περιλαμβάνει τις λειτουργίες επικύρωση και μεταγλώττιση του κώδικα, όπου πραγματοποιείτε συντακτικός έλεγχος και δημιουργείτε το εκτελέσιμο αρχείο και την λειτουργία φόρτωση, όπου πραγματοποιείτε μεταγλώττιση και έπειτα το εκτελέσιμο αρχείο φορτώνεται στην πλακέτα Arduino. Επίσης περιλαμβάνει το υπό-μενού Συμπερίληψη Βιβλιοθηκών, με το οποίο γίνεται εισαγωγή

βιβλιοθηκών στο πρόγραμμα με σκοπό την υλοποίηση σύνθετων λειτουργιών στον κώδικα. Επιπλέον, περιλαμβάνει το μενού Εργαλεία μέσω του οποίο μπορεί να γίνει διαχείριση των βιβλιοθηκών, παρακολούθηση των δεδομένων μέσω της σειριακής θύρας και παρακολούθηση της γραφικής αναπαράστασης των δεδομένων που λαμβάνονται μέσω της σειριακής θύρας επιλέγοντας τον σειριακό σχεδιογράφο. Το τελευταίο μενού στην γραμμή μενού είναι αυτό της Βοήθειας, το οποίο περιλαμβάνει πληροφορίες σχετικά με το περιβάλλον ανάπτυξης Arduino IDE, όπως για παράδειγμα τεκμηρίωση της γλώσσας Wiring C, οδηγό για σύνδεση πλακέτας και φόρτωση προγράμματος, αντιμετώπιση προβλημάτων κτλ [82].

Όσον αφορά την σειρά με τα ψηφιακά κουμπιά (2) κάτω από την γραμμή μενού, ουσιαστικά αποτελούν συντομεύσεις για τις πιο συνηθισμένες επιλογές του μενού. Πιο συγκεκριμένα, το πρώτο κουμπί από αριστερά κάνει επικύρωση του κώδικα, το δεύτερο κουμπί κάνει μεταγλώττιση και ανέβασμα του κώδικα στην πλακέτα Arduino, το τρίτο κουμπί δημιουργεί καινούργιο αρχείο για την ανάπτυξη κώδικα, ενώ το τέταρτο και το τελευταίο κουμπί αποτελούν το άνοιγμα και την αποθήκευση αρχείου αντίστοιχα. Το τελευταίο ψηφιακό κουμπί τέρμα δεξιά, ανοίγει ένα παράθυρο για την παρακολούθηση της σειριακής θύρας [82].

Στην συνέχεια, στον συντάκτη κειμένου (3) όπως ειπώθηκε προηγουμένως, γράφονται οι εντολές του προγράμματος. Πιο συγκεκριμένα, περιλαμβάνει δύο συναρτήσεις, μία void setup() και μία void loop(). Η συνάρτηση void setup(), τρέχει μία φορά όταν ενεργοποιείται η πλακέτα Arduino και εντός αυτής καθορίζονται οι θύρες εισόδου και εξόδου καθώς και κάποιες εντολές ώστε το σύστημα να βρεθεί σε κάποια αρχική κατάσταση πριν την έναρξη του κυρίως προγράμματος. Εν συνεχεία, η συνάρτηση void loop είναι ουσιαστικά ένας ατέρμον βρόγχος ο οποίος τρέχει ξανά κάθε φορά που ολοκληρώνεται και εντός αυτού του βρόγχου γράφονται οι εντολές του κυρίως κώδικα. Επιπλέον, κάποιες εντολές μπορούν να γραφτούν και εκτός των δύο συναρτήσεων void setup() και void loop(). Πιο συγκεκριμένα, πριν από την συνάρτηση voit setup(), μπορεί να γίνει εισαγωγή βιβλιοθηκών καθώς ορισμός και αρχικοποίηση μεταβλητών και σταθερών. Ενώ, κάτω από την συνάρτηση void loop(), μπορούν να γραφτούν διάφορες συναρτήσεις οι οποίες μπορούν να κληθούν από το κυρίως πρόγραμμα. Η χρήση συναρτήσεων, συμβάλει στην ανάπτυξη ενός ευανάγνωστου και ομοιόμορφου κώδικα [82].

Το τελευταίο τμήμα του περιβάλλοντος ανάπτυξης Arduino IDE είναι το μαύρο πλαίσιο (4) στο κάτω μέρος στο οποίο εμφανίζονται μηνύματα σφάλματος και προειδοποιήσεις. Τα μηνύματα αυτά εμφανίζονται όταν ο Compiler εντοπίζει κάποιο πρόβλημα κατά την μεταγλώττιση του κώδικα ή κατά το ανέβασμα του κώδικα στην πλακέτα Arduino [82].

1.4 Τρισδιάστατη εκτύπωση (3D Printing)

Επιπροσθέτως, για την υλοποίηση της συγκεκριμένη διπλωματικής εργασίας τυπώθηκαν κάποια κομμάτια με την χρήση τρισδιάστατου εκτυπωτή ή αλλιώς 3D Printer. Σε αυτήν την υποενότητα αναλύονται τα βασικά χαρακτηριστικά και ο τρόπος λειτουργίας των τρισδιάστατων εκτυπωτών, καθώς και οι κατηγορίες των εκτυπωτών. Επιπλέον, γίνεται αναφορά γύρω από τον τύπο των υλικών που μπορεί να χρησιμοποιήσει ο κάθε τρισδιάστατος εκτυπωτής. Επίσης, αναφέρονται και επεξηγούνται τα δύο γραφικά

περιβάλλοντα σχεδίασης που χρησιμοποιήθηκαν για την υλοποίηση της συγκεκριμένης διπλωματικής εργασίας.

1.4.1 Χαρακτηριστικά & κατηγορίες των τρισδιάστατων εκτυπωτών (3D Printers)

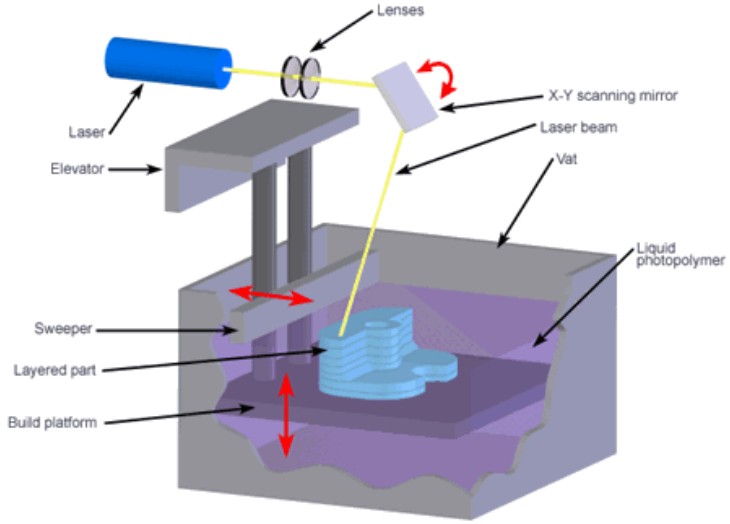
Αρχικά, στην τρισδιάστατη εκτύπωση χρησιμοποιούνται πολλές τεχνολογίες για την κατασκευή των επιπέδων των κομματιών. Η κάθε μία τεχνολογία διαφέρει από την άλλη ως προς την επιλογή υλικού, το κόστος, την ταχύτητα κατασκευής, το φινίρισμα της επιφάνεια και την ανθεκτικότητα [94]. Οι κατηγορίες της τρισδιάστατης εκτύπωσης είναι οι εξής:

- Στερεολιθογραφία (Stereolithography)
- Επιλεκτική συσσώρευση λέιζερ (Selective Laser Sintering)
- Μοντελοποίηση με συντηγμένη εναπόθεση (Fused Deposition Modeling)
- Επεξεργασία ψηφιακού φωτός (Digital Light Process)
- Πολλαπλός Τζετ Σύντηξης (Multi Jet Fusion)
- PolyJet
- Άμεση μεταλλική πυροσυσσωμάτωση με λέιζερ (Direct Metal Laser Sintering)
- Τήξη με Δέσμη Ηλεκτρονίων (Electron Beam Melting)

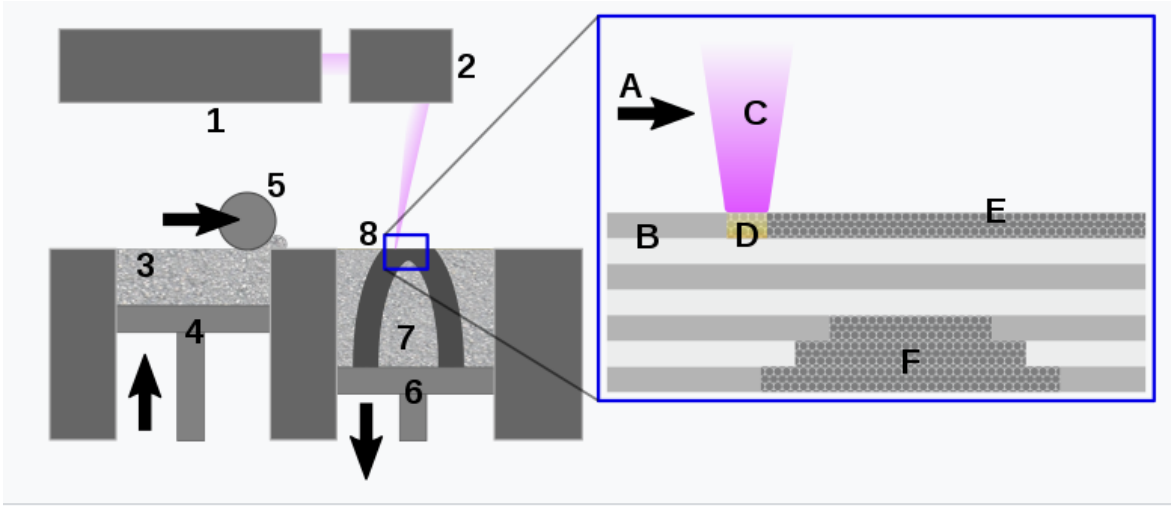
Η στερεολιθογραφία ή αλλιώς Stereolithography (Εικόνα 1.93) αποτελεί την πρώτη καινοτομία τρισδιάστατης εκτύπωσης και χρησιμοποιεί την τεχνική του πολυμερισμού του φωτός (Vat polymerization). Το υλικό που χρησιμοποιεί είναι μια ρητίνη φωτο-πολυμερούς η οποία αποκαθίσταστε από μία πηγή φωτός. Ακόμα, οι εκτυπωτές αυτοί παρέχουν αυξημένο επίπεδο λεπτομέρειας, ομαλό φινίρισμα επιφανειών και σφιχτές αντιστάσεις. Επιπλέον, στην στερεολιθογραφία γίνεται χρήση γαλβανομετρικών καθρεπτών οι οποίοι κατευθύνουν την υπεριώδης δέσμη λέιζερ στην δεξαμενή με την ρητίνη η οποία είναι φωτοχημικά στερεοποιημένη με αποτέλεσμα να σχηματίζεται ένα μόνο στρώμα. Έπειτα, η πλατφόρμα εντός της δεξαμενής κατεβαίνει ένα επίπεδο και μία λεπίδα με μεγάλη επιφάνεια στρώνει ξανά την ρητίνη πάνω από το πρώτο επίπεδο. Η διαδικασία αυτή επαναλαμβάνεται για κάθε ένα επίπεδο. Μόλις ολοκληρωθεί το κομμάτι τότε καθαρίζεται με ένα διαλυτικό μέσω για να φύγει η περιττή ρητίνη από την επιφάνεια του. Επίσης, αν η δεξαμενή έχει διάφανο πυθμένα μπορεί να εστιαστεί η υπεριώδης δέσμη προς τα πάνω, τότε η εκτύπωση του κομματιού μπορεί να γίνει από κάτω προς τα πάνω. Ουσιαστικά, η πλατφόρμα ακουμπά στον πυθμένα της δεξαμενής και το στρώμα αντικειμένου προσκολλάτε πάνω στην πλατφόρμα η οποία σιγά-σιγά ανεβαίνει προς τα πάνω. Έπειτα το φωτο-πολυμερές υγρό ρέει από της άκρες του προηγούμενου στρώματος γεμίζοντας το κενό χώρο ώστε να δημιουργηθεί το επόμενο στρώμα. Αυτή η διαδικασία επαναλαμβάνεται και για τα υπόλοιπα στρώματα. Με αυτόν τον τρόπο, μπορούν να δημιουργηθούν κατασκευές οι οποίες έχουν μεγαλύτερο ύψος από την ίδια την δεξαμενή. Κατά την δημιουργία των μοντέλων CAD, πέρα από τις συντεταγμένες X, Y & Z δημιουργούνται και στηρίγματα τα οποία αφαιρούνται με το χέρι μόλις ολοκληρωθεί η κατασκευή. Τα στηρίγματα αποσκοπούν στην αποφυγή της παραμόρφωσης της

κατασκευής εξαιτίας της βαρύτητας, της πίεσης από την πλευρική λεπίδα και από την κατακόρυφη ταλάντωσης της πλατφόρμας. Η κατηγορία αυτή χρησιμοποιείται κυρίως για την κατασκευή ιατρικών μοντέλων και πιο συγκεκριμένα ανατομικά τμήματα ασθενών βασισμένα σε δεδομένα που προκύπτουν από σαρώσεις υπολογιστών [94] [95] [96].

Ένας άλλος τύπος τρισδιάστατης εκτύπωσης είναι η επιλεκτική συσσώρευση λέιζερ ή αλλιώς Selective Laser Sintering (SLS) (Εικόνα 1.94). Πιο συγκεκριμένα, αυτή η κατηγορία χρησιμοποιεί ένα ισχυρό παλμικό λέιζερ για την σύντηξη μικρών σωματιδίων πλαστικού, μετάλλου, κεραμικού ή γυαλιού που βρίσκονται υπό μορφή σκόνης. Το λέιζερ σαρώνει επιλεκτικά διατομές σύμφωνα με το αρχείο CAD πάνω στην επιφάνεια της σκόνης. Μόλις σχεδιαστεί το πρώτο στρώμα, τότε η πλατφόρμα με το στρώμα σκόνης κατεβαίνει κατά ένα στρώμα και ένα νέο στρώμα σκόνης εφαρμόζεται από πάνω. Αυτή η διαδικασία επαναλαμβάνεται και για τα επόμενα στρώματα μέχρι την ολοκλήρωση της κατασκευής. Επίσης, η συγκεκριμένη κατηγορία τρισδιάστατης εκτύπωσης δεν απαιτεί στηρίξεις τμημάτων στις διάφορες κατασκευές, διότι όλη η κατασκευή περιβάλλεται από σκόνη που κρατά σταθερά αυτά τα τμήματα. Ως επί το πλείστων, αυτή η κατηγορία τρισδιάστατης εκτύπωσης χρησιμοποιείται κυρίως για την κατασκευή πρωτότυπων εξαρτημάτων και για παραγωγή ανταλλακτικών τελικής χρήσης σε αεροδιαστημικές, στρατιωτικές, ιατρικές, φαρμακευτικές και ηλεκτρονικές εφαρμογές [94] [97].



Εικόνα 1.93: Τρισδιάστατη εκτύπωση με στερεολιθογραφία (stereolithography) [140].



Selective laser sintering process
1 Laser **2** Scanner system **3** Powder delivery system **4** Powder delivery piston **5** Roller **6** Fabrication piston **7** Fabrication powder bed **8** Object being fabricated (see inset) **A** Laser scanning direction **B** Sintered powder particles (brown state) **C** Laser beam **D** Laser sintering **E** Pre-placed powder bed (green state) **F** Unsintered material in previous layers

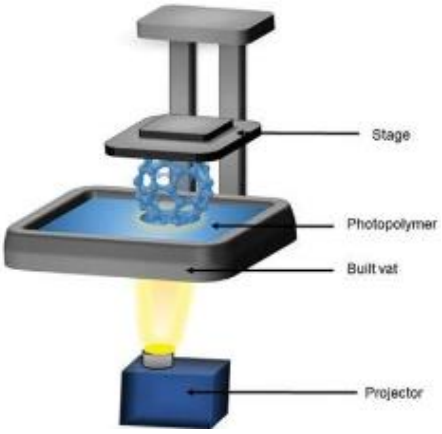
Εικόνα 1.94: Τρισδιάστατη εκτύπωση με επιλεκτική συσσώρευση λέιζερ (Selective Laser Sintering) [141].

Μια άλλη κατηγορία τρισδιάστατης εκτύπωση, η οποία χρησιμοποιήθηκε για την υλοποίηση της συγκεκριμένης εργασίας, είναι η μοντελοποίηση με συντηγμένη εναπόθεση (Fused Deposition Modeling) (FDM) ή αλλιώς κατασκευή με συντηγμένη ίνα (Fused Filament Fabrication) (FFF) (Εικόνα 1.95) [101] [102]. Αυτή η κατηγορία περιλαμβάνει του κλασσικούς τρισδιάστατους εκτυπωτές που μπορούν να χρησιμοποιηθούν ακόμα και στο σπίτι καθώς και τους πιο εξειδικευμένους εκτυπωτές που χρησιμοποιούνται για βιομηχανικές εφαρμογές. Πιο συγκεκριμένα, οι εκτυπωτές αυτοί χρησιμοποιούν την εξώθηση θερμοπλαστικού πλαστικού υλικού υπό την μορφή ίνας για την κατασκευή των μοντέλων. Το θερμοπλαστικό υλικό είναι τυλιγμένο σε καρούλι υπό την μορφή νήματος και μέσω ενός θερμαινόμενου στομίου που ονομάζεται εξωθητής (extruder) υγροποιείτε και βγαίνει υπό την μορφή ίνας πάνω σε μία θερμαινόμενη πλατφόρμα. Ο εξωθητής (extruder) κινείτε ως προς τους άξονες X & Y ώστε να διαμορφώσει το πρώτο στρώμα και όταν το ολοκληρώσει κινείται ως προς τον άξονα Z για την διαμόρφωση του επόμενου στρώματος. Ωστόσο, κάποιοι εκτυπωτές μετακινούν την κεφαλή εξώθησης (extruder) προς τα επάνω, ενώ κάποιοι άλλοι μετακινούν την πλατφόρμα προς τα κάτω για την διαμόρφωση του επόμενου στρώματος. Η διαδικασία αυτή επαναλαμβάνεται έως ότου ολοκληρωθεί η κατασκευή. Επίσης, στην κεφαλή εξώθησης τοποθετούνται ανεμιστήρες οι οποίοι έχουν σκοπό να κρατάνε σταθερή την θερμοκρασία εξώθησης καθώς και να ψύχουν το υλικό για να στερεοποιηθεί. Επίσης, σε αυτή την κατηγορία εκτυπωτών θα πρέπει να δημιουργούνται στηρίγματα ανάλογα με το μοντέλο που πρέπει να κατασκευαστεί ώστε να μην πέσει εξαιτίας της βαρύτητας. Ωστόσο αυτά τα στηρίγματα μπορούν να αποκολληθούν με το χέρι αφού ολοκληρωθεί η κατασκευή. Εν συνεχεία, τα θερμοπλαστικά υλικά που μπορούν να χρησιμοποιηθούν στους τρισδιάστατους εκτυπωτές οικιακής χρήσης είναι το PLA (Πολυγαλακτικό οξύ) και το ABS (Στυρόλιο βουταδιενίου ακρυλονιτριλίου). Πιο συγκεκριμένα, το PLA είναι ένας βιοδιασπώμενος πολυεστέρας ο οποίος προέρχεται από ανανεώσιμες πηγές φυτικής προέλευσης [99]. Από την άλλη το ABS είναι πιο ανθεκτικό και πιο σκληρό ως υλικό παρέχοντας στιβαρότητα, αλλά σε περίπτωση υπερθέρμανσής του το υλικό αυτό είναι καρκινογόνο πράγμα επικίνδυνο για την ανθρώπινη υγεία [100]. Τα υλικά που χρησιμοποιούνται στους τρισδιάστατους

εκτυπωτές βιομηχανικής χρήσης είναι κυρίως μηχανικά θερμοπλαστικά (PA, TPU & PETG), ABS, Polycarbonate και Ultem. Με αυτά τα υλικά επιτυγχάνεται υψηλή αντοχή κρούσης, θερμική σταθερότητα, χημική αντοχή και βιοσυμβατότητα τα οποία είναι απαραίτητα σε βιομηχανικές εφαρμογές [94] [98].

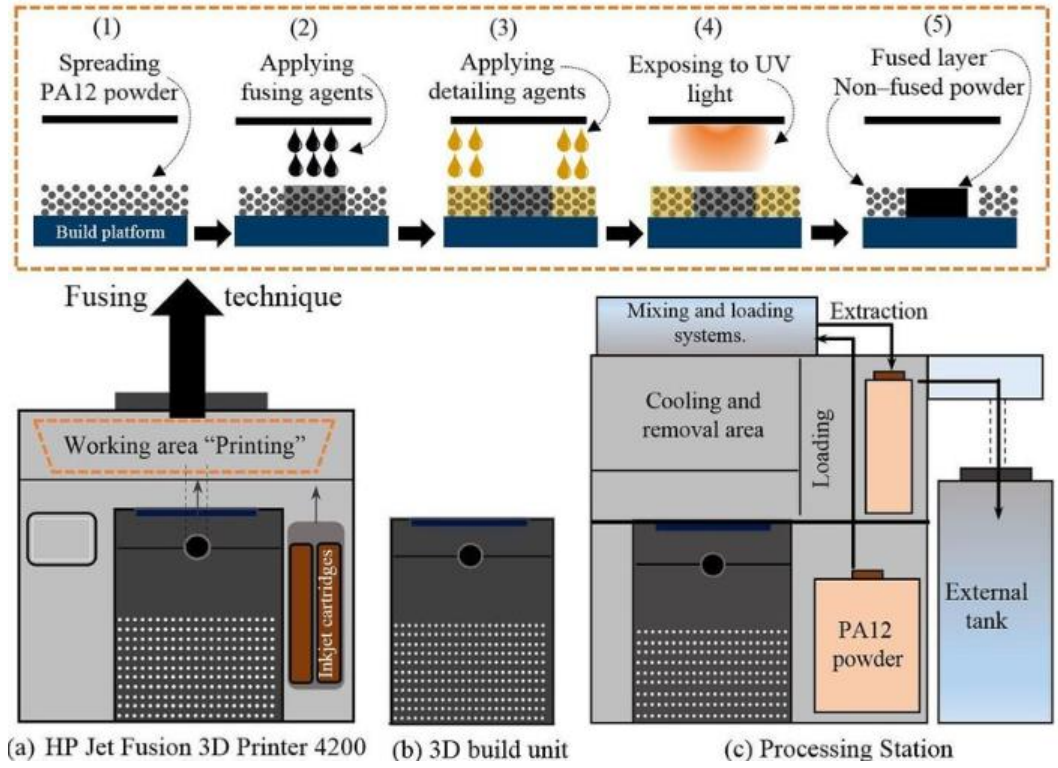
Εικόνα 1.95: Τρισδιάστατοι εκτυπωτές οικιακής χρήσης (αριστερά) και βιομηχανικής χρήσης (δεξιά) με μοντελοποίηση συντηγμένης εναπόθεσης (Fused Deposition Modeling) ή αλλιώς κατασκευή με συντηγμένη ίνα (Fused Filament Fabrication) [142].

Μία ακόμα κατηγορία τρισδιάστατης εκτύπωσης ονομάζεται επεξεργασία ψηφιακού φωτός ή αλλιώς Digital Light Process (DLP) (Εικόνα 1.96). Η κατηγορία αυτή είναι παρόμοια με την στερεολιθογραφία, μόνο που αντί για δέσμη λέιζερ χρησιμοποιεί μια προβαλλόμενη πηγή φωτός για την δημιουργία ολόκληρου του στρώματος ταυτόχρονα. Με αυτόν τον τρόπο, η εκτύπωση του μοντέλου γίνεται πιο γρήγορα εξαιτίας της κατασκευής ολόκληρου του στρώματος ταυτόχρονα. Η διαδικασία αυτή επαναλαμβάνεται έως ότου υλοποιηθούν όλα τα στρώματα του μοντέλου. Αυτή η κατηγορία τρισδιάστατης εκτύπωσης μπορεί να χρησιμοποιηθεί για την εκτύπωση περίπλοκων αντικειμένων με την χρήση φωτό-πολυμερής ρητίνης όπως παιχνίδια, καλούπια κοσμημάτων, οδοντιατρικά καλούπια κτλ, διότι. Ωστόσο, αυτή η κατηγορία τρισδιάστατης εκτύπωσης δεν έχει πολύ καλή ακρίβεια πράγμα που την κάνει ακατάλληλη για την κατασκευή μηχανικών μοντέλων. Επίσης, ένας τέτοιος τρισδιάστατο εκτυπωτής δεν είναι κατάλληλος για την κατασκευή πολύ μεγάλων μοντέλων μιας και το μέγεθος του αντικειμένου εξαρτάται από τον προβολέα και την ανάλυση της εικόνας που προβάλει [94] [103] [104].



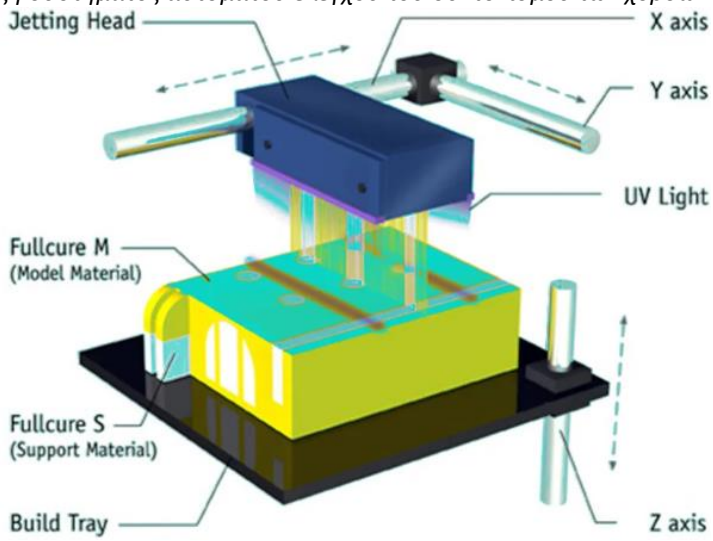
Εικόνα 1.96: Τρισδιάστατοι εκτυπωτές επεξεργασίας ψηφιακού φωτός (Digital Light Process) [143].

Μία άλλη ενδιαφέρουσα κατηγορία είναι η τρισδιάστατη εκτύπωση με πολλαπλό τζετ σύντηξης ή αλλιώς Multi Jet Fusion (MJF) (Εικόνα 1.97) [106]. Αυτή η κατηγορία τρισδιάστατης εκτύπωσης, κατασκευάζει ένα μοντέλο πολύ πιο γρήγορα από οποιαδήποτε άλλη κατηγορία με αποτέλεσμα το κόστος δημιουργίας να είναι χαμηλό. Αυτό επιτυγχάνεται με την γρήγορη και γραμμική εναπόθεση του υλικού. Επίσης, μπορούν να κατασκευαστούν πολλά αντικείμενα σε μία γραμμή παραγωγής χωρίς να επηρεάζεται η ταχύτητα κατασκευής. Πιο συγκεκριμένα, αυτή η κατηγορία εκτυπωτών χρησιμοποιεί ένα σύμπλεγμα inkjet για την εκτόξευση πολλών σταγονιδίων φωτο-πολυμερούς υλικού πάνω σε ένα προθερμασμένο λεπτό στρώμα σκόνης υλικού. Εξαιτίας της χρήσης σκόνης δεν απαιτείται η χρήση στηριγμάτων κατά την κατασκευή του τρισδιάστατου μοντέλου. Η στερεοποίηση ή αλλιώς ο πολυμερισμός του υλικού γίνεται με την χρήση υπεριώδους ακτινοβολίας. Μετά την κατασκευή του πρώτου στρώματος, η πλατφόρμα κατεβαίνει κατά μία στρώση ώστε να δημιουργηθεί χώρος για την επόμενη εναπόθεση υλικού. Η διαδικασία αυτή επαναλαμβάνεται έως ότου δημιουργηθεί το τρισδιάστατο αντικείμενο. Αφού ολοκληρωθεί η τρισδιάστατη κατασκευή, ο εκτυπωτής αυτός περιέχει έναν ξεχωριστό σταθμό επεξεργασίας στον οποίο γίνεται ψύξη, από-συσκευασία και ανάκτηση της περίσσιας σκόνης για μελλοντική χρήση. Επίσης, για την αφαίρεση οποιασδήποτε σκόνης από το τρισδιάστατο κομμάτι γίνεται χρήση της τεχνικής εκτόξευσης χαντρών, η με την εκτόξευση αέρος ή με την εκτόξευση νερού. Ωστόσο, η αφαίρεση της σκόνης μπορεί να γίνει με την χρήση υπερήχων ή με δονητική μηχανή φινιρίσματος. Επιπλέον, τα υλικά που χρησιμοποιούνται σε αυτή την κατηγορία τρισδιάστατης εκτύπωσης είναι τα άκαμπτα και τα εύκαμπτα πλαστικά. Πιο συγκεκριμένα, τα άκαμπτα πλαστικά που χρησιμοποιούνται είναι το Nylon PA11, το Nylon PA12 & το PP, ενώ στα εύκαμπτα πλαστικά χρησιμοποιείται κυρίως το Estane 3D TPU M95A. Ουσιαστικά, αυτή η κατηγορία τρισδιάστατης εκτύπωσης χρησιμοποιείται για την παραγωγή, πολύπλοκων εξαρτημάτων με μεγάλο όγκο, με λεπτομέρεια και με δομική ακεραιότητα. Μερικές δημοφιλής κατασκευές αποτελούν ηλεκτρονικά και μηχανικά εξαρτήματα, καθώς και περιβλήματα [105].



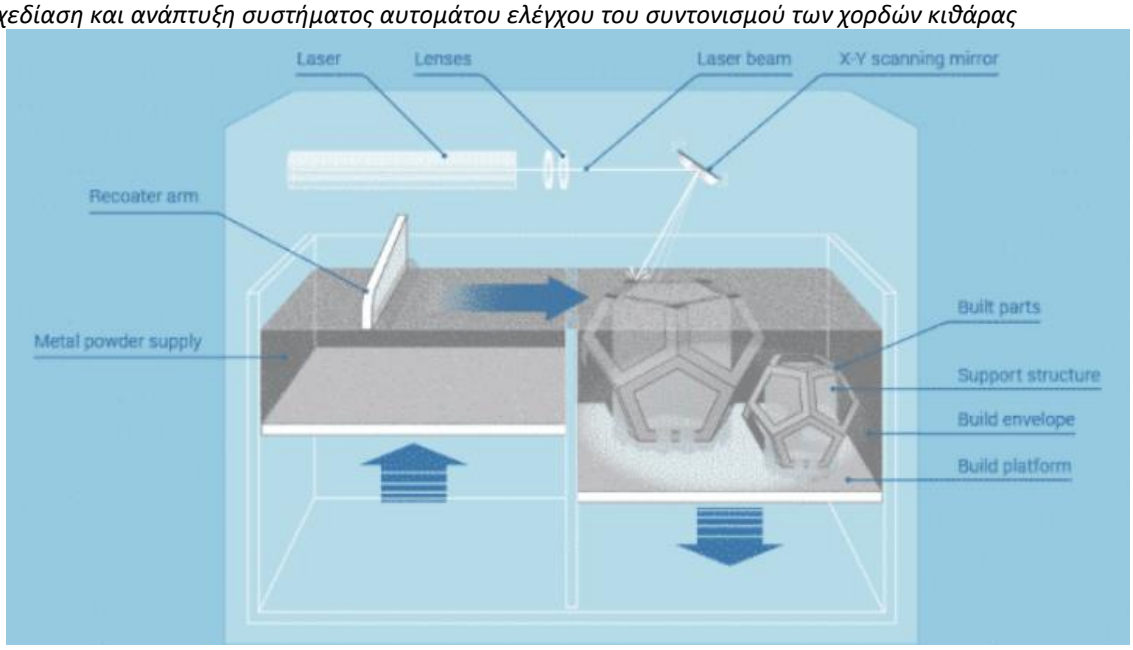
Εικόνα 1.97: Τρισδιάστατος εκτυπωτής (HP Jet Fusion 3D Printer 4200) με πολλαπλό τζετ σύντηξης (Multi Jet Fusion) [144].

Μία ιδιαίτερη κατηγορία είναι η τρισδιάστατη εκτύπωση με PolyJet (Εικόνα 1.98) η οποία δημιουργεί τρισδιάστατα μοντέλα γρήγορα και με μεγάλη λεπτομέρεια [108]. Αρχικά, στον 3D Polyjet εκτυπωτή, εισάγεται το CAD αρχείο το οποίο καθορίζει την σειρά εκτύπωσης, την τοποθέτηση του φωτοπολυμερούς υλικού και του υλικού στήριξης. Στην συνέχεια, στην πλάκα εκτύπωσης του εκτυπωτή εκτοξεύονται σταγονίδια από υγρό φωτοπολυμερές και με υπεριώδης ακτινοβολία γίνεται ο φωτοπολυμερισμός του. Επίσης, η συγκεκριμένη κατηγορία τρισδιάστατης εκτύπωσης μπορεί να υποστηρίξει την ανάμειξη διαφορετικών υλικών στο ίδιο στρώμα. Επιπλέον, σε περίπλοκα μοντέλα στα οποία απαιτείται στήριξη εκτοξεύεται ένα αφαιρούμενο υλικό το οποίο ονομάζεται FullCore 750. Μετά την ολοκλήρωση της εκτύπωσης, το μοντέλο καθαρίζεται με την χρήση νερού αλλά και με το χέρι. Οι τρισδιάστατοι εκτυπωτές αυτής της κατηγορίας έχουν πολύ καλή ακρίβεια με αποτέλεσμα να δημιουργούνται λείες επιφάνειες και πολύ λεπτά στρώματα όπου χρειάζεται. Οι συγκεκριμένοι τρισδιάστατη εκτυπωτές, μπορούν να χρησιμοποιήσουν ψηφιακά υλικά, τα οποία αποτελούνται από ένα μείγμα με έξι διαφορετικές ρητίνες, ψηφιακό πλαστικό ABS, εύκαμπτα υλικά, διάφανα υλικά, άκαμπτα και αδιαφανή υλικά, προσομοιωμένα υλικά πολυπροπυλενίου, υλικά υψηλής θερμοκρασίας και βίο-συμβατά υλικά. Ουσιαστικά, η συγκεκριμένη κατηγορία τρισδιάστατων εκτυπωτών μπορεί να χρησιμοποιηθεί για την κατασκευή πρωτότυπων αρχιτεκτονικών μοντέλων, ανατομικών μοντέλων και διαφόρων εξαρτημάτων [94] [107].



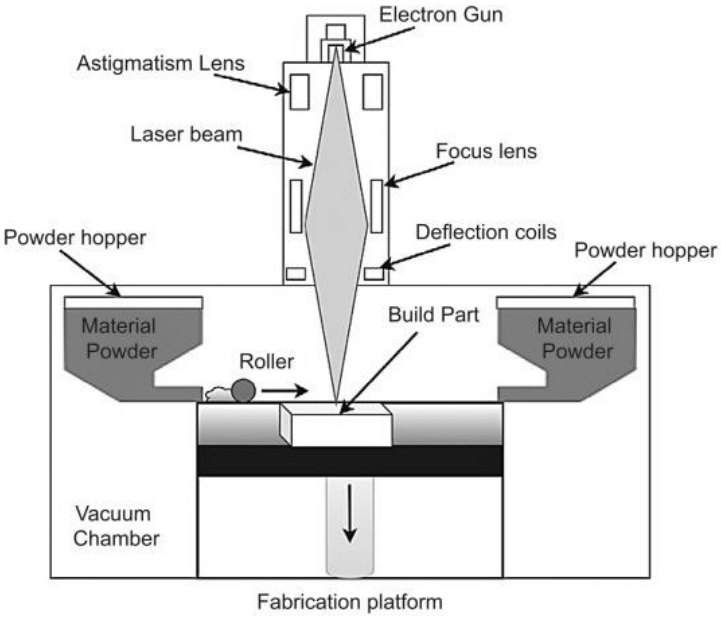
Εικόνα 1.98: Τρισδιάστατος εκτυπωτής με PolyJet [145].

Μια άλλη ενδιαφέρουσα κατηγορία τρισδιάστατης εκτύπωσης γίνεται με την τεχνική της άμεσης μεταλλικής πυροσυσσωμάτωσης με λέιζερ ή αλλιώς (Direct Metal Laser Sintering) (DMLS) (Εικόνα 1.99) [109]. Μια άλλη ονομασία που έχει δοθεί είναι επιλεκτική τήξη με λέιζερ ή αλλιώς selective laser melting. Πιο συγκεκριμένα, πάνω στην πλατφόρμα κατασκευής του τρισδιάστατου εκτυπωτή, απλώνεται ένα στρώμα πούδρας και στην συνέχεια ένα λέιζερ καθοδηγούμενο από ένα αρχείο CAD στοχεύει σε επιλεγμένα σημεία. Στα σημεία αυτά, γίνεται πυροσυσσωμάτωση του υλικού και έτσι δημιουργείτε το πρώτο στρώμα. Έπειτα, η πλατφόρμα κατασκευής κατεβαίνει κατά ένα στρώμα πάχους και το νέο στρώμα πούδρας απλώνεται ώστε να κατασκευαστεί το επόμενο στρώμα. Προφανώς, η ίδια διαδικασία επαναλαμβάνεται έως ότου ολοκληρωθεί το τρισδιάστατο μοντέλο. Ουσιαστικά, το λέιζερ θερμαίνει τα σωματίδια πούδρας μέχρι να φτάσουν στο σημείο τήξης και να γίνει η συσσωμάτωση. Με αυτόν τον τρόπο τα τρισδιάστατα μοντέλα έχουν εξαιρετική ακρίβεια και ποιότητα επιφάνειας. Τα είδη των μετάλλων που χρησιμοποιούνται στην συγκεκριμένη τεχνική τρισδιάστατης εκτύπωσης είναι ο ανοξείδωτος χάλυβας και το αλουμίνιο, ενώ τα χρησιμοποιούνται και κράματα νικελίου, τιτανίου, χαλκού βολφραμίου κτλ. Ωστόσο, αυτή η τεχνική χρησιμοποιείτε κυρίως στην βιομηχανία για την κατασκευή περιορισμένων μοντέλων με περίπλοκο σχεδιασμό και γεωμετρία [94] [110].



Εικόνα 1.99: Τρισδιάστατος εκτυπωτής με άμεση μεταλλική πυροσυσσωμάτωση με λέιζερ (Direct Metal Laser Sintering [146].

Τέλος, μια άλλη κατηγορία τρισδιάστατης εκτύπωσης γίνεται με την χρήση της τεχνικής της τήξης με Δέσμη Ηλεκτρονίων ή αλλιώς (Electron Beam Melting) (EBM) (Εικόνα 1.100) [112]. Πιο συγκεκριμένα, η τεχνική αυτή χρησιμοποιεί μια δέσμη ηλεκτρονίων, η οποία καθοδηγείται από ένα μαγνητικό πεδίο σύμφωνα με ένα αρχείο CAD. Αυτή η δέσμη ηλεκτρονίων έχει σκοπό την τήξη συγκεκριμένων σημείων του στρώματος που αποτελείται από σκόνη μετάλλου. Επίσης, η τεχνική αυτή εφαρμόζεται σε θάλαμο κενού για προστασία από οξειδώσεις ώστε να μην συμβεί κάποιο ατύχημα από τα υλικά υψηλής αντίδρασης. Μετά την ολοκλήρωση της εκτύπωσης του πρώτου στρώματος, η πλατφόρμα κατασκευής κατεβαίνει κατά ένα στρώμα πάχους και στην συνέχεια απλώνεται πάνω σε αυτό το επόμενο στρώμα σκόνης μετάλλου. Η διαδικασία αυτή επαναλαμβάνεται έως ότου ολοκληρωθεί το τρισδιάστατο μοντέλο. Τα υλικά που μπορούν να χρησιμοποιηθούν σε αυτού του είδους εκτυπωτών είναι μέταλλα τα οποία έχουν υψηλό σημείο τήξης. Οι τρισδιάστατοι εκτυπωτές αυτής της κατηγορίας χρησιμοποιούνται κυρίως για την κατασκευή εξαρτημάτων τόσο στην αυτοκινητοβιομηχανία και στην άμυνα, όσο και για ιατρικές, πετροχημικές και αεροδιαστημικές εφαρμογές. Με αυτήν την τεχνική, κατασκευάζονται εξαρτήματα υψηλής αντοχής τα οποία χρησιμοποιούν στο έπακρο τις ιδιότητες του μετάλλου και αποφεύγονται οι ακαθαρσίες που προκύπτουν στην τεχνική χύτευσης [94] [111].



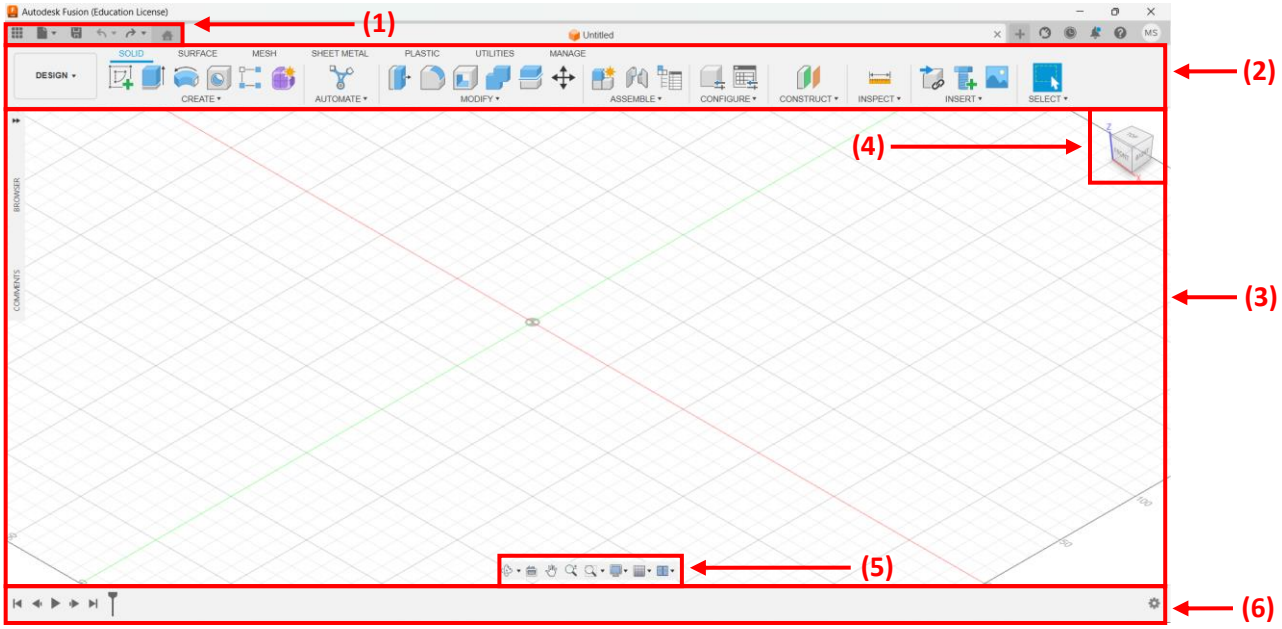
Εικόνα 1.100: Τρισδιάστατος εκτυπωτής ο οποίος χρησιμοποιεί την τεχνική τήξης με δέσμη ηλεκτρονίων (Electron Beam Melting)[147].

1.4.2 Εφαρμογές σχεδίασης με γραφικό περιβάλλον για τρισδιάστατη εκτύπωση

Στην συγκεκριμένη διπλωματική εργασία, για την δημιουργία των τρισδιάστατων μοντέλων χρησιμοποιήθηκε η τεχνική μοντελοποίησης με συντηγμένη εναπόθεση (Fused Deposition Modeling). Ωστόσο, για την σχεδίαση των τρισδιάστατων μοντέλων χρησιμοποιήθηκαν δύο γραφικά περιβάλλοντα. Το ένα είναι το Autodesk Fusion το οποίο χρησιμοποιήθηκε κυρίως για τον γεωμετρικό σχεδιασμό των τρισδιάστατων μοντέλων και το άλλο είναι το Ultimaker Cura με το οποίο καθορίζονται οι διάφορες λεπτομέρειες τόσο για το κάθε στρώμα όσο και για τις στηρίξεις του τρισδιάστατου μοντέλου.

Αρχικά, το γραφικό περιβάλλον της Autodesk Fusion χωρίζεται σε έξι βασικά μέρη (Εικόνα 1.101). Το πρώτο βασικό μέρος του γραφικού περιβάλλοντος είναι η γραμμή μενού (1) η οποία περιλαμβάνει το μενού Show Data Panel, στο οποίο περιλαμβάνονται τα αρχεία που έχουν δημιουργηθεί, το μενού File, στο οποίο περιλαμβάνονται η δημιουργία νέου αρχείου, η αποθήκευση, το άνοιγμα είδη υπάρχον αρχείου, την εξαγωγή (Export) και ρυθμίσεις προβολής. Επειδή, στην συγκεκριμένη εργασία το κάθε τρισδιάστατο μοντέλο επεξεργάστηκε περεταίρω με το γραφικό πρόγραμμα σχεδιασμού UltiMaker Cura, θα πρέπει να γίνει εξαγωγή (Export) του αρχείου σε μορφή STL. Επίσης, περιλαμβάνει το εικονίδιο της δισκέτας για αποθήκευση του αρχείου, την αναίρεση, την επανάληψη και το εικονίδιο Home το οποίο περιλαμβάνει την επιλογή για την δημιουργία νέου αρχείου, το άνοιγμα ήδη υπάρχον αρχείου καθώς προβάλει και τα αρχεία που έχουν δημιουργηθεί. Το δεύτερο βασικό μέρος είναι η εργαλειοθήκη (2), η οποία παρέχει τα απαραίτητα εργαλεία για οποιαδήποτε κατηγορία σχεδιασμού. Πιο συγκεκριμένα, περιλαμβάνει ένα παράθυρο το οποίο παρέχει την δυνατότητα επιλογής τύπου σχεδιασμού (Design, Generative design, Render, Animation, Simulation, Manufacture & Drawing). Επιπλέον, περιλαμβάνει ένα πλήθος εργαλείων ανάλογα με τις προτιμήσεις του σχεδιαστή. Τα εργαλεία τα οποία χρησιμοποιήθηκαν για την υλοποίηση της συγκεκριμένης διπλωματικής εργασίας βρίσκονται στην εργαλειοθήκη Solid. Ουσιαστικά, τα βασικότερα

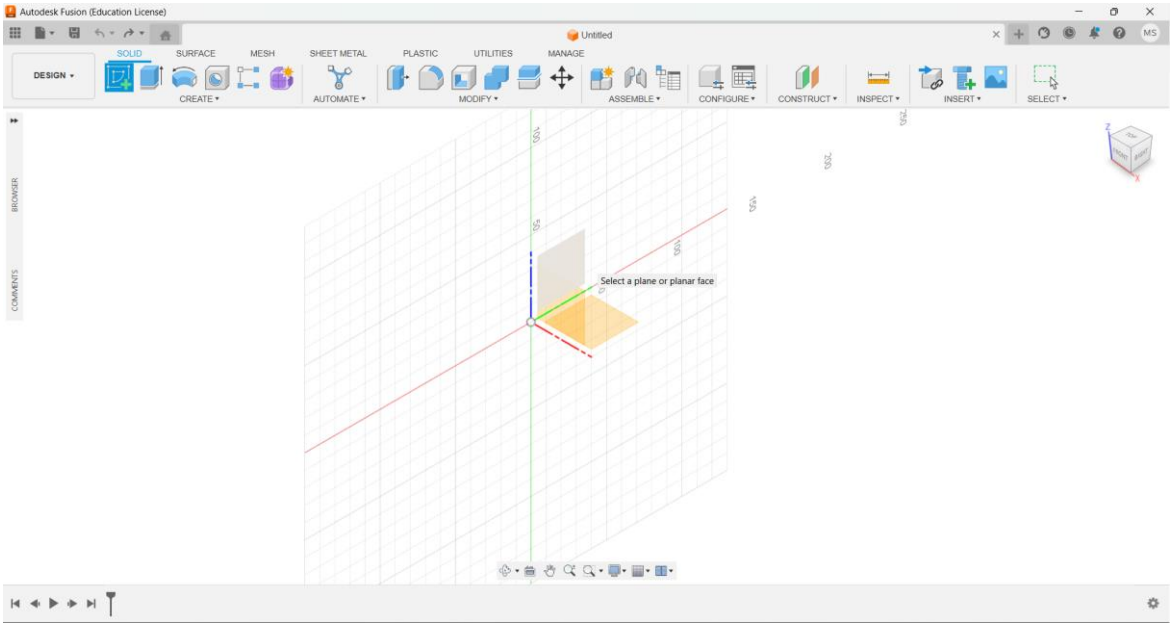
είναι η δημιουργία σχεδίου (Create Sketch) με το οποίο ο σχεδιαστής επιλέγει σε πιο από τα τρία επίπεδα (X,Y ή Y,Z ή Z,X) θέλει να σχεδιάσει ένα δισδιάστατο μοντέλο (Εικόνα 1.102). Ωστόσο, το κάθε δισδιάστατο επίπεδο χωρίζεται σε τέσσερα τεταρτημόρια ώστε ο σχεδιαστής ναι μεν να έχει χώρο σχεδίασης, αλλά και να μπορεί μετά στο τρισδιάστατο επίπεδο να κάνει ενέργειες ως προς συγκεκριμένους άξονες.



Εικόνα 1.101: Τα έξι βασικά μέρη του γραφικού περιβάλλοντος Autodesk Fusion [148].

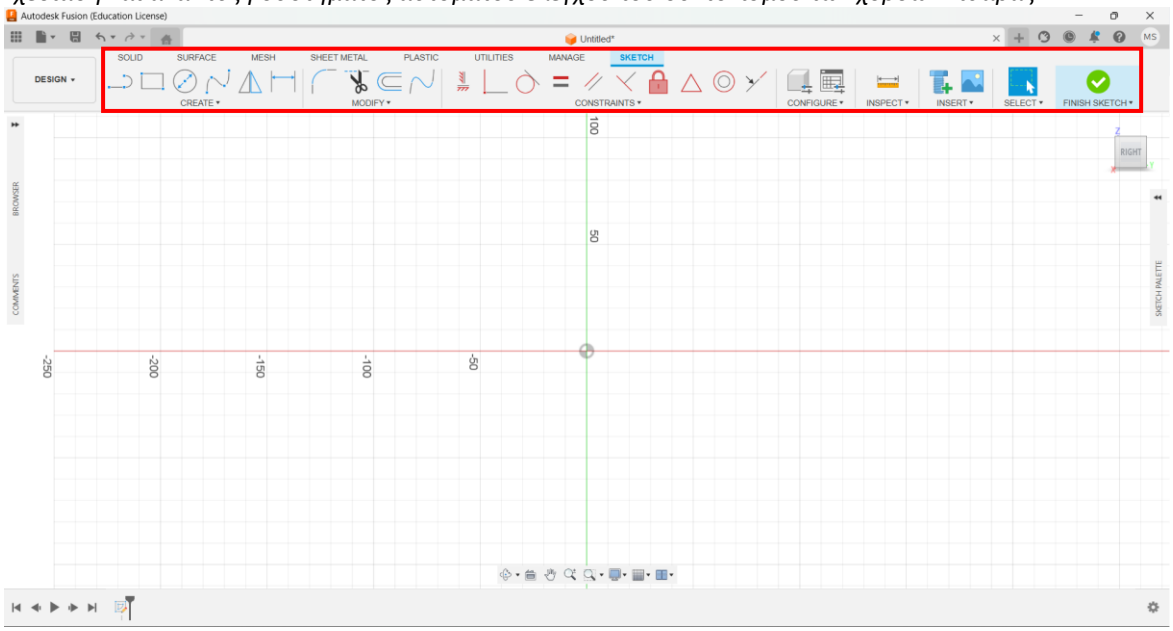
Μόλις ο σχεδιαστής επιλέξει το επίπεδο σχεδίασης, τότε αυτομάτως το επίπεδο σχεδιασμού αλλάζει σε δισδιάστατο και η εργαλειοθήκη αλλάζει σε σχεδίου (Sketch) (Εικόνα 1.103). Η εργαλειοθήκη αυτή περιλαμβάνει γεωμετρικά σχήματα, καμπύλες και άλλα εργαλεία σχεδιασμού και μέτρησης τα οποία είναι χρήσιμα για τον δισδιάστατο σχεδιασμό. Μετά την ολοκλήρωση του δισδιάστατου σχεδίου το επίπεδο σχεδιασμού επιστρέφει στην τρισδιάστατη μορφή με το δισδιάστατο σχέδιο στο αντίστοιχο επίπεδο που σχεδιάστηκε. Έπειτα, ένα άλλο χρήσιμο εργαλείο ονομάζεται εξώθηση (Extrude) το οποίο μετατρέπει ένα δισδιάστατο σχέδιο σε τρισδιάστατο κατά μήκος του αντίστοιχου κατακόρυφου άξονα. Επιπλέον, σημαντικό ρόλο στον τρισδιάστατο σχεδιασμό παίζει το εργαλείο περιστροφής (Revolve), το οποίο μετατρέπει ένα δισδιάστατο σχέδιο σε τρισδιάστατο περιστρέφοντάς το γύρω από τον κατακόρυφο άξονα. Ακόμα, μερικά βασικά εργαλεία τρισδιάστατου σχεδιασμού είναι το καμπύλωμα (Fillet), με το οποίο δημιουργείται μια καμπυλότητα στις επιθυμητές άκρες του σχεδίου, το κέλυφος (Shell) με το οποίο δημιουργείτε ένα κούφωμα εντός του τρισδιάστατου σχεδίου καθορίζοντας το πάχος των τοίχων και η επιθεώρηση (inspect), με την οποία μετριούνται αποστάσεις, πάχη, περίμετροι, ακτίνες κτλ. Εν συνεχεία, το τρίτο βασικό μέρος του Autodesk Fusion είναι το τρισδιάστατο επίπεδο σχεδιασμού (3) στο οποίο εμφανίζονται όλα τα δισδιάστατα και τρισδιάστατα μοντέλα που έχουν σχεδιαστεί. Ωστόσο, πάνω στο επίπεδο σχεδιασμού υπάρχει το τέταρτο βασικό μέρος το οποίο είναι ένας κύβος προσανατολισμού (4). Ο κύβος προσανατολισμού, δίνει την δυνατότητα στον σχεδιαστή να επιλέξει οπτική πλευρά του μοντέλου του. Επίσης, όταν πατηθεί το εικονίδιο με το σπιτάκι τότε το επίπεδο σχεδιασμού επιστρέφει στην αρχική οπτική γωνία. Επίσης, πάνω στο επίπεδο σχεδιασμού υπάρχει και το πέμπτο βασικό μέρος ένα μικρό μενού με διάφορες επιλογές για το επίπεδο σχεδιασμού (5). Πιο συγκεκριμένα, περιλαμβάνει την τροχιά (orbit) κατά την οποία ο

σχεδιαστής μπορεί να αλλάξει οπτική γωνία στο μοντέλο που έχει σχεδιαστεί, την μετατόπιση (pan), την μεγέθυνση, την σμίκρυνση, την μεγέθυνση ενός συγκεκριμένου παραθύρου, τις ρυθμίσεις της απεικόνισης, τις ρυθμίσεις του πλέγματος και των κουκίδων καθώς και την επιλογή πολλαπλών παραθύρων προβολής του σχεδίου. Το έκτο και τελευταίο βασικό μέρος του Autodesk Fusion βρίσκεται στο κάτω μέρος του γραφικού περιβάλλοντος το οποίο αποτελεί το ιστορικό των ενεργειών σχεδίασης (6). Επίσης, στο ιστορικό ενεργειών σχεδίασης σε περίπτωση που αλλαχτεί κάποια παλιότερη ενέργεια από τον σχεδιαστεί, μπορούν να επηρεαστούν και κάποιες από τις μετέπειτα ενέργειες σχεδίασης. Αν η ενέργεια που θα επηρεαστεί δεν είναι τόσο σημαντική τότε θα έχει ένα κίτρινο φόντο, αν όμως η ενέργεια επηρεαστεί με τέτοιον τρόπο που δεν θα μπορεί να πραγματοποιηθεί θα έχει ένα κόκκινο φόντο και το πιο πιθανό σενάριο είναι η επανάληψη της ενέργειας αυτής εκ νέου.



Εικόνα 1.102: Η επιλογή του δισδιάστατου επιπέδου κατά την επιλογή της δημιουργίας σχεδίου (Create Sketch) [149].

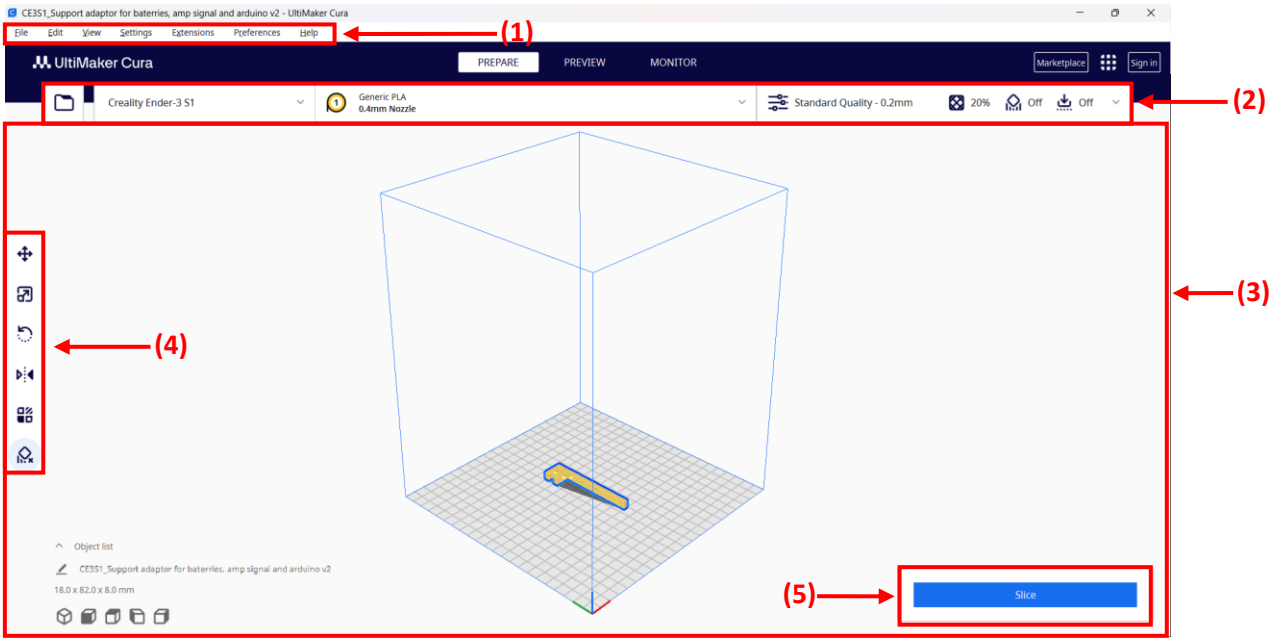
Ωστόσο, η Autodesk παρέχει μερικούς γρήγορους οδηγούς έναρξης σχετικά με το γραφικό περιβάλλον προγραμματισμού Fusion [113]. Πιο συγκεκριμένα, παρέχει εξοικείωση του σχεδιαστή με το γραφικό περιβάλλον σχεδίασης και επεξηγεί όλα τα εργαλεία σχεδίασης. Επίσης, η Autodesk παρέχει οδηγούς για το πώς μπορεί να γίνει η δισδιάστατη και η τρισδιάστατη σχεδίαση. Τέλος, μέσα από τους οδηγούς καθοδηγεί τον χρήστη για το πως μπορεί να κοινοποιήσει τα σχέδιά του, καθώς και το πως πρέπει να κάνει εξαγωγή του αρχείου και σε ποια μορφή για να πραγματοποιηθεί η τρισδιάστατη εκτύπωση.



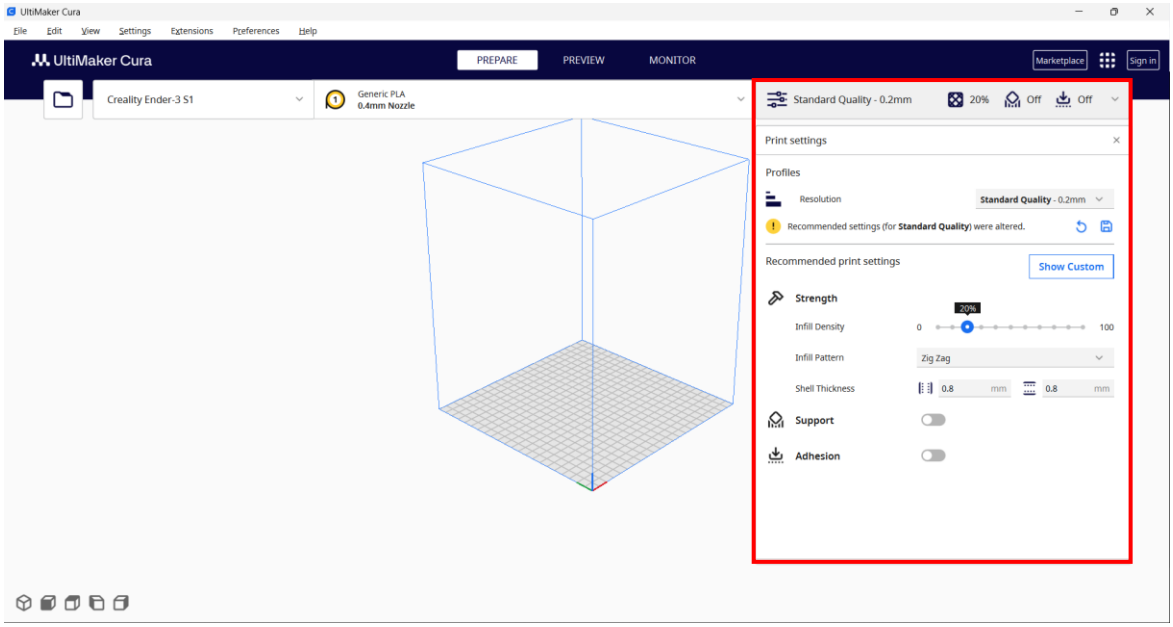
Εικόνα 1.103: Η μετάβαση σε δισδιάστατο επίπεδο σχεδιασμού και η αλλαγή της βιβλιοθήκης σε Sketch [150].

Από την άλλη, το γραφικό περιβάλλον του UltiMaker Cura χωρίζεται σε πέντε βασικά μέρη (Εικόνα 1.104). Πιο συγκεκριμένα, το πρώτο βασικό μέρος είναι η μπάρα μενού (1) στο πάνω μέρος, η οποία περιέχει το υπό-μενού αρχείο (File) για δημιουργία νέου έργου, άνοιγμα ήδη υπάρχον έργου, αποθήκευση έργου, εξαγωγή (Export) και έξοδο. Επίσης, περιλαμβάνει το υπό-μενού επεξεργασία (Edit), το οποίο περιέχει τις ενέργειες αναίρεση, επανάληψη, επιλογή και διαγραφή μοντέλων, καθώς και ομαδοποίηση των μοντέλων. Ακόμα, περιλαμβάνει το υπό-μενού προβολή (View) το οποίο δίνει επιλογές για την οπτική γωνία του μοντέλου. Επιπλέον, περιλαμβάνει το υπό-μενού ρυθμίσεις (Settings) το οποίο περιέχει γενικές ρυθμίσεις για τον τρισδιάστατο εκτυπωτή, όπως επιλογή μοντέλου, επιλογή υλικού και μεγέθους ακροφυσίου. Επιπροσθέτως, περιλαμβάνει το υπό-μενού επεκτάσεις (Extensions) το οποίο παρέχει τις δυνατότητες αναβάθμισης της εφαρμογής, την διαχείριση υποστήριξης της εφαρμογής και την τροποποίηση του κώδικα G. Ωστόσο, περιλαμβάνει ένα υπό-μενού προτιμήσεων (Preferences) το οποίο δίνει την δυνατότητα στον χρήστη να κάνει κάποιες ρυθμίσεις σχετικά με το γραφικό περιβάλλον του UltiMaker Cura και ένα υπό-μενού βοήθειας (Help) το οποίο παρέχει πληροφορίες σχετικά με το γραφικό περιβάλλον της εφαρμογής. Το δεύτερο βασικό μέρος του γραφικού περιβάλλοντος UltiMaker Cura είναι η εργαλειοθήκη (2) η οποία δίνει την δυνατότητα στον σχεδιαστή να ανοίξει ένα ήδη υπάρχον αρχείο, να επιλέξει το μοντέλο και το υλικό του τρισδιάστατου εκτυπωτή καθώς και κάποιες ρυθμίσεις σχετικά με την διαδικασία εκτύπωσης του τρισδιάστατου μοντέλου (Εικόνα 1.105). Πιο συγκεκριμένα, οι ρυθμίσεις αυτές έχουν να κάνουν με την ανάλυση του πάχους της κάθε στρώσης καθώς και με την ανθεκτικότητα του μοντέλου καθορίζοντας την πυκνότητα και το μοτίβο γεμίσματος του τρισδιάστατου μοντέλου. Επίσης, η εργαλειοθήκη αυτή δίνει την δυνατότητα προσθήκης στηριγμάτων σε συγκεκριμένα σημεία όπου υπάρχει κίνδυνος καταστροφής του μοντέλου εξαιτίας της βαρύτητας. Ακόμα, δίνει την δυνατότητα της δημιουργίας προσκόλλησης του τρισδιάστατου μοντέλου στην πλατφόρμα κατασκευής. Αυτή η επιλογή είναι σημαντική διότι αν το μοντέλο έχει βάση με μικρή επιφάνεια, υπάρχει μεγάλη πιθανότητα πτώσης κατά την διάρκεια της εκτύπωσης. Οι δυνατότητες σχεδίασης που προαναφέρθηκαν,

συνίστανται από το ίδιο γραφικό περιβάλλον της εφαρμογής. Ωστόσο, υπάρχουν περισσότερες δυνατότητες σχεδίασης επιλέγοντας το κουμπί εμφάνισης προσαρμογών (Show Costume) (Εικόνα 1.106) [114]. Πιο συγκεκριμένα, ο χρήστης μπορεί να επιλέξει το τύπο της προσκόλλησης, το τύπο της δημιουργίας στηρίξεων, την ψύξη του μοντέλου, την διαδρομή του Extruder, την ταχύτητα, τις θερμοκρασίες στο Extruder και στην πλατφόρμα κατασκευής ανάλογα με το υλικό, τον τύπο και την πυκνότητα γεμίσματος, το πάχος των στρωμάτων και των τοίχων του μοντέλου καθώς και την ποιότητα η οποία έχει να κάνει με το ύψος του κάθε στρώματος.



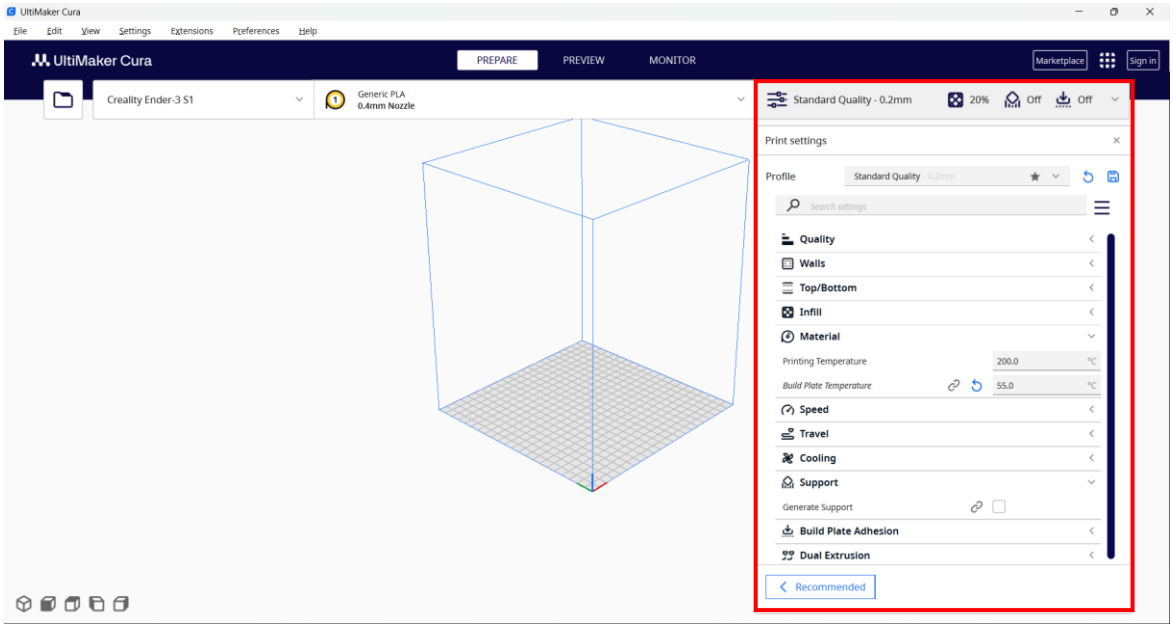
Εικόνα 1.104: Τα πέντε βασικά μέρη του γραφικού περιβάλλοντος της εφαρμογής UltiMaker Cura [151].



Εικόνα 1.105: Οι ρυθμίσεις που συνίστανται με την τρισδιάστατη εκτύπωση του μοντέλου [152].

Το τρίτο βασικό μέρος του γραφικού περιβάλλοντος της εφαρμογής είναι το επίπεδο σχεδιασμού (3) το οποίο δείχνει τις διαστάσεις της πλατφόρμας εκτύπωσης καθώς και το

ωφέλιμο ύψος. Αυτές οι διαστάσεις καθορίζονται από το μοντέλο του εκτυπωτή και δεν μπορούν να αλλαχτούν. Αυτομάτως, οι διαστάσεις αυτές καθορίζουν το μέγιστο μέγεθος του τρισδιάστατου μοντέλου που μπορεί να εκτυπωθεί. Επίσης, στο κάτω μέρος του επιπέδου σχεδιασμού υπάρχει ένα μικρό μενού με το οποίο ο χρήστης μπορεί να επιλέξει την οπτική γωνία στο μοντέλο. Εν συνεχεία, το τέταρτο και βασικό μέρος του γραφικού περιβάλλοντος είναι το μενού επεξεργασίας του τρισδιάστατου μοντέλου (4) το οποίο εμφανίζεται μόνο όταν υπάρχει κάποιο μοντέλο στο επίπεδο σχεδιασμού. Ουσιαστικά, περιλαμβάνει την μετατόπιση του μοντέλου σε οποιονδήποτε από τους τρεις άξονες, την αλλαγή μεγέθους του μοντέλου μεταβάλλοντας την κλίμακα του, την περιστροφή του μοντέλου καθώς και τον καθρεπτισμό του μοντέλου. Επίσης, περιλαμβάνει την επιλογή του τύπου εκτύπωσης του κάθε μοντέλου (κανονικό ή στήριγμα) καθώς και την απαγόρευση της δημιουργίας στηρίγματος σε οποιοδήποτε από τα μοντέλα. Το πέμπτο και τελευταίο βασικό μέρος της συγκεκριμένης εφαρμογής είναι το κουμπί Slice (5) το οποίο εμφανίζεται και αυτό μόνο όταν υπάρχει κάποιο μοντέλο στο επίπεδο σχεδιασμού. Με αυτό το κουμπί επιτυγχάνεται ο διαχωρισμός του τρισδιάστατου μοντέλου σε στρώματα και η δημιουργία ενός αρχείου σε κώδικα G ο οποίος αποθηκεύεται είτε στον υπολογιστή είτε σε μία εξωτερική μνήμη. Η εξωτερική μνήμη η οποία περιλαμβάνει οποιοδήποτε τρισδιάστατο μοντέλο σε κώδικα G είναι αυτό που χρειάζεται ο τρισδιάστατος εκτυπωτής για την εκκίνηση της εκτύπωσης.



Εικόνα 1.106: Επιπρόσθετες ρυθμίσεις για την τρισδιάστατη εκτύπωση του μοντέλου [153].

2 ΚΕΦΑΛΑΙΟ 2ο : Εργαστηριακό Μέρος

Σε αυτό το κεφάλαιο, γίνεται ανάλυση του εργαστηριακού μέρους της συγκεκριμένης διπλωματικής εργασίας. Πιο συγκεκριμένα, εξηγείται η λογική πίσω από την σχεδίαση και την ανάπτυξη του συστήματος αυτομάτου ελέγχου για τον συντονισμό των χορδών μιας κιθάρας. Εν συνεχεία, γίνεται ανάλυση των επιμέρους υποσυστημάτων ξεχωριστά τόσο ως προς την λειτουργία τους όσο και ως προς τον σκοπό τους στο σύστημα συντονισμού των χορδών. Με αυτόν τον τρόπο γίνεται κατανοητός ο ρόλος με τον οποίο συμβάλει το κάθε υποσύστημα στο σύστημα συντονισμού των χορδών. Έπειτα, γίνεται μια ανάλυση των χαρακτηριστικών των δύο τρισδιάστατων εκτυπωτών που χρησιμοποιήθηκαν για την τρισδιάστατη εκτύπωση για την υλοποίηση της διπλωματικής εργασίας. Τέλος, γίνεται επεξήγηση των τρισδιάστατων σχεδίων που δημιουργήθηκαν με σκοπό την στήριξη ολόκληρου του συστήματος συντονισμού επάνω στην κεφαλή της κιθάρας.

2.1 Η σχεδίαση και η ανάπτυξη του συστήματος αυτομάτου ελέγχου

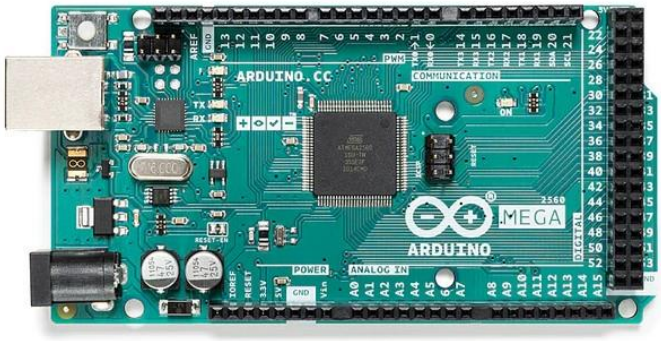
Αρχικά, για την σχεδίαση ενός οποιουδήποτε συστήματος είναι βασικό να κατανοήσει κανείς τους φυσικούς παράγοντες στους οποίους θα βασιστεί. Πιο συγκεκριμένα, οι φυσικές μεταβλητές είναι αυτές που καθοδηγούν τον σχεδιαστή για την εύρεση και την χρήση των απαραίτητων αισθητήρων. Με αυτόν τον τρόπο μπορεί να γίνει μέτρηση της φυσικής αυτής μεταβλητής και να εισαχθεί στο αντίστοιχο σύστημα το οποίο θα το αντιλαμβάνεται ως πραγματική τιμή. Στην συνέχεια, το σύστημα συγκρίνει την πραγματική τιμή με την επιθυμητή τιμή που θα πρέπει να έχει η φυσική μεταβλητή και ανάλογα με την διαφορά παράγεται ένα σήμα σφάλματος. Έπειτα, το σύστημα διαχειρίζεται τους αντίστοιχους ενεργοποιητές, ανάλογα με το σήμα σφάλματος, προσπαθώντας να μηδενίσει την διαφορά μεταξύ της επιθυμητής και της πραγματικής τιμής. Οι ενεργοποιητές, είναι συστήματα τα οποία έχουν σκοπό να επηρεάσουν την φυσική μεταβλητή και κατά συνέπεια την πραγματική τιμή. Με αυτόν τον τρόπο, το σύστημα αλληλεπιδρά με το περιβάλλον του προσπαθώντας να το ισορροπήσει στην επιθυμητή τιμή.

Με βάση την προαναφερθείσα λογική, σχεδιάστηκε και αναπτύχθηκε το σύστημα αυτομάτου ελέγχου για τον συντονισμό των χορδών μιας κιθάρας. Πιο συγκεκριμένα, για τον συντονισμό των χορδών απαιτείται η επίγνωση μιας φυσικής μεταβλητής η οποία δεν είναι άλλη από την συχνότητα ταλάντωσης των χορδών. Η ανίχνευση της συχνότητας μπορεί να γίνει με πολλούς τρόπους, όπως με την σύλληψη του ήχου μέσω μικροφώνου ή μέσω των μαγνητών αν πρόκειται για ηλεκτρική κιθάρα ή μέσω των δονήσεων που παράγονται στο σώμα του οργάνου όταν διεγείρεται κάποια από τις χορδές. Πριν όμως παρθεί η τελική απόφαση πρέπει να αναλυθεί ο κάθε ένας από τους προηγούμενους τρόπους και να επιλεχθεί ο καλύτερος. Η σύλληψη του ήχου μέσω μικροφώνου είναι μία εύκολη και γρήγορη λύση αλλά ταυτόχρονα ελαττωματική. Ουσιαστικά, με την χρήση του μικροφώνου, συλλαμβάνεται μαζί με τον ήχο του οργάνου και ο θόρυβος από το περιβάλλον, γεγονός ανεπιθύμητο για το σύστημα διότι οδηγεί σε εσφαλμένες μετρήσεις και υπολογισμούς. Ωστόσο, σε περίπτωση κουρδίσματος μιας ηλεκτρικής κιθάρας θα πρέπει να είναι συνδεδεμένη με έναν ενισχυτή ώστε ο ήχος να είναι αρκετά δυνατός και να υπερτερεί από τον θόρυβο του περιβάλλοντος. Σε αντίθετη περίπτωση, επειδή οι ηλεκτρικές κιθάρες κατασκευαστικά δεν περιλαμβάνουν ηχείο και ο ήχος που παράγεται

όταν διεγείρεται μία χορδή είναι χαμηλής εντάσεως, έχει ως αποτέλεσμα η σύλληψη από το μικρόφωνο να μην είναι ικανοποιητική έως και αδύνατη. Επομένως, η επιλογή μικροφώνου για την σύλληψη του ήχου στην συγκεκριμένη εργασία απορρίπτεται. Τώρα, στην περίπτωση της σύλληψης του ήχου μέσω μαγνητών είναι μια πολύ καλή λύση όσον αφορά την αντιμετώπιση του θορύβου από το περιβάλλον. Το πρόβλημα με αυτή την επιλογή είναι ότι μόνο οι ηλεκτρικές και οι ηλεκτροακουστικές κιθάρες θα μπορούν να κουρδιστούν με αυτό το σύστημα, ενώ οι κλασσικές δεν θα μπορούν διότι δεν περιλαμβάνουν μαγνήτες. Δηλαδή, με αυτό τον τρόπο, το σύστημα εξειδικεύετε για συγκεκριμένο τύπο κιθάρων, γεγονός ανεπιθύμητο για την συγκεκριμένη διπλωματική εργασία. Επομένως, η σύλληψη του ήχου μέσω μαγνητών απορρίπτεται. Η τελευταία λύση είναι η σύλληψη των δονήσεων από το μουσικό όργανο όταν διεγείρεται κάποια από τις χορδές. Αυτό επιτυγχάνεται με την χρήση ενός μικροφώνου επαφής το οποίο τοποθετείται στην κεφαλή του οργάνου και διαβάζει τις δονήσεις που παράγονται στο μουσικό όργανο. Με αυτόν τον τρόπο, εξαλείφεται ο θόρυβος από το περιβάλλον και μπορεί να εφαρμοστεί σε όλους τους τύπους κιθάρας άσχετα με το αν είναι ηλεκτρική, ακουστική ή κλασσική. Συνεπώς, για την συγκεκριμένη διπλωματική εργασία επιλέχθηκε η σύλληψη των δονήσεων μέσω του μικροφώνου επαφής Korg CM300 (Εικόνα Εικόνα 2.1) [115] [116]. Έτσι με τους κατάλληλους υπολογισμούς μπορεί να εντοπιστεί η θεμελιώδης συχνότητα ταλάντωσης των χορδών.

Εικόνα 2.1: Το μικρόφωνο επαφής Korg CM300 [154].

Εν συνεχεία, θα πρέπει να επιλεχθεί ένας μικροελεγκτής ο οποίος με τους κατάλληλους υπολογισμούς θα εντοπίσει την πραγματική συχνότητα των χορδών και μέσω των κατάλληλων ενεργοποιητών θα μεταβάλει την συχνότητα ταλάντωσης των χορδών. Πιο συγκεκριμένα, επιλέχθηκε το Arduino Mega2560 (Εικόνα Εικόνα 2.2) διότι είναι προσιτός όσον αφορά το κόστος, παρέχει αρκετές εισόδους/εξόδους, είναι εύκολος ως προς τον προγραμματισμό του και η πλατφόρμα Arduino παρέχει δωρεάν το περιβάλλον ανάπτυξης προγραμματισμού [84]. Ωστόσο, υπάρχει μια εξοικείωση με τα Arduino και αυτό έπαιξε σημαντικό ρόλο για την επιλογή του συγκεκριμένου μικροελεγκτή.



Εικόνα 2.2: Το Arduino Mega 2560 [155].

Επιπροσθέτως, πρέπει να επιλεχτούν οι κατάλληλοι ενεργοποιητές οι οποίοι θα μεταβάλλουν τις συχνότητες ταλάντωσης των χορδών της κιθάρας. Αυτό μπορεί να επιτευχθεί περιστρέφοντας τα αντίστοιχα κλειδιά των χορδών της κιθάρας. Επομένως, για την περιστροφή του κάθε κλειδιού θα πρέπει να γίνει χρήση κάποιου είδους ηλεκτρικού κινητήρα. Ας δούμε ποιος τύπος κινητήρα είναι ο καταλληλότερος για την συγκεκριμένη εφαρμογή. Πιο συγκεκριμένα, οι υποψήφιοι τύποι κινητήρων είναι οι DC κινητήρες, οι σερβοκινητήρες και οι βηματικοί κινητήρες. Όσον αφορά τους DC κινητήρες (Εικόνα Εικόνα 2.3), είναι μια πολύ καλή λύση διότι έχουν μικρό μέγεθος με αποτέλεσμα να μπορούν να τοποθετηθούν επάνω στα κλειδιά της κιθάρας [117]. Όμως τα κλειδιά της κιθάρας για να περιστραφούν χρειάζονται αρκετή ροπή και αργή περιστροφή για να επιτευχθεί με ακρίβεια ο συντονισμός των χορδών. Όμως, όσο πιο μικρό το μέγεθός των DC κινητήρων τόσο μικρότερη είναι η ροπή τους, με αποτέλεσμα να μην μπορούν να περιστρέψουν τα κλειδιά της κιθάρας. Επίσης, οι DC κινητήρες περιστρέφονται με πολλές στροφές γεγονός που οδηγεί στην γρήγορη περιστροφή του κλειδιού και κατά συνέπεια στην χαμηλή ακρίβεια του συντονισμού των χορδών. Ωστόσο, οι στροφές μπορούν να ελαττωθούν με την χρήση μειωτήρα (Εικόνα Εικόνα 2.4) ο οποίος δεν είναι τίποτα άλλο από ένα σύστημα γραναζιών με κάποια αναλογία (πχ. 40:1, δηλαδή με σαράντα στροφές στην είσοδο του μειωτήρα, επιτυγχάνεται μία πλήρης περιστροφή στην έξοδο) [118]. Με αυτόν τον τρόπο μπορούν να ελαττωθούν οι στροφές του κινητήρα αλλά προκύπτει το πρόβλημα του μεγέθους μιας και οι μειωτήρες είναι ογκώδης. Επίσης, οι DC κινητήρες χρειάζονται ένα ολοκληρωμένο τσιπ για τον έλεγχο της φοράς περιστροφής τους το οποίο περιλαμβάνει το κύκλωμα μιας γέφυρας τύπου «Η». Συνεπώς, για την συγκεκριμένη εφαρμογή, οι DC κινητήρες απορρίπτονται λόγο μεγέθους, ροπής και στροφών.

Εικόνα 2.3: DC κινητήρες διαφόρων μεγεθών [156].

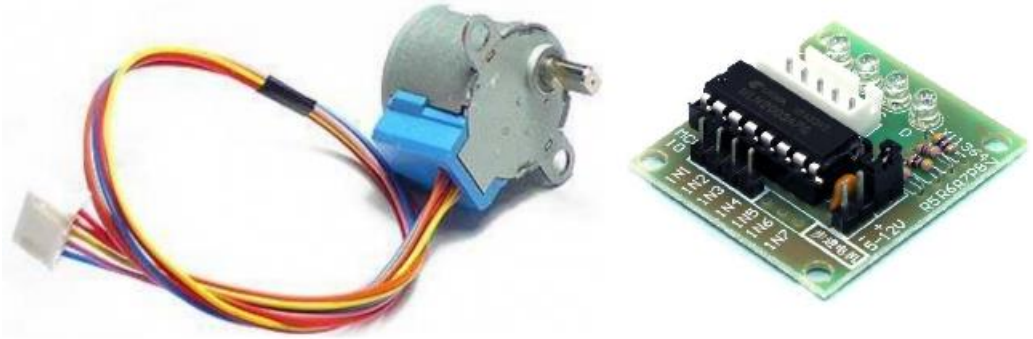
Εικόνα 2.4: DC κινητήρες με μειωτήρα [157].

Μία άλλη λύση, είναι η χρήση των σερβοκινητήρων (Εικόνα Εικόνα 2.5), οι οποίοι έχουν εντελώς διαφορετική λειτουργία από τους DC κινητήρες [119]. Πιο συγκεκριμένα, οι σερβοκινητήρες εκτός από την τροφοδοσία, δέχονται και ένα σήμα με διαμορφωμένο πλάτος παλμού (PWM) το οποίο έχει σκοπό να περιστρέψει τον άξονα στην επιθυμητή γωνία. Επιπροσθέτως, οι σερβοκινητήρες είναι μικροί, παρέχουν μια ικανοποιητική ροπή και μπορούν να περιστρέψουν τον άξονά τους με αρκετά καλή ακρίβεια προσφέροντας με αυτόν τον τρόπο ακρίβεια στο κούρδισμα. Ο μοναδικός περιορισμός των σερβοκινητήρων είναι οι μοίρες περιστροφής. Ουσιαστικά, υπάρχουν σερβοκινητήρες οι οποίοι μπορούν να περιστραφούν από 0^o έως 360^o γεγονός που δεν επωφελή την συγκεκριμένη εφαρμογή, διότι το κλειδί της κιθάρας μπορεί να χρειαστεί να περιστραφεί παραπάνω από 360^o για να κουρδιστεί σωστά η χορδή. Συνεπώς, οι σερβοκινητήρες απορρίπτονται για την συγκεκριμένη εφαρμογή.

Εικόνα 2.5: Διάφοροι σερβοκινητήρες [158].

Μία άλλη λύση, είναι η χρήση βηματικών κινητήρων για την περιστροφή των κλειδιών (Εικόνα Εικόνα 2.6) [120]. Αυτοί οι τύποι κινητήρων έχουν έναν ξεχωριστό τρόπο τόσο κατασκευής όσο και λειτουργίας. Πιο συγκεκριμένα, αποτελούνται από πηνία και μαγνήτες και η περιστροφή του άξονα τους χωρίζεται σε βήματα. Αυτό επωφελή την συγκεκριμένη εφαρμογή μιας και η περιστροφή των κλειδιών του οργάνου μπορεί να γίνει αργά. Συνεπώς, το κούρδισμα των χορδών μπορεί να γίνει με ακρίβεια. Επομένως, για την συγκεκριμένη διπλωματική εργασία επιλέχθηκε η χρήση βηματικού κινητήρα και πιο συγκεκριμένα ο τύπος 28BYJ-48 ο οποίος είναι μικρός σε μέγεθος, παρέχει ικανοποιητική ροπή και είναι προσιτός όσον αφορά το κόστος. Επιπλέον, για να μπορέσει να ελεγχθεί ο συγκεκριμένος βηματικός κινητήρας θα πρέπει να γίνει χρήση οδηγού (drive shield) και πιο συγκεκριμένα του τύπου ULN2003 (Εικόνα Εικόνα 2.7) [120] [122].

Εικόνα 2.6: Διάφοροι βηματικοί κινητήρες [159].



Εικόνα 2.7: Ο βηματικός κινητήρας 28BYJ-48 με τον αντίστοιχο οδηγό (drive shield) ULN2003 [160].

Τέλος, θα πρέπει να εφαρμοστεί μια διεπαφή ώστε ο χρήστης να μπορεί αλληλεπιδρά με το σύστημα. Πιο συγκεκριμένα, θα μπορούσε να γίνει χρήση μιας οθόνης LCD (Εικόνα Εικόνα 2.8) η οποία θα προτρέπει τον χρήστη να διεγείρει κάποια από τις χορδές ώστε το σύστημα να ξεκινήσει το αυτόματο κούρδισμα [123]. Επίσης, θα μπορούσε να εμφανίζει την επιθυμητή και την πραγματική συχνότητα της κάθε χορδής ώστε να ενημερώνεται ο χρήστης για την πρόοδο του κουρδίσματος. Όμως, τοποθετώντας μόνο μία LCD οθόνη δεν δίνει στον χρήστη τόσο την δυνατότητα αλληλεπίδρασης άλλα μόνο παρακολούθησης. Δηλαδή, στην περίπτωση που θα πρέπει το σύστημα να δώσει στον χρήστη την δυνατότητα επιλογής κουρδίσματος αυτό δεν μπορεί να γίνει μόνο με την χρήση μιας απλής LCD οθόνης. Συνεπώς, θα πρέπει να γίνει προσθήκη ενός επιπλέον υποσυστήματος μέσω του οποίου θα μπορεί ο χρήστης να επιλέξει το επιθυμητό κούρδισμα. Με αυτό τον τρόπο όμως το σύστημα γίνεται ακόμα πιο πολύπλοκο γεγονός το οποίο οδηγεί στην απόρριψη της ιδέας αυτής. Μια άλλη λύση, θα ήταν η δημιουργία μιας εφαρμογής για smartphone η οποία θα συνδεόταν με τον μικροελεγκτή μέσω Bluetooth ή μέσω Wifi. Σκοπός της εφαρμογής είναι η καθοδήγηση του χρήστη για το ποια χορδή θα πρέπει να διεγείρει, η εμφάνιση της πραγματικής και της επιθυμητής συχνότητας της κάθε χορδής, καθώς επίσης θα δίνει την δυνατότητα στον χρήστη να επιλέξει το επιθυμητό κούρδισμα. Αυτή η ιδέα είναι πάρα πολύ καλή αλλά όχι τόσο πρακτική, διότι ο μουσικός όταν θα θέλει να κουρδίσει την κιθάρα θα πρέπει ταυτόχρονα να ισορροπεί το όργανο πάνω του, να κρατάει με το ένα χέρι το smartphone και με το άλλο χέρι να προσπαθεί να διεγείρει την επιθυμητή χορδή. Επομένως, για λόγους καλής πρακτικής η ιδέα αυτή απορρίπτεται. Μια άλλη λύση θα ήταν η χρήση μιας LCD οθόνης με ενσωματωμένο πληκτρολόγιο (Εικόνα Εικόνα 2.9) ώστε ο χρήστης να μπορεί να επιλέξει το επιθυμητό κούρδισμα [124]. Επομένως, θα μπορούσε να δημιουργηθεί ένα μενού με τα επιθυμητά κουρδίσματα τα οποία θα εμφανίζονται μέσω της LCD οθόνης. Έπειτα αφού επιλεχθεί κάποιο από τα επιθυμητά κουρδίσματα, τότε ξεκινάει η διαδικασία

αυτόματου κουρδίσματος των χορδών εμφανίζοντας στην LCD οθόνη την χορδή που πρέπει να διεγερθεί κάθε φόρα. Με αυτό τον τρόπο, το σύστημα γίνεται λειτουργικό, εύκολα κατανοητό από τον χρήστη, πιο απλό και οικονομικό ως προς την ανάπτυξη του. Συνεπώς, για την ανάπτυξη του συστήματος επιλέχθηκε η χρήση LCD οθόνης με ενσωματωμένο πληκτρολόγιο.

Εικόνα 2.8: Απλή οθόνη LCD 16 χαρακτήρων και 2 σειρών [161].

Εικόνα 2.9: Οθόνη LCD 16 χαρακτήρων και 2 σειρών με ενσωματωμένο πληκτρολόγιο και δυνατότητα ενσωμάτωσης πάνω στην πλακέτα Arduino Mega2560 [162].

2.2 Ανάλυση των υποσυστημάτων

Σε αυτή την ενότητα, γίνεται αναφορά στα χαρακτηριστικά των υποσυστημάτων καθώς και ανάλυση της λειτουργίας τους. Με αυτόν τον τρόπο, γίνεται κατανοητή η λειτουργία των υποσυστημάτων καθώς και ο σκοπός τους στο συνολικό σύστημα αυτόματου κουρδίσματος. Επίσης, είναι πολύ σημαντικό πριν την ανάπτυξη οποιουδήποτε συστήματος να κατανοηθεί και να δοκιμαστεί το κάθε υποσύστημα ξεχωριστά. Έτσι, ο σχεδιαστής αντιλαμβάνεται ακριβώς πως να χρησιμοποιήσει το κάθε υποσύστημα κατά την ανάπτυξη του συνολικού συστήματος. Ακόμα, στην ενότητα αυτή γίνεται επεξήγηση του κώδικα ο οποίος είναι απαραίτητος για τον έλεγχο του κάθε υποσυστήματος. Με αυτόν τον τρόπο, το στήσιμο του κώδικα του συνολικού συστήματος, που παρουσιάζεται στο επόμενο κεφάλαιο, μπορεί να χωριστεί σε κομμάτια κάνοντάς τον πιο απλό και πιο ευανάγνωστο.

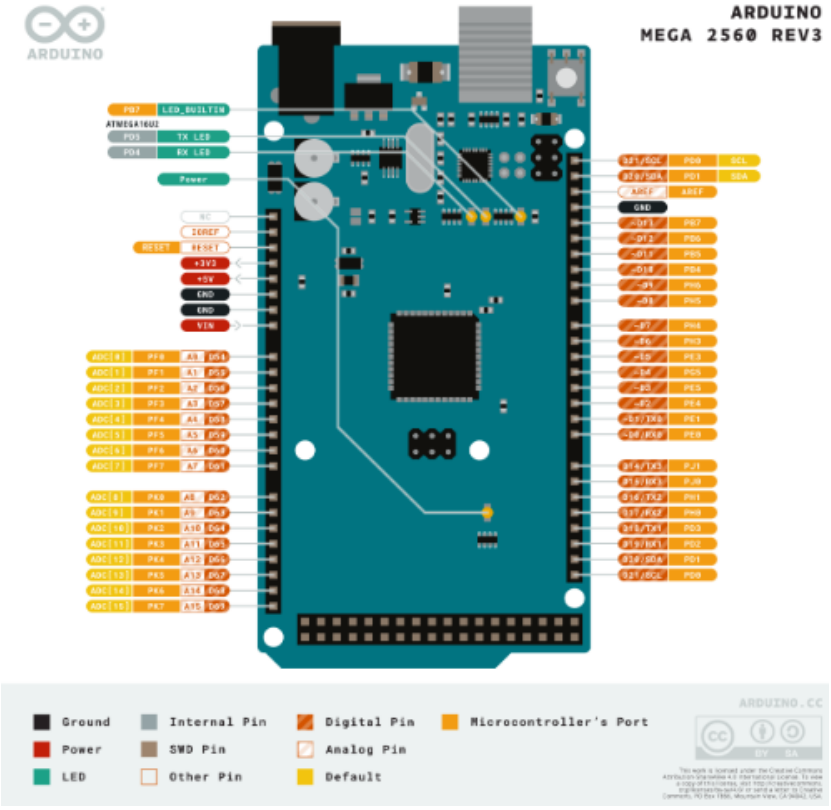
2.2.1 **Το Arduino Mega 2560**

Αρχικά, το βασικότερο υποσύστημα για την ανάπτυξη του συστήματος αυτομάτου κουρδίσματος είναι ο μικροελεγκτής. Πιο συγκεκριμένα, ο μικροελεγκτής αποτελεί τον εγκέφαλο του συστήματος αυτομάτου κουρδίσματος διότι καθορίζει την σειρά των διεργασιών καθώς επίσης κάνει και τους απαραίτητους υπολογισμούς για την εύρεση της θεμελιώδης συχνότητας των χορδών. Όπως ειπώθηκε προηγουμένως, για την ανάπτυξη του συγκεκριμένου συστήματος έγινε χρήση του Arduino Mega2560 (Εικόνα Εικόνα 2.8), το οποίο περιλαμβάνει τον μικροελεγκτή ATmega2560 της εταιρίας Atmel. Πιο συγκεκριμένα, η εξωτερική τάση τροφοδοσίας της πλακέτας Arduino Mega2560 πρέπει να είναι από 7Volt-12Volt ενώ η τάση λειτουργίας του μικροελεγκτή είναι στα 5Volt. Στην συγκεκριμένη εργασία, χρησιμοποιείται μία επαναφορτιζόμενη μπαταρία των 9Volt για την τροφοδοσία του Arduino Mega 2560. Εν συνεχεία, ο συγκεκριμένος μικροελεγκτής περιέχει 54 ψηφιακές εισόδους-εξόδους μέσω ακροδεκτών εκ των οποίων οι 15 υποστηρίζουν διαμόρφωση πλάτος παλμού (PWM). Το μέγιστο ρεύμα που μπορεί να δώσει ο καθένας ακροδέκτης εισόδου-εξόδου ξεχωριστά είναι 40mA. Ο συγκεκριμένος μικροελεγκτής εκτός από τις ψηφιακές θύρες εισόδου εξόδου, περιλαμβάνει και 16 αναλογικές εισόδους με 10bit ανάλυση καθώς και ακροδέκτες τροφοδοσίας (Εικόνα Εικόνα 2.11). Επιπλέον, ο μικροελεγκτής περιλαμβάνει μία μνήμη Flash των 256kB, εκ των οποίων τα 8kB χρησιμοποιούνται από τον Bootloader. Επίσης, περιλαμβάνει μία μνήμη SRAM των 8kB και μία EEPROM των 4kB. Ακόμα, η ταχύτητα ρολογιού του συγκεκριμένου μικροελεγκτή αγγίζει τα 16MHz. Επιπλέον, ο προγραμματισμός του Arduino Mega2560 γίνεται σε γλώσσα προγραμματισμού C μέσω του περιβάλλοντος Arduino IDE, το οποίο παρέχεται δωρεάν [82] [84].

Εικόνα 2.10: Το Arduino Mega2560 με τον μικροελεγκτή ATmega2560 της Atmel [163].

Συνεπώς, πάνω στο Arduino Mega 2560 συνδέονται όλα τα υποσυστήματα και ελέγχονται κατάλληλα. Πιο συγκεκριμένα, όλο το σύστημα ενεργοποιείτε με την χρήση ενός διακόπτη και ο μικροελεγκτής εμφανίζει αρχικά στην LCD οθόνη τα πιθανά κουρδίσματα για να επιλέξει ο χρήστης. Αφού γίνει η επιλογή κουρδίσματος, τότε στην LCD οθόνη εμφανίζεται ένα μήνυμα το οποίο προτρέπει τον χρήστη να διεγείρει την πρώτη χορδή. Έπειτα, ο μικροελεγκτής λαμβάνει το αναλογικό σήμα από το μικρόφωνο επαφής και υπολογίζει την θεμελιώδη συχνότητα της χορδής. Εν συνεχεία, υπολογίζει την διαφορά μεταξύ της επιθυμητής και της πραγματικής τιμής της συχνότητας και ανάλογα με την τιμή του σφάλματος που προκύπτει περιστρέφει κατάλληλα τον κινητήρα της αντίστοιχης χορδής. Μόλις το σφάλμα μηδενιστή ή εισέλθει εντός των ορίων του κατωφλιού που έχει οριστεί, τότε στην οθόνη LCD εμφανίζεται ένα μήνυμα το οποίο προτρέπει τον χρήστη να διεγείρει την επόμενη χορδή. Η διαδικασία αυτή

επαναλαμβάνεται έως ότου κουρδιστούν όλες οι χορδές. Αφού κουρδιστούν όλες οι χορδές, τότε ο μικροελεγκτής εμφανίζει ξανά το μενού κουρδίσματος στην LCD οθόνη. Στην περίπτωση όμως που ο χρήστης δεν επιθυμεί να κουρδίσει ξανά την κιθάρα τότε απλά μέσω του διακόπτη απενεργοποιεί το σύστημα αυτόματου κουρδίσματος.



Εικόνα 2.11: Διάγραμμα των ακροδεκτών του Arduino Mega2560 [164].

2.2.2 Το μικρόφωνο επαφής Korg CM300

Το μικρόφωνο επαφής αποτελεί ένα από τα βασικότερα υποσυστήματα, μιας και ευθύνεται για την σύλληψη των δονήσεων από το σώμα του οργάνου όταν διεγείρεται κάποια από τις χορδές. Για την υλοποίηση της συγκεκριμένης εφαρμογής, χρησιμοποιήθηκε το μικρόφωνο επαφής CM300 της εταιρίας Korg (Εικόνα Εικόνα 2.12). Πιο συγκεκριμένα, περιλαμβάνει ένα πιεζοστοιχείο ως αισθητήριο κραδασμών, το οποίο παράγει στην έξοδό του ένα σήμα ανάλογο των δονήσεων που δέχεται. Ωστόσο, το σήμα που παράγεται, είναι ασθενές και θα πρέπει να χρησιμοποιηθεί ένα κύκλωμα ενίσχυσης το οποίο αναλύεται στην επόμενη ενότητα. Το μικρόφωνο επαφής περιέχει ένα λαστιχένιο εσωτερικό το οποίο το βοηθά στην τοποθέτηση και στην στήριξη πάνω στο όργανο. Επίσης, με το λάστιχο προστατεύεται και το μουσικό όργανο από τυχόν γρατζουνιές και φθορές που θα μπορούσαν να προκύψουν κατά την τοποθέτηση του μικροφώνου επαφής. Ακόμα, το μέγιστο άνοιγμα του μικροφώνου επαφής φτάνει μέχρι τα 30mm, το οποίο το καθιστά κατάλληλο για την τοποθέτησή του στην κεφαλή της κιθάρας, ενώ η επαναφορά του γίνεται με την βοήθεια ενός ελατηρίου. Επιπροσθέτως, το μικρόφωνο επαφής ζυγίζει περίπου 35gr γεγονός το οποίο επωφελή το σύστημα διότι είναι αρκετά ελαφρύ και δεν προσθέτει κάποιο ιδιαίτερο βάρος στο μουσικό όργανο. Τέλος, ο σχεδιασμός του καλωδίου έχει γίνει με τέτοιον τρόπο που προσφέρει ανθεκτικότητα και αποτελεσματική

μετάδοση του σήματος. Επίσης, περιλαμβάνει στην άκρη του ένα αρσενικό βύσμα Jack για να μπορεί να συνδεθεί στην αντίστοιχη θύρα κάποιου κουρδιστηριού [115] [116].

Εικόνα 2.12: Μικρόφωνο επαφής CM300 της εταιρίας Korg [165].

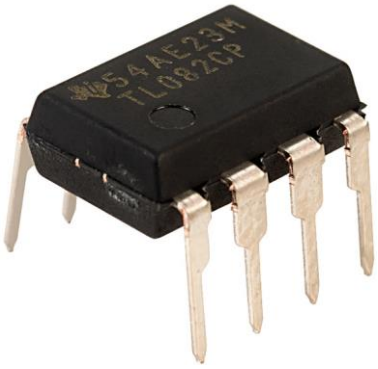
2.2.3 Η πλακέτα ενίσχυσης και μετατόπισης του σήματος

Σε αυτή την ενότητα, αναλύεται η πλακέτα ενίσχυσης του σήματος που παράγεται από το μικρόφωνο επαφής. Όπως ειπώθηκε στην προηγούμενη ενότητα, το σήμα που παράγεται από το μικρόφωνο επαφής, είναι ασθενές και γι αυτό το λόγο αναπτύχθηκε ένα κύκλωμα ενίσχυσης. Πιο συγκεκριμένα, το κύκλωμα ενίσχυσης αναπτύχθηκε σύμφωνα με την δημοσίευση της Amanda Ghassaei για την ηχητική είσοδο σε Arduino (https://www.instructables.com/Arduino-Audio-Input/). Αρχικά, τα υλικά που χρησιμοποιήθηκαν για την υλοποίηση του κυκλώματος ενίσχυσης είναι:

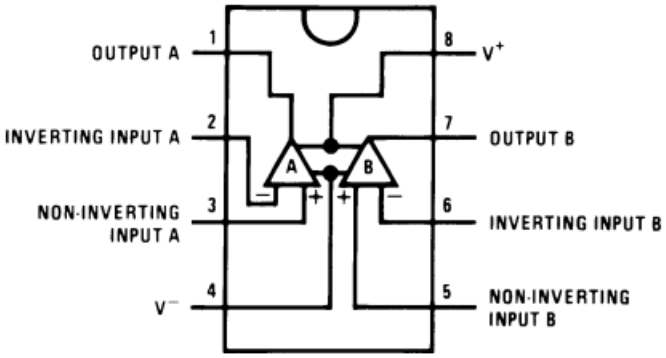
- 1 τελεστικός ενισχυτής TL082 της Texas Instruments
- 2 επαναφορτιζόμενες μπαταρίες των 9Volt
- 3 αντιστάσεις των 100KOhm
- 1 αντίσταση των 22KOhm
- 1 ηλεκτρολυτικός πυκνωτής των 10μF
- 1 κεραμικός πυκνωτής των 47nF
- 1 πλακέτα PCB 5x7 cm
- 1 κλέμες τριών θέσεων 2.54mm
- 2 κλέμες δύο θέσεων 2.54mm

Για αρχή, είναι σημαντικό να κατανοηθεί ο τρόπος συνδεσμολογίας του τελεστικού ενισχυτή TL082 της Texas Instruments (Εικόνα Εικόνα 2.13) [125]. Σύμφωνα με το εγχειρίδιο δεδομένων, που δίνει η Texas Instruments, το ολοκληρωμένο κύκλωμα TL082 περιλαμβάνει δύο τελεστικούς ενισχυτές οι οποίοι υλοποιούνται με την χρήση τρανζίστορ JFET. Πιο συγκεκριμένα, το ολοκληρωμένο κύκλωμα TL082 περιλαμβάνει οκτώ ακροδέκτες από τους οποίους οι δύο αποτελούν την τροφοδοσία του ολοκληρωμένου, οι

άλλοι δύο αποτελούν τις μη-αναστρέφουσες εισόδους των ενισχυτών, οι άλλοι δύο τις αναστρέφουσες εισόδους των ενισχυτών και οι δύο εναπομείναντες τις εξόδους των τελεστικών ενισχυτών (Εικόνα Εικόνα 2.14). Σύμφωνα με την Εικόνα 2.14, ο ακροδέκτης υπ' αριθμόν 1 αποτελεί την έξοδο του πρώτου τελεστικού ενισχυτή, ο ακροδέκτης υπ' αριθμόν 2 αποτελεί την αναστρέφουσα είσοδο του πρώτου τελεστικού ενισχυτή, ενώ ο ακροδέκτης υπ' αριθμόν 3 αποτελεί την μη-αναστρέφουσα είσοδο του πρώτου τελεστικού ενισχυτή. Εν συνεχεία, ο ακροδέκτης υπ' αριθμόν 4 αποτελεί το χαμηλό δυναμικό της τροφοδοσίας του ολοκληρωμένου (-V), ο ακροδέκτης υπ' αριθμόν 5 αποτελεί την μη-αναστρέφουσα είσοδο του δεύτερου τελεστικού ενισχυτή, ο ακροδέκτης υπ' αριθμόν 6 αποτελεί την αναστρέφουσα είσοδο του δεύτερου τελεστικού ενισχυτή, ο ακροδέκτης υπ' αριθμόν 7 αποτελεί την έξοδο του δεύτερου τελεστικού ενισχυτή, ενώ ο τελευταίος ακροδέκτης υπ' αριθμόν 8 αποτελεί το υψηλό δυναμικό της τροφοδοσίας του ολοκληρωμένου (+V). Ακόμα, η τάση τροφοδοσίας που χρειάζεται το ολοκληρωμένο κύκλωμα για να λειτουργήσει είναι ±18Volt, η οποία παρέχεται από τις δύο επαναφορτιζόμενες μπαταρίες των 9Volt όταν συνδεθούν σε σειρά. Επίσης, το εύρος θερμοκρασίας για την ορθή λειτουργία του ολοκληρωμένου πρέπει είναι μεταξύ 0^{o}C έως 70^{o}C, ενώ η τάση μεταξύ των διαφορικών εισόδων του ολοκληρωμένου κυκλώματος πρέπει να είναι μεταξύ των ±30Volt [126].



Εικόνα 2.13: Ο τελεστικός ενισχυτής TL082 της Texas Instruments [166].



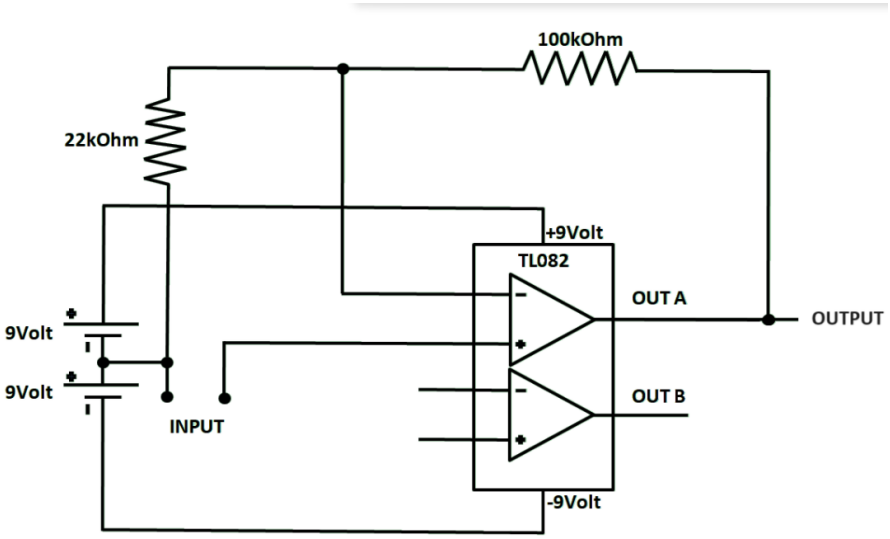
Εικόνα 2.14: Το διάγραμμα συνδέσεως των ακροδεκτών του ολοκληρωμένου κυκλώματος τελεστικών ενισχυτών TL082 [167].

Αφού κατανοήθηκε ο τρόπος συνδεσμολογίας το ολοκληρωμένου κυκλώματος του τελεστικού ενισχυτή, μπορεί να αναλυθεί το κύκλωμα ενίσχυσης του σήματος από το μικρόφωνο επαφής (Εικόνα Εικόνα 2.15). Σύμφωνα λοιπόν με το κύκλωμα από την δημοσίευση της Amanda Ghassaei, γίνεται χρήση του ενός τελεστικού ενισχυτή για την

ενίσχυση του σήματος [127]. Πιο συγκεκριμένα, το σήμα από το μικρόφωνο επαφής χρησιμοποιείται ως είσοδος στην μη-αναστρέφουσα είσοδο του τελεστικού ενισχυτή. Από την άλλη, στην αναστρέφουσα είσοδο του ενισχυτή συνδέεται η έξοδος του τελεστικού ενισχυτή μέσω μίας 100kOhm αντίστασης καθώς και το μηδενικό δυναμικό των δύο 9Volt μπαταριών που είναι συνδεδεμένες εν σειρά μέσω μίας αντίστασης των 22kOhm. Με αυτόν τον τρόπο, έχει καθοριστεί το κέρδος ενίσχυσης του σήματος. Ουσιαστικά, το κέρδος ενίσχυσης του σήματος (Gain) μπορεί να υπολογιστεί σύμφωνα με τον ακόλουθο τύπο:

$$\frac{V_{out}}{V_{in}} = 1 + \frac{R_2}{R_1} \rightarrow \frac{V_{out}}{V_{in}} = 1 + \frac{100kOhm}{22kOhm} \rightarrow \frac{V_{out}}{V_{in}} = 1 + 4.54 \rightarrow \frac{V_{out}}{V_{in}} = 5{,}54$$

Σε αυτό το σημείο αξίζει να σημειωθεί ότι, αντί για ένα ποτενσιόμετρο των 10kOhm που χρησιμοποιείται στο κύκλωμα από την δημοσίευση της Amanda Ghassaei, χρησιμοποιήθηκε μία αντίσταση των 22kOhm ώστε το κέρδος να πέσει στην επιθυμητή τιμή. Σκοπός της συγκεκριμένης εναλλαγής είναι η εξασφάλιση ότι το σήμα εξόδου από τον τελεστικό ενισχυτή δεν υπερβαίνει τα 5Volt. Αυτή είναι η μόνη απαίτηση που έχει το κύκλωμα ενίσχυσης, διότι η αναλογική θύρα του Arduino Mega2560 μπορεί να διαβάσει σήματα από 0Volt έως 5Volt. Εδώ όμως προκύπτει το εξής πρόβλημα, το ενισχυμένο σήμα από το τελεστικό ενισχυτή μπορεί να έχει και αρνητικές τιμές, με αποτέλεσμα η αναλογική είσοδος να μην μπορεί να τις εντοπίσει και να τις αναγνωρίζει ως 0Volt. Επομένως, το ενισχυμένο σήμα θα πρέπει να βρίσκεται μόνιμα πάνω από τα 0Volt και κάτω από τα 5Volt. Συνεπώς, για την αντιμετώπιση του προβλήματος αυτού θα πρέπει να γίνει χρήση ενός κυκλώματος μετατόπισης του σήματος εντός του εύρους 0Volt έως 5Volt [127].



Εικόνα 2.15: Κύκλωμα ενίσχυσης του σήματος από το μικρόφωνο επαφής [168].

Για την δημιουργία του κυκλώματος μετατόπισης, χρησιμοποιήθηκε ένας διαιρέτης τάσης και δύο πυκνωτές (Εικόνα Εικόνα 2.16) [127]. Πιο συγκεκριμένα, το ενισχυμένο σήμα διέρχεται από το εσωτερικό ενός ηλεκτρολυτικού πυκνωτή των 10μF και εν συνεχεία από τον διαιρέτη τάσης με τις δύο αντιστάσεις των 100kOhm συνδεδεμένες εν σειρά. Ο διαιρέτη τάσης συνδέεται μεταξύ των 5Volt και του GND (0Volt) που παρέχει το ίδιο το Arduino Mega2560. Συνεπώς, αφού οι δύο αντιστάσεις του διαιρέτη τάσης έχουν

της ίδια τιμή αντίστασης, τότε η τάση ανάμεσα στις δύο αντιστάσεις είναι στα 2.5Volt. Αυτό αποδεικνύεται σύμφωνα με τους ακόλουθους τύπους:

$$\left.\begin{array}{l} (1)\ I = \frac{V_{in}}{R_1+R_2} \\ (2)\ V_{out} = I \cdot R_2 \end{array}\right\} V_{out} = \frac{V_{in}}{R_1+R_2} \cdot R_2 \rightarrow V_{out} = \frac{R_2}{R_1+R_2} \cdot V_{in}$$

Επομένως, αντικαθιστώντας τις τιμές των αντιστάσεων 100kOhm και της τάσης τροφοδοσίας 5Volt, προκύπτει:

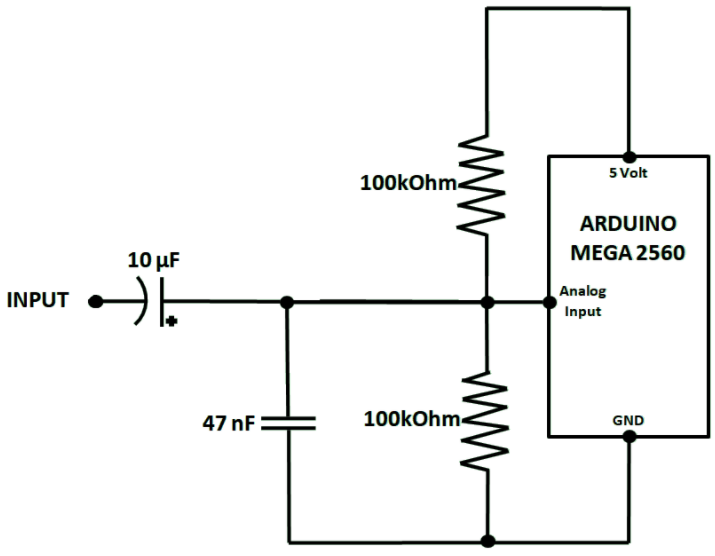
$$V_{out} = \frac{100kOhm}{100kOhm + 100kOhm} \cdot 5Volt \rightarrow$$

$$V_{out} = \frac{100kOhm}{200kOhm} \cdot 5Volt \rightarrow$$

$$V_{out} = \frac{1}{2} \cdot 5Volt \rightarrow$$

$$V_{out} = 2.5Volt$$

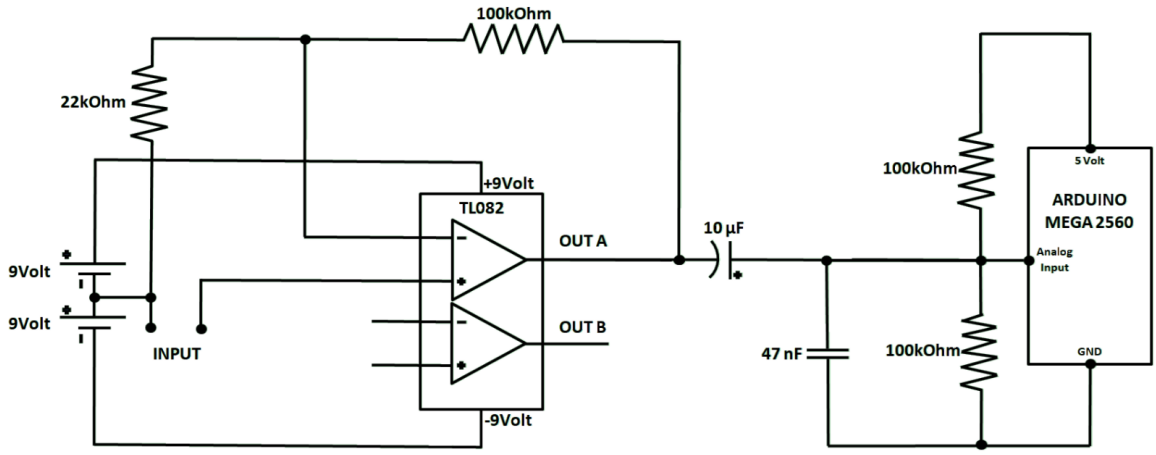
Συνεπώς, όταν η τάση του πυκνωτή από την πλευρά του ενισχυτή φορτίζεται και εκφορτίζεται, τότε στο άλλο άκρο του πυκνωτή προκαλείται στιγμιαία συσσώρευση και απώθηση των 2.5Volt που παράγονται από τον διαιρέτη τάσης. Αυτό έχει ως αποτέλεσμα η τάση εξόδου το διαιρέτη τάσης να ταλαντώνεται με κέντρο τα 2.5Volt. Επομένως, η αναλογική θύρα του Arduino Mega2560 μπορεί τώρα να εντοπίσει την μεταβολή του σήματος. Επιπλέον, ένας ακόμα πυκνωτής των 47nF συνδέεται μεταξύ των 2.5Volt και του GND.



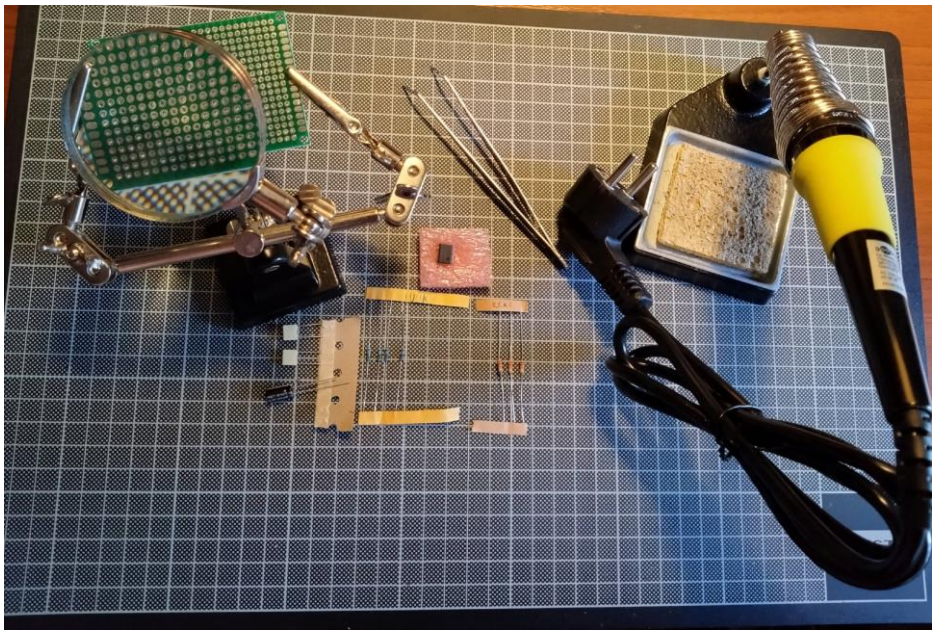
Εικόνα 2.16: Το κύκλωμα μετατόπισης του ενισχυμένου σήματος μεταξύ 0Volt και 5Volt [169].

Επομένως, συνδέοντας τα δύο κυκλώματα μαζί, δημιουργείται το επιθυμητό κύκλωμα ενίσχυσης και μετατόπισης του σήματος που προκύπτει από μικρόφωνο επαφής (Εικόνα Εικόνα 2.17) [127]. Ωστόσο, για την υλοποίηση της πλακέτας ενίσχυσης και

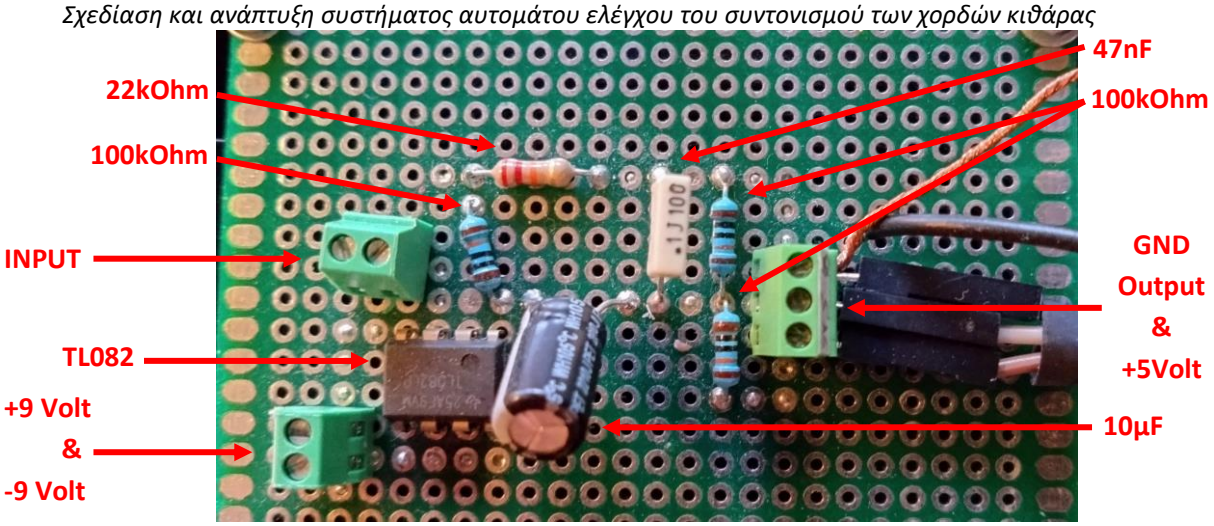
μετατόπισης του σήματος, χρησιμοποιήθηκαν εργαλεία όπως, κολλητήρι, βάση στηρίξεως κολλητηριού καλάι, μία βάση στηρίξεως πλακέτας (PCB) με μεγεθυντικό φακό και τσιμπίδα (Εικόνα Εικόνα 2.18). Η πλακέτα που κατασκευάστηκε είναι αρκετά μικρή και μπορεί να ενσωματωθεί επάνω στην κεφαλή της κιθάρας (Εικόνα Εικόνα 2.19).



Εικόνα 2.17: Το ολοκληρωμένο κύκλωμα ενίσχυσης και μετατόπισης του σήματος από το μικρόφωνο επαφής [170].



Εικόνα 2.18: Τα εργαλεία και τα υλικά για την υλοποίηση της πλακέτας ενίσχυσης και μετατόπισης του σήματος [171].

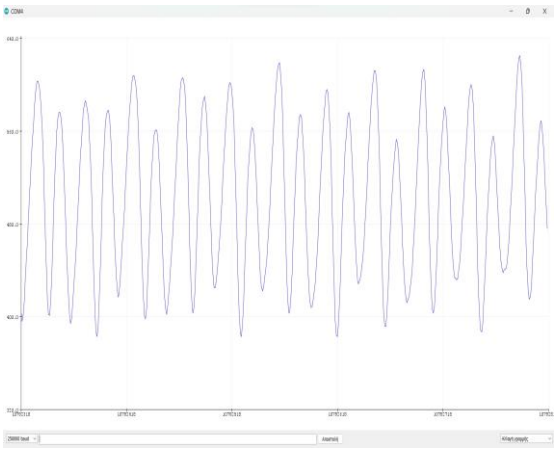


Εικόνα 2.19: Η πλακέτα ενίσχυσης και μετατόπισης του σήματος από το μικρόφωνο επαφής [172].

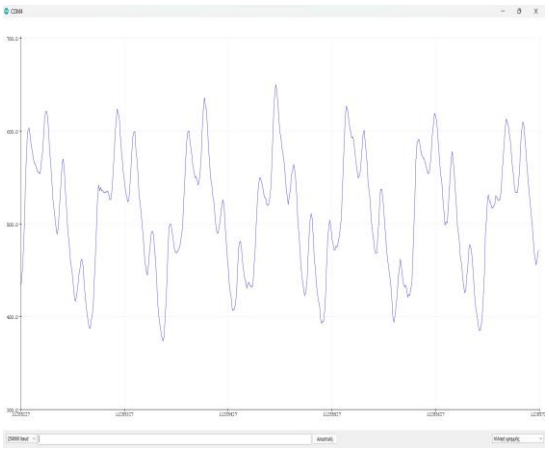
Μετά την ολοκλήρωση της πλακέτας, συνδέθηκε σε αυτή το μικρόφωνο επαφής για την λήψη του σήματος εισόδου, η αναλογική θύρα που θα διαβάζει το σήμα εξόδου από την πλακέτα και οι αντίστοιχες τροφοδοσίες. Επομένως, μέσω της πλακέτας αυτής το Arduino Mega 2560 είναι σε θέση να διαβάσει τα σήματα από το μικρόφωνο επαφής. Εντελώς δοκιμάστηκα, δημιουργήθηκε ένας απλός κώδικας για την ανάγνωση του ενισχυμένου σήματος από το μικρόφωνο επαφής (Εικόνα 2.12). Μέσω του σχεδιογράφου σειριακής που παρέχει το περιβάλλον προγραμματισμού Arduino IDE εμφανίζονται οι κυματομορφές όλων των χορδών της κιθάρας διεγείροντας την κάθε μία μεμονωμένα (Εικόνα Εικόνα 2.20).

Πίνακας 2.1: Απλός κώδικας για την ανάγνωση του ενισχυμένου σήματος από το μικρόφωνο επαφής [17].

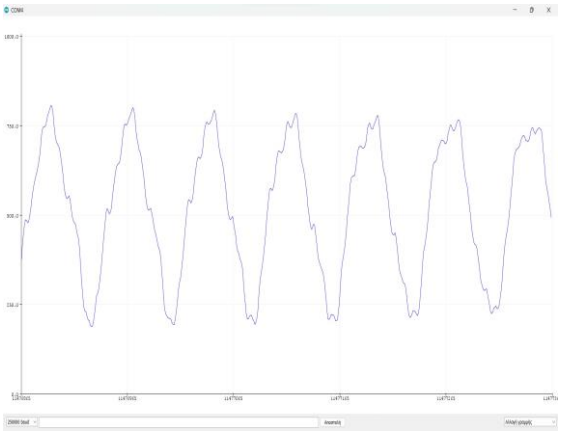
Κώδικας προγραμματισμού	Ανάλυση κώδικα
void setup() {	Η συνάρτηση setup() η οποία τρέχει τις εντολές μόνο μία φορά μόλις ενεργοποιηθεί το Arduino.
Serial.begin(2000000);	Ενεργοποίηση της σειριακής επικοινωνίας και καθορισμός του Baud Rate στα 2000000bps.
pinMode(A15, INPUT); }	Ορισμός της αναλογικής θύρας A15 ως θύρα εισόδου.
void loop() {	Η συνάρτηση loop() η οποία εκτελείται ξανά μόλις ολοκληρωθεί.
Serial.println(analogRead(A15)); }	Εκτύπωση των δεδομένων που διαβάζει κάθε φορά η αναλογική θύρα στην σειριακή οθόνη.



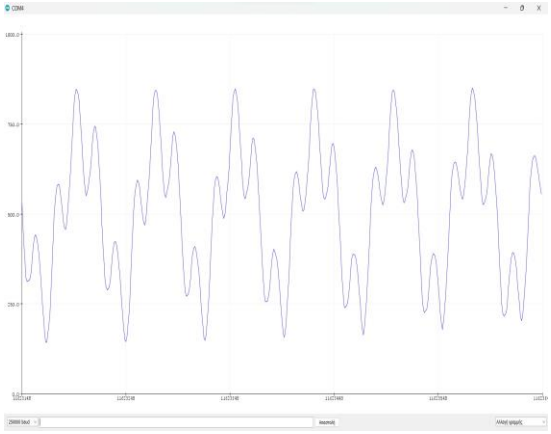
α)



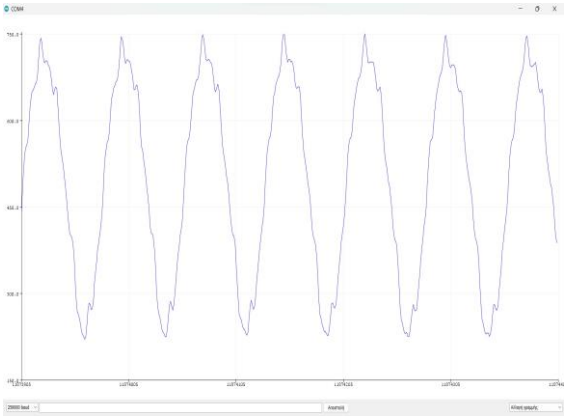
β)



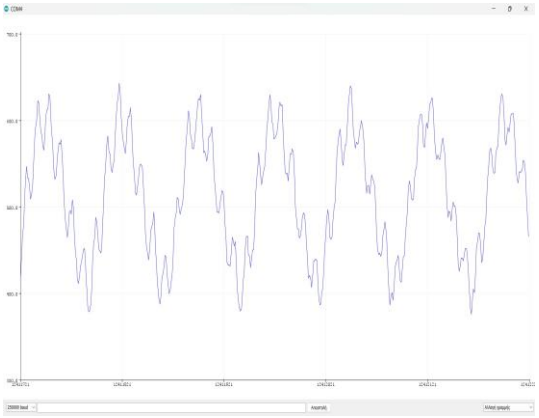
γ)



δ)



ε)



στ)

Εικόνα 2.20: Οι κυματομορφές των χορδών της κιθάρας, α) Μι μπάσο (E), β) Λα (A), γ) Ρε (D), δ) Σολ (G), ε) Σι (B) & στ) Μι καντίνι (e) [173].

Μετά την δοκιμή και την εμφάνιση των αποτελεσμάτων για το σήμα που παράγεται όταν διεγείρεται κάθε μία χορδή ξεχωριστά, δημιουργήθηκε ένας άλλος κώδικας σύμφωνα με την δημοσίευση της Amanda Ghassaei (Πίνακας 2.2) [128] [129]. Πιο συγκεκριμένα, ο κώδικας αυτός αποσκοπεί όχι μόνο στην σύλληψη του ενισχυμένου σήματος από το μικρόφωνο επαφής, αλλά και στον υπολογισμό της θεμελιώδης συχνότητας με την οποία

ταλαντώνετε κάθε μία χορδή. Επίσης, στην περίπτωση που το σήμα ψαλιδίζεται, δηλαδή υπερβαίνει τα 5Volt ή πέφτει κάτω από τα 0Volt, τότε ενεργοποιείται ένα LED ως ένδειξη προς τον χρήστη. Ακόμα, δεν χρησιμοποιείται η συνάρτηση analogRead() διότι εκτελεί αρκετές εντολές με αποτέλεσμα να ελαττώνεται η ταχύτητα δειγματοληψίας. Επιπροσθέτως, στο πρόγραμμα χρησιμοποιείται και η ψηφιακή θύρα No12 ως έξοδος η οποία δίνει έξοδο κάθε φορά που η το σήμα ξεπερνά τα 2.5Volt με θετική κλίση. Ωστόσο, στην συγκεκριμένη διπλωματική εργασία η θύρα No12 καταργήθηκε διότι είναι περιττή η χρήση της.

Πίνακας 2.2: Κώδικας μετατροπής της ανάλυσης δειγματοληψίας σε 8bit καθώς, σύλληψης του σήματος από το μικρόφωνο επαφής και υπολογισμός της θεμελιώδης συχνότητας [18].

Κώδικας προγραμματισμού	**Ανάλυση κώδικα**
boolean clipping = 0;	Δημιουργία και δήλωση της boolean μεταβλητής clipping με την τιμή 0.
byte newData = 0; byte prevData = 0;	Δημιουργία και δήλωση των byte μεταβλητών newData & prevData με την τιμή 0.
unsigned int time = 0;	Δημιουργία και δήλωση της unsigned int μεταβλητής time με την τιμή 0.
int timer[10]; int slope[10];	Δημιουργία δύο μονοδιάστατων πινάκων 10 στοιχείων τύπου integer με ονόματα timer και slope.
unsigned int totalTimer; unsigned int period;	Δημιουργία δύο μεταβλητών τύπου unsigned integer totalTimer και period.
byte index = 0;	Δημιουργία και δήλωση της byte μεταβλητής index με την τιμή 0.
float frequency;	Δημιουργία μεταβλητής τύπου float frequency.
int maxSlope = 0; int newSlope;	Δημιουργία δύο μεταβλητών τύπου integer maxSlope και newSlope και δήλωση της πρώτης με τιμή 0.
byte noMatch = 0; byte slopeTol = 3;	Δημιουργία και δήλωση των byte μεταβλητών noMatch = 0 και slopeTol = 3.
int timerTol = 10;	Δημιουργία και δήλωση της integer μεταβλητής με την τιμή 10.

unsigned int ampTimer = 0;	Δημιουργία και δήλωση της unsigned integer μεταβλητής ampTimer με την τιμή 0.

byte maxAmp = 0;
byte checkMaxAmp;
byte ampThreshold = 30;
	Δημιουργία τριών μεταβλητών byte maxAmp, checkMaxAmp και ampThreshold και δήλωση της maxAmp με την τιμή 0 και της ampThreshold με την τιμή 30.

void setup(){	Η συνάρτηση setup(), η οποία εκτελείτε μία φορά κατά το ξεκίνημα.
Serial.begin(9600);	Ενεργοποίηση της σειριακής επικοινωνίας και ορισμός του baud rate στα 9600bps.
pinMode(13,OUTPUT); pinMode(12,OUTPUT);	Καθορισμός των ψηφιακών θυρών No13 και No12 ως θύρες εξόδου.
cli();	Απενεργοποίηση διακοπών
ADCSRA = 0; ADCSRB = 0;	Μηδενισμός των bit των καταχωρητών ADCSRA και ADCSRB.
ADMUX |= (1 << REFS0); ADMUX |= (1 << ADLAR);	Ορισμός του bit REFS0 με 1 στον καταχωρητή ADMUX για καθορισμό της τάσης αναφοράς. Ορισμός του bit ADLAR με 1 στον καταχωρητή ADMUX για αριστερή στοίχιση της τιμής στον καταχωρητή ADC. Έτσι χρειάζεται μόνο να διαβαστούν τα 8 πρώτα bit από τον καταχωρητή ADCH.
ADCSRA |= (1 << ADPS2) | (1 << ADPS0); ADCSRA |= (1 << ADATE); ADCSRA |= (1 << ADIE); ADCSRA |= (1 << ADEN); ADCSRA |= (1 << ADSC);	Καθορισμός bit στον καταχωρητή ADCSRA. Ορισμός ADPS2 & ADPS0 με 1 για τον καθορισμό του αναλογικού-ψηφιακού ρολογιού με προκλιμάκωση στα 32. Ορισμός του ADATE με 1 για την αυτόματη ενεργοποίηση του αναλογικού-ψηφιακού μετατροπέα. Ορισμός του ADIE με 1 για την ενεργοποίηση των διακοπών μετά την ολοκλήρωση των μετρήσεων. Ορισμός του ADEN με 1 για την ενεργοποίηση του αναλογικό-

	ψηφιακού μετατροπέα. Ορισμός του ADSC με 1 για το ξεκίνημα της μέτρηση του αναλογικό-ψηφιακού μετατροπέα.
sei(); }	Ενεργοποίηση διακοπών.
ISR(ADC_vect) {	Η ρουτίνα της διακοπής η οποία εκτελείτε όταν υπάρχει νέα τιμή στον καταχωρητή ADCH.
PORTB &= B11101111; prevData = newData;	Μηδενισμός της θύρας εξόδου No12. Αποθήκευση της παλιάς τιμής της μεταβλητής newData στην μεταβλητή prevData.
newData = ADCH;	Αποθήκευση της νέας τιμής από την δειγματοληψία στον αναλογικό ακροδέκτη A0 στην μεταβλητή newData.
if (prevData < 127 && newData >=127){	Αν η τιμή της μεταβλητής prevData είναι μικρότερη από 127 και η τιμή της μεταβλητής newData είναι μεγαλύτερη ή ίση με 127, τότε το σήμα αυξάνεται και ξεπερνά το μεσαίο σημείο.
newSlope = newData - prevData;	Έπειτα, υπολογίζεται η κλίση του σήματος η οποία είναι η διαφορά μεταξύ των μεταβλητών newData και prevData και αποθηκεύεται στην μεταβλητή newSlope .
if (abs(newSlope-maxSlope)<slopeTol){	Αν η απόλυτη τιμή της διαφοράς των μεταβλητών newSlope και maxSlope είναι μικρότερη από την μεταβλητή slopeTol, τότε:
slope[index] = newSlope;	Η μεταβλητή newSlope αποθηκεύεται στον πίνακα slope στην θέση που δείχνει ο δείκτης index.
timer[index] = time;	Η μεταβλητή time αποθηκεύεται στον πίνακα timer στην θέση που δείχνει ο δείκτης index.

time = 0;	Μηδενισμός της μεταβλητής time.
if (index == 0){	Αν η τιμή του δείκτη index είναι ίση με το μηδέν.

PORTB |= B00010000;
noMatch = 0;
index++;
}
	Ενεργοποίηση της θύρας No12.
	Μηδενισμός της μεταβλητής noMatch.
	Αύξηση του δείκτη index κατά 1.

else if (abs(timer[0]-timer[index])<timerTol && abs(slope[0]-newSlope)<slopeTol){	Αλλιώς αν η απόλυτη τιμή της διαφοράς της πρώτης και της τρέχουσας τιμής (index) του πίνακα timer είναι μικρότερη από την τιμή της μεταβλητής timerTol και η απόλυτη τιμή της διαφοράς της πρώτης τιμής του πίνακα slope με την μεταβλητή newSlope είναι μικρότερη από την τιμή της μεταβλητής slopeTol, τότε η διάρκεια του χρόνου και η κλίση ταιριάζουν με τα επιθυμητά.
totalTimer = 0;	Τότε, μηδενίζεται η μεταβλητή totalTimer.

for (byte i=0;i<index;i++){
totalTimer+=timer[i];
}
	Για όσο ο μετρητής i έχει μικρότερη τιμή από τον δείκτη index, ξεκινώντας από το 0 και αυξάνοντας κατά 1, τότε στην μεταβλητή totalTimer αποθηκεύεται το άθροισμα των τιμών του πίνακα timer.

period = totalTimer;	Αποθήκευση της μεταβλητής totalTimer στην μεταβλητή period.
timer[0] = timer[index];	Αποθήκευση της τιμής του στοιχείου του πίνακα timer της θέσης index στην θέση 0.
slope[0] = slope[index];	Αποθήκευση της τιμής του στοιχείου του πίνακα slope της θέσης index στην θέση 0.
index = 1;	Αποθήκευση της τιμής 1 στην μεταβλητή index.

PORTB |= B00010000;
noMatch = 0;
}
	Ενεργοποίηση εξόδου της θύρας No12.
	Μηδενισμός της μεταβλητής noMatch.

else{	Αλλιώς, αυξάνεται ο δείκτης index

index++;
if (index > 9){
reset();
}
}
}
	κατά 1. Αν η τιμή του δείκτη index είναι μεγαλύτερη από 9, τότε καλείτε η συνάρτηση reset().

else if (newSlope>maxSlope){	Αλλιώς αν η τιμή της μεταβλητής newSlope είναι μεγαλύτερη από αυτή της μεταβλητής maxSlope, τότε:
maxSlope = newSlope;	Η τιμή της μεταβλητής newSlope αποθηκεύεται στην μεταβλητή maxSlope.

time = 0;
noMatch = 0;
index = 0;
}
	Μηδενισμός των μεταβλητών time, noMatch και index.

else{
noMatch++;
if (noMatch>9){
reset();
}
}
}
	Αλλιώς, η μεταβλητή noMatch αυξάνεται κατά 1. Αν η μεταβλητή noMatch υπερβεί την τιμή 9, τότε καλείτε η συνάρτηση reset().

if (newData == 0 || newData == 1023){
PORTB |= B00100000;
clipping = 1;
}
	Αν η τιμή της μεταβλητής newData είναι ίση με το 0 ή ίση με 1023, τότε ενεργοποίηση της θύρας εξόδου No13 και αποθήκευση της τιμής 1 στην flag μεταβλητή clipping. Δηλαδή, ειδοποίηση για ψαλίδισμα του σήματος.

time++;	Αύξηση της μεταβλητής time κατά 1.
ampTimer++;	Αύξηση της μεταβλητής ampTimer κατά 1.

if (abs(127-ADCH)>maxAmp){
maxAmp = abs(127-ADCH);
}
	Αν η απόλυτη τιμή της διαφοράς μεταξύ της τιμής 127 και της μεταβλητής ADCH είναι μεγαλύτερη από την μεταβλητή maxAmp,τότε η απόλυτη τιμή της διαφοράς αποθηκεύεται στην μεταβλητή maxAmp.

	Αν η τιμή της μεταβλητής ampTimer

if (ampTimer==1000){
ampTimer = 0;
checkMaxAmp = maxAmp;
maxAmp = 0;
}
 	
}
	είναι ίση με 1000, τότε μηδενίζεται η τιμή της μεταβλητής ampTimer, η τιμή της μεταβλητής maxAmp αποθηκεύεται στην μεταβλητή checkMaxAmp και μετά η μεταβλητή maxAmp μηδενίζεται.

void reset(){
index = 0;
noMatch = 0;
maxSlope = 0;
}
	Ορισμός της συνάρτησης reset().
	Μηδενισμός του δείκτη index και των μεταβλητών noMatch και maxSlope.

void checkClipping(){
if (clipping){
PORTB &= B11011111;
clipping = 0;
}
}
	Ορισμός της συνάρτησης checkClipping(). Αν η flag μεταβλητή clipping δεν είναι μηδέν, έχει δηλαδή οποιαδήποτε άλλη τιμή, τότε μηδενίζεται η θύρα εξόδου No13 και η flag μεταβλητή clipping.

void loop(){
 	
checkClipping();
 	
if (checkMaxAmp>ampThreshold){
 	
frequency = 38462/float(period);
 	
Serial.print(frequency);
Serial.println(" hz");
}
 	
delay(100);
 	
}
	Ορισμός της συνάρτησης loop()
	
	Κλήση της συνάρτησης checkClipping().
	Αν η μεταβλητή checkMaxAmp είναι μεγαλύτερη από ampThreshold, τότε γίνεται ο υπολογισμός της θεμελιώδης συχνότητας. Πιο συγκεκριμένα, το πηλίκο από την διαίρεση της τιμής 38462 με την τιμή της μεταβλητής period, η οποία έχει γίνει προς στιγμήν μεταβλητή float, είναι η συχνότητα ταλάντωσης της χορδής και αποθηκεύεται στην μεταβλητή frequency.
	Εκτύπωση της τιμής της μεταβλητής frequency, ή αλλιώς της θεμελιώδης συχνότητας στην σειριακή οθόνη καθώς και της μονάδας μέτρησης (hz).
	Καθυστέρηση 100msec.

Επειδή όμως το Arduino Mega 2560 έχει χαμηλή συχνότητα δειγματοληψίας, θα πρέπει να γίνει μείωση των bit ανάλυσης. Με αυτόν τον τρόπο αυξάνεται η συχνότητα δειγματοληψίας, αλλά μειώνεται η ανάλυση. Σύμφωνα με τον κώδικα από την δημοσίευση της Amanda Ghassaei γίνεται μείωση των bit ανάλυσης προγραμματιστικά στον μικροελεγκτή από τα 10bit στα 8bit. Με αυτόν τον τρόπο οι στάθμες που χωρίζουν το εύρος 0Volt έως 5Volt από 2^{10}=1024 στάθμες μειώνονται σε 2^{8}=256 στάθμες. Επίσης, οι στάθμες επηρεάζουν με την σειρά τους το μέγιστο σφάλμα δειγματοληψίας. Πιο συγκεκριμένα, αν χωριστεί το εύρος 0Volt έως 5Volt σε 1024 στάθμες, τότε:

$$5\text{Volt} \div 1024 = 4.88\text{mVolt}$$

Δηλαδή, η κάθε στάθμη απέχει από την επόμενη και από την προηγούμενη κατά 4.88mVolt. Επομένως, το μέγιστο σφάλμα που μπορεί να προκύψει είναι όταν το αναλογικό σήμα δειγματοληπτιθεί στο μισό των δύο σταθμών. Σε αυτή την περίπτωση, ο αναλογικό-ψηφιακός μετατροπέας θα πρέπει να κατατάξει το σήμα είτε στην μία στάθμη είτε στην άλλη. Έτσι, στην περίπτωση που η ανάλυση είναι 10bit και κατά συνέπεια υπάρχουν 1024 στάθμες, το μέγιστο σφάλμα είναι:

$$4.88\text{mVolt} \div 2 = 2.44\text{mVolt}$$

Στην περίπτωση όμως, που η ανάλυση είναι 8bit και οι στάθμες 256 τότε το εύρος 0Volt έως 5Volt μπορεί να χωριστεί ανά:

$$5\text{Volt} \div 256 = 19.53\text{mVolt}$$

Δηλαδή, η κάθε στάθμη σε αυτήν την περίπτωση απέχει από την επόμενη και από την προηγούμενη κατά 19.53mVolt. Επομένως, σε αυτή την περίπτωση του μέγιστο σφάλμα που μπορεί να προκύψει είναι όταν πραγματοποιηθεί δειγματοληψία στα μισά δύο σταθμών. Έτσι στην περίπτωση όπου η ανάλυση είναι στα 8bit και υπάρχουν 256 στάθμες, το μέγιστο σφάλμα είναι:

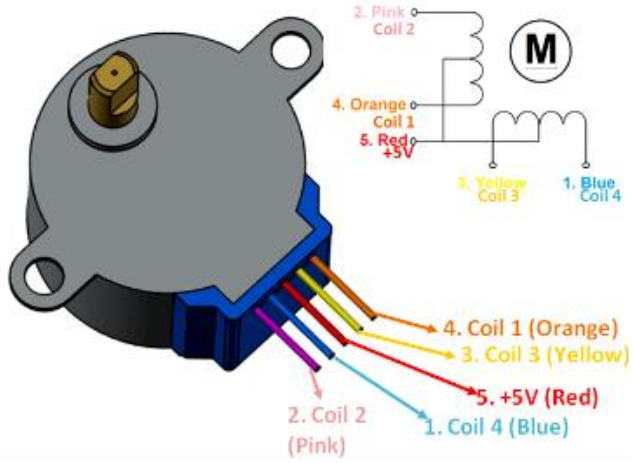
$$19.53\text{mVolt} \div 2 = 9.76\text{mVolt}$$

2.2.4 **Ο βηματικός κινητήρας 28BYJ-48 και ο οδηγός ULN2003**

Ένα από τα βασικότερα σημεία του συστήματος αυτομάτου ελέγχου για τον συντονισμό των χορδών κιθάρας, είναι ο τύπος κινητήρα ο οποίος ευθύνεται για την περιστροφή των κλειδιών του οργάνου και κατά συνέπεια το κούρδισμα των χορδών. Για την συγκεκριμένη εργασία, όπως αναφέρθηκε προηγουμένως, επιλέχθηκε ο βηματικός κινητήρας 28BYJ-48 (Εικόνα Εικόνα 2.21) [130]. Σε αυτήν την ενότητα λοιπόν, αναλύεται ο τρόπος κατασκευής και λειτουργίας του συγκεκριμένου βηματικού κινητήρα καθώς και της πλακέτα οδήγησής του.

Εικόνα 2.21: Ο βηματικός κινητήρας 28BYJ-48 [174].

Αρχικά, αυτού του τύπου οι κινητήρες, αποτελούνται από τέσσερα πηνία στον στάτη, ενώ ο ρότορας τους αποτελείται από μαγνήτες. Επίσης, οι συγκεκριμένοι βηματικοί κινητήρες, παρόλο που χρειάζονται DC τροφοδοσία για να λειτουργήσουν δεν περιλαμβάνουν συλλέκτη και ψήκτρες όπως οι απλοί DC κινητήρες. Ουσιαστικά, ο συγκεκριμένος βηματικός κινητήρας αποτελείται από μία μονοπολική διάταξη τεσσάρων πηνίων και χρειάζεται τροφοδοσία 5Volt (Εικόνα Εικόνα 2.22) [131]. Ωστόσο, παρόλο που χρειάζεται 5Volt τροφοδοσία για να λειτουργήσει ο βηματικός κινητήρας, παράγει μια ικανοποιητική ροπή η οποία μπορεί να περιστρέψει τα κλειδιά της κιθάρας. Η ροπή που μπορεί να ασκήσει ο συγκεκριμένος κινητήρας είναι μεγαλύτερη από 34.3mN·m [132].



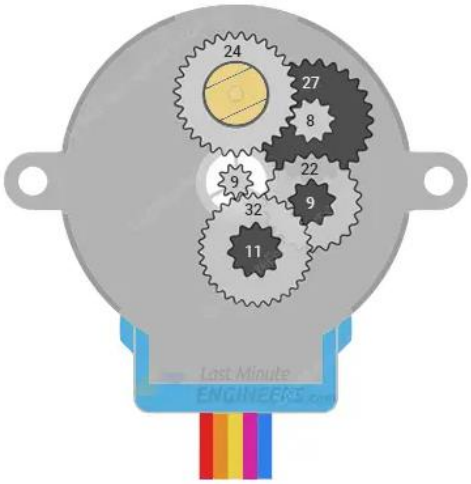
Εικόνα 2.22: Η διαμόρφωση των καλωδίων και ο τρόπος σύνδεσης των πηνίων του βηματικού κινητήρα 28BYJ-48 [175].

Επίσης, ο συγκεκριμένος βηματικού κινητήρας μπορεί να ρυθμιστεί σε λειτουργία μισού βήματος. Δηλαδή, θα πρέπει να κάνει 64 βήματα σύνολο, με το κάθε βήμα περιστροφής που πραγματοποιείται να είναι 5.625^{o} για να κάνει μια πλήρης περιστροφή 360^{o}. Στην περίπτωση όμως που είναι επιθυμητή η περιστροφή του άξονα του βηματικού κινητήρα σύμφωνα με την λειτουργία πλήρους βήματος, το κάθε βήμα θα πραγματοποιεί διπλάσια περιστροφή από την λειτουργία μισού βήματος. Δηλαδή, στην λειτουργία πλήρους βήματος, θα πρέπει να γίνουν 32 βήματα σύνολο, με το κάθε βήμα περιστροφής να είναι 11.25^{o} για να κάνει μια πλήρης περιστροφή 360^{o}. Επιπλέον, κατασκευαστικά ο βηματικός κινητήρας 28BYJ-48 περιλαμβάνει ένα σύστημα γραναζιών των οποίων η αναλογία είναι 64:1 (Εικόνα Εικόνα 2.23) [133] [134] [135]. Συνεπώς, στην λειτουργία μισού βήματος, ο βηματικός κινητήρας για μία πλήρης περιστροφή θα πρέπει να κάνει:

64 (gear ratio) x 64 (steps) = 4096 (steps)

Στην λειτουργία πλήρους βήματος, ο βηματικός κινητήρας για μία πλήρης περιστροφή θα πρέπει να κάνει:

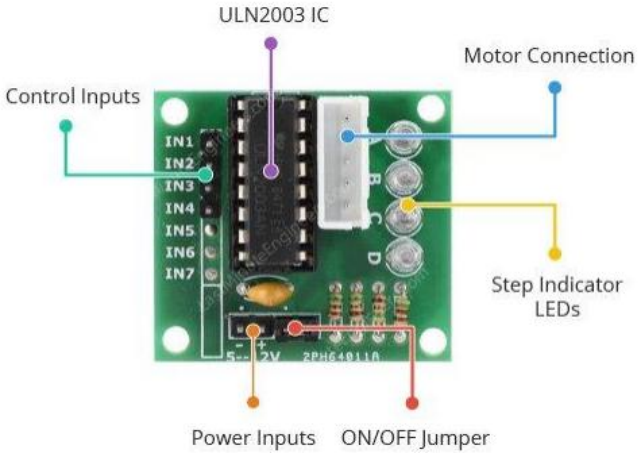
64(gear ratio) x 32(steps) = 2048 (steps)



Εικόνα 2.23: Το σύστημα γραναζιών του βηματικού κινητήρα με αναλογία 64:1 [176].

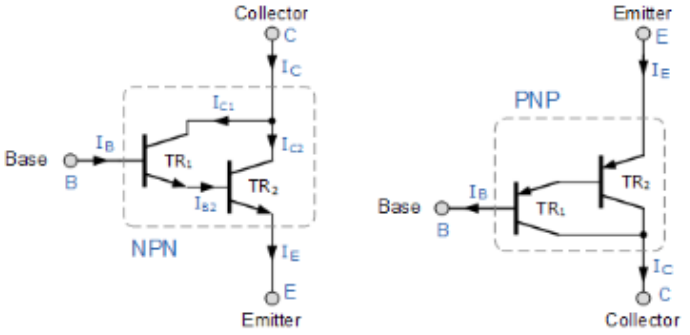
Επομένως, ανάλογα με την λειτουργία του βηματικού κινητήρα, αλλάζει και ο αριθμός των βημάτων που πρέπει να γίνουν ώστε ο άξονας του βηματικού κινητήρα να πραγματοποιήσει μια πλήρης περιστροφή. Επίσης, η λειτουργία μισού βήματος παρόλο που χρειάζεται περισσότερα βήματα για μία πλήρης περιστροφή παρέχει μεγαλύτερη ακρίβεια σε σχέση με την λειτουργία πλήρους βήματος. Από την άλλη, στην λειτουργία πλήρους βήματος, παρόλο που χρειάζονται λιγότερα βήματα για μία πλήρης περιστροφή παρέχεται μεγαλύτερη ταχύτητα σε σχέση με την λειτουργία μισού βήματος.

Για να μπορεί να ελεγχθεί καλύτερα και πιο εύκολα η περιστροφή του βηματικού κινητήρα, γίνεται χρήση της πλακέτας οδήγησης ULN2003 (Εικόνα Εικόνα 2.24) [136]. Η πλακέτα οδήγησης αυτή περιλαμβάνει ένα ολοκληρωμένο κύκλωμα (IC) αποτελούμενο από επτά ζευγάρια τρανζίστορ τύπου Darlington, όπου κάθε ζευγάρι μπορεί να οδηγήσει φορτία μέχρι 50V και 500mA. Για την δημιουργία ζευγαριών τρανζίστορ Darlington, θα πρέπει να γίνει χρήση δύο διπολικών τρανζίστορ όπου ο εκπομπός του πρώτου τρανζίστορ συνδέεται στην βάση του δεύτερου τρανζίστορ, ενώ οι συλλέκτες των δύο τρανζίστορ συνδέονται μαζί (Εικόνα Εικόνα 2.25) [138] [139].

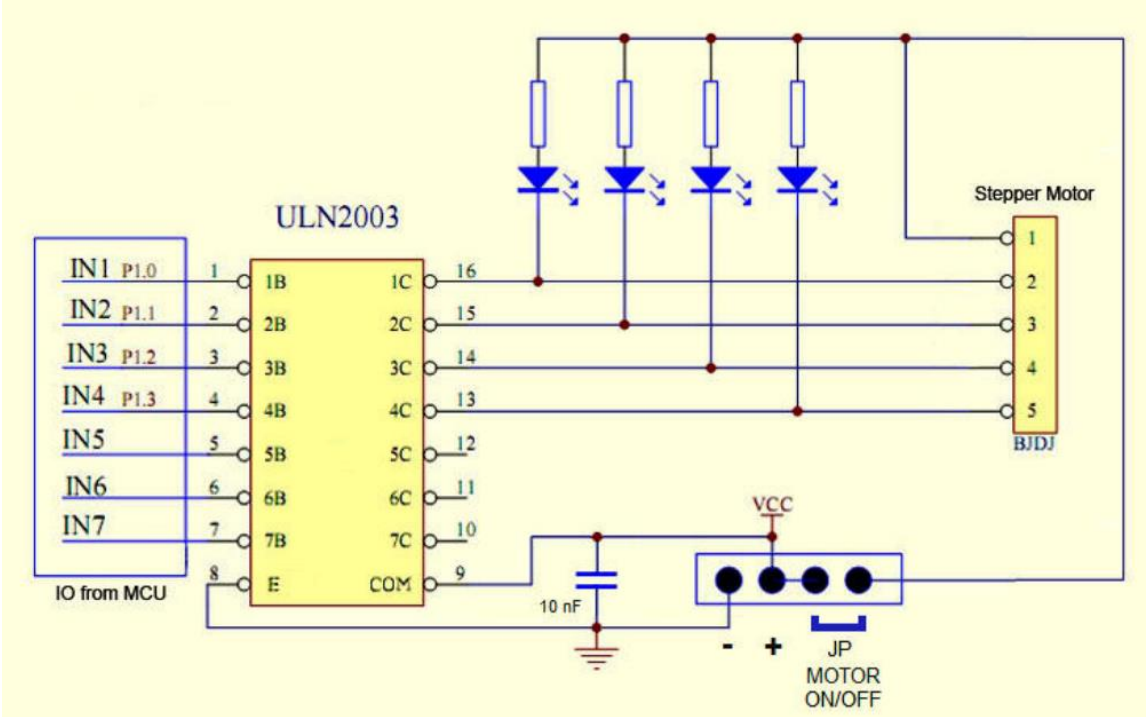


Εικόνα 2.24: Η πλακέτα οδήγησης ULN2003 του βηματικού κινητήρα [177].

Με αυτόν τον τρόπο, το ρεύμα που ενισχύεται από το πρώτο τρανζίστορ μπορεί να ενισχυθεί περισσότερο από το δεύτερο τρανζίστορ. Στην συγκεκριμένη πλακέτα οδήγησης, χρησιμοποιούνται μόνο τα τέσσερα από τα επτά ζευγάρια τρανζίστορ τύπου Darlington. Επιπλέον, η πλακέτα οδήγησης περιλαμβάνει τέσσερις αρσενικούς ακροδέκτες εισόδου, οι οποίοι καθορίζουν την φορά και την ταχύτητα περιστροφής του άξονα και συνδέονται με τέσσερις ψηφιακές εξόδους του Arduino. Ακόμα περιλαμβάνει, φύσα για την σύνδεση του κινητήρα, ακροδέκτες τροφοδοσίας της πλακέτας από 5V έως 12Volt, ένα βραχυκυκλωτήρα ο οποίος όταν αφαιρεθεί-απομονώνει την τροφοδοσία από τον κινητήρα και τέσσερα Led τα οποία ανάβουν ανάλογα με το ποια είσοδος έχει ενεργοποιηθεί (Εικόνα Εικόνα 2.26) [137]. Για την υλοποίηση της συγκεκριμένης εργασίας, η τροφοδοσία της πλακέτας οδήγησης του βηματικού κινητήρα γίνεται με εξωτερική πηγή και πιο συγκεκριμένα με δύο μπαταρίες των 3.5Volt και όχι μέσω των 5Volt που παρέχει το Arduino. Με αυτό τον τρόπο αποφεύγεται ο ηλεκτρικός θόρυβος από τον κινητήρα που μπορεί να προκαλέσει ζημιά στο Arduino.



Εικόνα 2.25: Ζευγάρια τρανζίστορ Darlington NPN & PNP [178].



Εικόνα 2.26: Το ηλεκτρονικό κύκλωμα της πλακέτας οδήγησης ULN2003 για τον βηματικό κινητήρα [179].

Ωστόσο, υπάρχουν τρεις περιπτώσεις περιστροφής του άξονα του βηματικού κινητήρα μέσω της πλακέτας οδήγησής του. Η πρώτη περίπτωση είναι η κανονική κίνηση του κινητήρα, η οποία παρέχει την μέγιστη ροπή, αλλά έχει την μέγιστη κατανάλωση ενέργειας. Σε αυτή λοιπόν την περίπτωση, τα πηνία του βηματικού κινητήρα ενεργοποιούνται ανά ζεύγη (πίνακας 2.3) [134] [135]. Σύμφωνα με τον ακόλουθο πίνακα, ακολουθώντας τα βήματα ένα έως τέσσερα, ο βηματικός κινητήρας περιστρέφει τον άξονά του προς την μία κατεύθυνση. Σε περίπτωση που είναι επιθυμητή η περιστροφή προς την αντίθετη κατεύθυνση θα πρέπει τα πηνία να ενεργοποιηθούν από το τέταρτο βήμα προς το πρώτο.

Πίνακας 2.3: Η σειρά ενεργοποίησης των πηνίων για την επίτευξη της κανονικής κίνησης του κινητήρα[19].

Βήμα	Πηνίο Νº1	Πηνίο Νº 2	Πηνίο Νº3	Πηνίο Νº4
1	High	High	Low	Low
2	Low	High	High	Low
3	Low	Low	High	High
4	High	Low	Low	High

Η δεύτερη περίπτωση περιστροφής του άξονα είναι η κίνηση κύματος ή ολόκληρου βήματος. Σε αυτήν την περίπτωση, παρέχεται μια μέτρια ροπή με μια μέτρια κατανάλωση. Για την επίτευξη της κίνησης αυτής ενεργοποιείται κάθε φορά ένα από τα τέσσερα πηνία του βηματικού κινητήρα (πίνακας 2.4) [137]. Όπως και πριν, έτσι και εδώ ακολουθώντας τα βήματα ένα έως τέσσερα ο βηματικός κινητήρας περιστρέφει τον άξονά του προς την μία κατεύθυνση. Συνεπώς, για να περιστραφεί ο άξονας προς την αντίθετη κατεύθυνση, θα πρέπει τα πηνία να ενεργοποιηθούν από το τέταρτο βήμα προς το πρώτο.

Πίνακας 2.4: Η σειρά ενεργοποίησης των πηνίων για την επίτευξη της κίνησης κύματος ή αλλιώς του ολόκληρου βήματος [20].

Βήμα	Πηνίο Νº1	Πηνίο Νº 2	Πηνίο Νº3	Πηνίο Νº4
1	High	Low	Low	Low
2	Low	High	Low	Low
3	Low	Low	High	Low
4	Low	Low	Low	High

Η τρίτη και τελευταία περίπτωση περιστροφής του άξονα του βηματικού κινητήρα είναι η κίνηση μισού βήματος. Για την επίτευξη της περίπτωσης αυτής, λαμβάνονται στοιχεία από τις δύο προηγούμενες περιπτώσεις περιστροφής του άξονα του βηματικού κινητήρα. Πιο συγκεκριμένα, στο πρώτο βήμα ενεργοποιείται ένα πηνίο, στο δεύτερο βήμα ενεργοποιούνται δύο πηνία, ενώ στο τρίτο βήμα ενεργοποιείται ένα πηνίο. Η

ενεργοποίηση των πηνίων συνεχίζει με αυτό το μοτίβο (πίνακας 2.5) [135]. Η ροπή που παράγεται όπως και η ενέργεια που καταναλώνεται σε αυτή την περίπτωση είναι ο μέσος όρος από τις δύο προηγούμενες περιπτώσεις. Επίσης, και σε αυτή την περίπτωση ακολουθώντας τα βήματα από το ένα έως το οκτώ, ο βηματικός κινητήρας περιστρέφει τον άξονά του προς την μία κατεύθυνση. Στην περίπτωση που είναι επιθυμητή η περιστροφή του άξονα προς την αντίθετη κατεύθυνση, θα πρέπει τα πηνία να ενεργοποιηθούν από το όγδοο βήμα προς το πρώτο.

Πίνακας 2.5: Η σειρά ενεργοποίησης των πηνίων για την επίτευξη της κίνησης μισού βήματος [21].

Βήμα	Πηνίο Νº 1	Πηνίο Νº 2	Πηνίο Νº 3	Πηνίο Νº 4
1	**High**	**Low**	**Low**	**Low**
2	**High**	**High**	**Low**	**Low**
3	**Low**	**High**	**Low**	**Low**
4	**Low**	**High**	**High**	**Low**
5	**Low**	**Low**	**High**	**Low**
6	**Low**	**Low**	**High**	**High**
7	**Low**	**Low**	**Low**	**High**
8	**High**	**Low**	**Low**	**High**

Εν συνεχεία, για να μπορέσει να περιστραφεί ο άξονας του βηματικού κινητήρα, θα πρέπει η πλακέτα οδήγησης του βηματικού κινητήρα να δεχτεί τα κατάλληλα σήματα από τις ψηφιακές εξόδους του μικροελεγκτή. Πιο συγκεκριμένα, ο μικροελεγκτής μέσω του κώδικα, καθορίζει τόσο την φορά όσο και την διάρκεια περιστροφής του άξονα του βηματικού κινητήρα. Ωστόσο, ο έλεγχος του βηματικού κινητήρα μπορεί να γίνει με δύο τρόπους, είτε χειροκίνητα είτε με την χρήση της βιβλιοθήκης για βηματικό κινητήρα που παρέχει το Arduino IDE. Όμως, η βιβλιοθήκη για τον έλεγχο του βηματικού κινητήρα, χρησιμοποιεί την κίνηση κύματος ή αλλιώς ολόκληρου βήματος (πίνακας 2.6), η οποία παρέχει μια μέτρια ροπή και μέτρια κατανάλωση [133] [134].

Πίνακας 2.6: Ο κώδικας προγραμματισμού για την περιστροφή του άξονα του βηματικού κινητήρα με την χρήση βιβλιοθήκης [22].

Κώδικας προγραμματισμού	Ανάλυση κώδικα
#include <Stepper.h> Stepper stepper(2048, 8, 9, 10, 11);	Εισαγωγή της βιβλιοθήκης Stepper.h Δημιουργία αντικειμένου τύπου Stepper με όνομα stepper και καθορισμός βημάτων πλήρους περιστροφής και τον ακροδεκτών

	ψηφιακών εξόδων για τον έλεγχο του βηματικού κινητήρα.
void setup() {	Η συνάρτηση setup(), η οποία εκτελείται μόνο μία φορά κατά την ενεργοποίηση του Arduino. Στην συνάρτηση αυτή αρχικοποιείται η ταχύτητα του βηματικού κινητήρα.
stepper.setSpeed(5);	
}	
void loop() {	Η συνάρτηση loop(), η οποία επαναλαμβάνεται κάθε φορά μετά την ολοκλήρωσή της.
stepper.step(2048);	Εντολή για την περιστροφή του άξονα του βηματικού κινητήρα 2048 βήματα ωρολογιακά.
delay(1000);	Εντολή καθυστέρησης 1000msec (1sec).
stepper.step(-2048);	Εντολή για μία πλήρης περιστροφή του άξονα του βηματικού κινητήρα 2048 βήματα αντι-ωρολογιακά.
delay(1000);	Εντολή καθυστέρησης 1000msec (1sec).
}	

Επειδή στην συγκεκριμένη εργασία, ο βηματικός κινητήρας πρέπει να παρέχει την μέγιστη ροπή του για να περιστρέψει το κλειδί, επιλέχθηκε ο χειροκίνητος έλεγχος του βηματικού κινητήρα. Συνεπώς για την επίτευξη της μέγιστης ροπής από τον βηματικό κινητήρα, επιλέχθηκε η κανονική κίνηση για την συγκεκριμένη εργασία (πίνακας 2.7). Ωστόσο, για λόγους πλήρωσης της ενότητας και μόνο εμφανίζονται οι κώδικες για τον χειροκίνητο έλεγχο των άλλων δύο περιπτώσεων περιστροφής του βηματικού κινητήρα (πίνακας 2.8 & πίνακας 2.9) [135].

Πίνακας 2.7: Ο κώδικας προγραμματισμού για την επίτευξη της κανονικής κίνησης του άξονα του βηματικού κινητήρα αριστερόστροφα και δεξιόστροφα [23].

Κώδικας προγραμματισμού	**Ανάλυση κώδικα**
#define Stepper_Pin1 42 #define Stepper_Pin2 43 #define Stepper_Pin3 44 #define Stepper_Pin4 45	Δήλωση και αρχικοποίηση των ψηφιακών ακροδεκτών για των έλεγχο του βηματικού κινητήρα.
void setup() {	Η συνάρτηση setup() η οποία εκτελείται μία φορά κατά την ενεργοποίηση του Arduino.

pinMode(Stepper_Pin1, OUTPUT);
pinMode(Stepper_Pin2, OUTPUT);
pinMode(Stepper_Pin3, OUTPUT);
pinMode(Stepper_Pin4, OUTPUT);
 	
digitalWrite(Stepper_Pin1, LOW);
digitalWrite(Stepper_Pin2, LOW);
digitalWrite(Stepper_Pin3, LOW);
digitalWrite(Stepper_Pin4, LOW);
}
	Καθορισμός των ψηφιακών ακροδεκτών ελέγχου της περιστροφής του βηματικού κινητήρα ως έξοδο.
	
	Ορισμός των ψηφιακών εξόδου 42, 43, 44, 45 σε χαμηλό δυναμικό δηλαδή 0Volt. Με αυτόν τον τρόπο μέσο της πλακέτας οδήγησης του βηματικού δεν διαρρέεται σε κανένα από τα τέσσερα πηνία ρεύμα.

void loop() {	Η συνάρτηση loop() η οποία επαναλαμβάνεται κάθε φορά μετά την ολοκλήρωσή της.
for(int i=0; i<=512; i++) {	Δομή επανάληψη for η οποία έχει έναν μετρητή i με τιμή 0 και κάθε φορά που ολοκληρώνεται η εκτέλεση του μπλοκ εντολών αυξάνεται κατά 1. Η αύξηση συνεχίζει έως ότου η συνθήκη γίνει ψευδής, δηλαδή μέχρι ο μετρητής i να φτάσει την τιμή 513. Η τιμή 512 καθορίζει το σύνολο των βημάτων. Στην συγκεκριμένη περίπτωση ο άξονα του βηματικού κινητήρα θα κάνει 2048 βήματα ωρολογιακά.

digitalWrite(Stepper_Pin1, HIGH);
digitalWrite(Stepper_Pin2, HIGH);
digitalWrite(Stepper_Pin3, LOW);
digitalWrite(Stepper_Pin4, LOW);
 	
delay(2);
	Ορισμός των ψηφιακών εξόδων 44 & 45 σε χαμηλό δυναμικό δηλαδή 0Volt και των ψηφιακών εξόδων 42 & 43 σε υψηλό δυναμικό 5Volt. Έτσι στο πρώτο βήμα ενεργοποιούνται δύο από τα τέσσερα πηνία και ο άξονα κάνει ένα βήμα ωρολογιακά.
	Καθυστέρηση 2msec.

digitalWrite(Stepper_Pin1, LOW);
digitalWrite(Stepper_Pin2, HIGH);
digitalWrite(Stepper_Pin3, HIGH);
digitalWrite(Stepper_Pin4, LOW);
 	
delay(2);
	Ορισμός των ψηφιακών εξόδων 42 & 45 σε χαμηλό δυναμικό δηλαδή 0Volt και των ψηφιακών εξόδων 43 & 44 σε υψηλό δυναμικό 5Volt. Έτσι στο δεύτερο βήμα ενεργοποιούνται δύο από τα τέσσερα πηνία και ο άξονα κάνει ένα βήμα ωρολογιακά.
	Καθυστέρηση 2msec.

digitalWrite(Stepper_Pin1, LOW);
digitalWrite(Stepper_Pin2, LOW);
digitalWrite(Stepper_Pin3, HIGH);
	Ορισμός των ψηφιακών εξόδων 42 & 43 σε χαμηλό δυναμικό δηλαδή 0Volt και των ψηφιακών εξόδων 43 & 44 σε υψηλό

digitalWrite(Stepper_Pin4, HIGH);	δυναμικό 5Volt. Έτσι στο τρίτο βήμα ενεργοποιούνται δύο από τα τέσσερα πηνία και ο άξονα κάνει ένα βήμα ωρολογιακά.
delay(2);	Καθυστέρηση 2msec.

digitalWrite(Stepper_Pin1, HIGH);
digitalWrite(Stepper_Pin2, LOW);
digitalWrite(Stepper_Pin3, LOW);
digitalWrite(Stepper_Pin4, HIGH);
	Ορισμός των ψηφιακών εξόδων 43 & 44 σε χαμηλό δυναμικό δηλαδή 0Volt και των ψηφιακών εξόδων 42 & 45 σε υψηλό δυναμικό 5Volt. Έτσι στο τέταρτο βήμα ενεργοποιούνται δύο από τα τέσσερα πηνία και ο άξονα κάνει ένα βήμα ωρολογιακά.

delay(2); }	Καθυστέρηση 2msec.
for(int i=0; i<=512; i++) {	Δομή επανάληψη for η οποία έχει έναν μετρητή i με τιμή 0 και κάθε φορά που ολοκληρώνεται η εκτέλεση του μπλοκ εντολών αυξάνεται κατά 1. Η αύξηση συνεχίζει έως ότου η συνθήκη γίνει ψευδής, δηλαδή μέχρι ο μετρητής i να φτάσει την τιμή 513. Η τιμή 512 καθορίζει το σύνολο των βημάτων. Στην συγκεκριμένη περίπτωση ο άξονα του βηματικού κινητήρα θα κάνει 2048 βήματα αντι-ωρολογιακά.

digitalWrite(Stepper_Pin1, LOW);
digitalWrite(Stepper_Pin2, LOW);
digitalWrite(Stepper_Pin3, HIGH);
digitalWrite(Stepper_Pin4, HIGH);
	Ορισμός των ψηφιακών εξόδων 42 & 43 σε χαμηλό δυναμικό δηλαδή 0Volt και των ψηφιακών εξόδων 44 & 45 σε υψηλό δυναμικό 5Volt. Έτσι στο πρώτο βήμα ενεργοποιούνται δύο από τα τέσσερα πηνία και ο άξονα κάνει ένα βήμα αντι-ωρολογιακά.

delay(2);	Καθυστέρηση 2msec.

digitalWrite(Stepper_Pin1, LOW);
digitalWrite(Stepper_Pin2, HIGH);
digitalWrite(Stepper_Pin3, HIGH);
digitalWrite(Stepper_Pin4, LOW);
	Ορισμός των ψηφιακών εξόδων 42 & 45 σε χαμηλό δυναμικό δηλαδή 0Volt και των ψηφιακών εξόδων 43 & 44 σε υψηλό δυναμικό 5Volt. Έτσι στο δεύτερο βήμα ενεργοποιούνται δύο από τα τέσσερα πηνία και ο άξονα κάνει ένα βήμα αντι-ωρολογιακά.

delay(2);	Καθυστέρηση 2msec.

digitalWrite(Stepper_Pin1, HIGH);
digitalWrite(Stepper_Pin2, HIGH);
digitalWrite(Stepper_Pin3, LOW);
	Ορισμός των ψηφιακών εξόδων 44 & 45 σε χαμηλό δυναμικό δηλαδή 0Volt και των ψηφιακών εξόδων 42 & 43 σε υψηλό

digitalWrite(Stepper_Pin4, LOW);	δυναμικό 5Volt. Έτσι στο τρίτο βήμα ενεργοποιούνται δύο από τα τέσσερα πηνία και ο άξονα κάνει ένα βήμα αντι-ωρολογιακά.
delay(2);	Καθυστέρηση 2msec.

digitalWrite(Stepper_Pin1, HIGH);
digitalWrite(Stepper_Pin2, LOW);
digitalWrite(Stepper_Pin3, LOW);
digitalWrite(Stepper_Pin4, HIGH);
	Ορισμός των ψηφιακών εξόδων 43 & 44 σε χαμηλό δυναμικό δηλαδή 0Volt και των ψηφιακών εξόδων 42 & 45 σε υψηλό δυναμικό 5Volt. Έτσι στο τέταρτο βήμα ενεργοποιούνται δύο από τα τέσσερα πηνία και ο άξονα κάνει ένα βήμα αντι-ωρολογιακά.

delay(2); }	Καθυστέρηση 2msec.

Πίνακας 2.8: Ο κώδικας προγραμματισμού για την επίτευξη της κίνησης κύματος ή αλλιώς του ολόκληρου βήματος του άξονα του βηματικού κινητήρα αριστερόστροφα και δεξιόστροφα [24].

Κώδικας προγραμματισμού	**Ανάλυση κώδικα**
#define Stepper_Pin1 42 #define Stepper_Pin2 43 #define Stepper_Pin3 44 #define Stepper_Pin4 45	Δήλωση και αρχικοποίηση των ψηφιακών ακροδεκτών για των έλεγχο του βηματικού κινητήρα.
void setup() {	Η συνάρτηση setup() η οποία εκτελείται μία φορά κατά την ενεργοποίηση του Arduino.

pinMode(Stepper_Pin1, OUTPUT);
pinMode(Stepper_Pin2, OUTPUT);
pinMode(Stepper_Pin3, OUTPUT);
pinMode(Stepper_Pin4, OUTPUT);
	Καθορισμός των ψηφιακών ακροδεκτών ελέγχου της περιστροφής του βηματικού κινητήρα ως έξοδο.

digitalWrite(Stepper_Pin1, LOW);
digitalWrite(Stepper_Pin2, LOW);
digitalWrite(Stepper_Pin3, LOW);
digitalWrite(Stepper_Pin4, LOW);
}
	Ορισμός των ψηφιακών εξόδου 42, 43, 44, 45 σε χαμηλό δυναμικό δηλαδή 0Volt. Με αυτόν τον τρόπο μέσο της πλακέτας οδήγησης του βηματικού δεν διαρρέεται σε κανένα από τα τέσσερα πηνία ρεύμα.

void loop() {	Η συνάρτηση loop() η οποία επαναλαμβάνεται κάθε φορά μετά την ολοκλήρωσή της.
	Δομή επανάληψη for η οποία έχει έναν μετρητή i με τιμή 0 και κάθε φορά που

	ολοκληρώνεται η εκτέλεση του μπλοκ εντολών αυξάνεται κατά 1. Η αύξηση συνεχίζει έως ότου η συνθήκη γίνει ψευδής,
for(int i=0; i<=512; i++) {	δηλαδή μέχρι ο μετρητής i να φτάσει την τιμή 513. Η τιμή 512 καθορίζει το σύνολο των βημάτων. Στην συγκεκριμένη περίπτωση ο άξονα του βηματικού κινητήρα θα κάνει 4096 βήματα ωρολογιακά.

digitalWrite(Stepper_Pin1, HIGH);
digitalWrite(Stepper_Pin2, LOW);
digitalWrite(Stepper_Pin3, LOW);
digitalWrite(Stepper_Pin4, LOW);
	Ορισμός των ψηφιακών εξόδων 43, 44 & 45 σε χαμηλό δυναμικό δηλαδή 0Volt και της ψηφιακής εξόδου 42 σε υψηλό δυναμικό 5Volt. Έτσι στο πρώτο βήμα ενεργοποιείται ένα από τα τέσσερα πηνία και ο άξονα κάνει ένα βήμα ωρολογιακά.

delay(2);	Καθυστέρηση 2msec.

digitalWrite(Stepper_Pin1, LOW);
digitalWrite(Stepper_Pin2, HIGH);
digitalWrite(Stepper_Pin3, LOW);
digitalWrite(Stepper_Pin4, LOW);
	Ορισμός των ψηφιακών εξόδων 42, 44 & 45 σε χαμηλό δυναμικό δηλαδή 0Volt και της ψηφιακής εξόδου 43 σε υψηλό δυναμικό 5Volt. Έτσι στο δεύτερο βήμα ενεργοποιείται ένα από τα τέσσερα πηνία και ο άξονα κάνει ένα βήμα ωρολογιακά.

delay(2);	Καθυστέρηση 2msec.

digitalWrite(Stepper_Pin1, LOW);
digitalWrite(Stepper_Pin2, LOW);
digitalWrite(Stepper_Pin3, HIGH);
digitalWrite(Stepper_Pin4, LOW);
	Ορισμός των ψηφιακών εξόδων 42, 43 & 45 σε χαμηλό δυναμικό δηλαδή 0Volt και της ψηφιακής εξόδου 44 σε υψηλό δυναμικό 5Volt. Έτσι στο τρίτο βήμα ενεργοποιείται ένα από τα τέσσερα πηνία και ο άξονα κάνει ένα βήμα ωρολογιακά.

delay(2);	Καθυστέρηση 2msec.

digitalWrite(Stepper_Pin1, LOW);
digitalWrite(Stepper_Pin2, LOW);
digitalWrite(Stepper_Pin3, LOW);
digitalWrite(Stepper_Pin4, HIGH);
	Ορισμός των ψηφιακών εξόδων 42,43 & 44 σε χαμηλό δυναμικό δηλαδή 0Volt και της ψηφιακής εξόδου 45 σε υψηλό δυναμικό 5Volt. Έτσι στο τέταρτο βήμα ενεργοποιείται ένα από τα τέσσερα πηνία και ο άξονα κάνει ένα βήμα ωρολογιακά.

delay(2); }	Καθυστέρηση 2msec.
	Δομή επανάληψη for η οποία έχει έναν μετρητή i με τιμή 0 και κάθε φορά που ολοκληρώνεται η εκτέλεση του μπλοκ εντολών αυξάνεται κατά 1. Η αύξηση

for(int i=0; i<=512; i++) {	συνεχίζει έως ότου η συνθήκη γίνει ψευδής, δηλαδή μέχρι ο μετρητής i να φτάσει την τιμή 513. Η τιμή 512 καθορίζει το σύνολο των βημάτων. Στην συγκεκριμένη περίπτωση ο άξονα του βηματικού κινητήρα θα κάνει 2048 βήματα αντι-ωρολογιακά.

digitalWrite(Stepper_Pin1, LOW);
digitalWrite(Stepper_Pin2, LOW);
digitalWrite(Stepper_Pin3, HIGH);
digitalWrite(Stepper_Pin4, LOW);
	Ορισμός των ψηφιακών εξόδων 42, 43 & 45 σε χαμηλό δυναμικό δηλαδή 0Volt και της ψηφιακής εξόδου 44 σε υψηλό δυναμικό 5Volt. Έτσι στο πρώτο βήμα ενεργοποιείται ένα από τα τέσσερα πηνία και ο άξονα κάνει ένα βήμα αντι-ωρολογιακά.

delay(2);	Καθυστέρηση 2msec.

digitalWrite(Stepper_Pin1, LOW);
digitalWrite(Stepper_Pin2, HIGH);
digitalWrite(Stepper_Pin3, LOW);
digitalWrite(Stepper_Pin4, LOW);
	Ορισμός των ψηφιακών εξόδων 42, 44 & 45 σε χαμηλό δυναμικό δηλαδή 0Volt και της ψηφιακής εξόδου 43 σε υψηλό δυναμικό 5Volt. Έτσι στο δεύτερο βήμα ενεργοποιείται ένα από τα τέσσερα πηνία και ο άξονα κάνει ένα βήμα αντι-ωρολογιακά.

delay(2);	Καθυστέρηση 2msec.

digitalWrite(Stepper_Pin1, HIGH);
digitalWrite(Stepper_Pin2, LOW);
digitalWrite(Stepper_Pin3, LOW);
digitalWrite(Stepper_Pin4, LOW);
	Ορισμός των ψηφιακών εξόδων 43, 44 & 45 σε χαμηλό δυναμικό δηλαδή 0Volt και της ψηφιακής εξόδου 42 σε υψηλό δυναμικό 5Volt. Έτσι στο τρίτο βήμα ενεργοποιείται ένα από τα τέσσερα πηνία και ο άξονα κάνει ένα βήμα αντι-ωρολογιακά.

delay(2);	Καθυστέρηση 2msec.

digitalWrite(Stepper_Pin1, LOW);
digitalWrite(Stepper_Pin2, LOW);
digitalWrite(Stepper_Pin3, LOW);
digitalWrite(Stepper_Pin4, HIGH);
	Ορισμός των ψηφιακών εξόδων 42, 43 & 44 σε χαμηλό δυναμικό δηλαδή 0Volt και της ψηφιακής εξόδου 45 σε υψηλό δυναμικό 5Volt. Έτσι στο τέταρτο βήμα ενεργοποιείται ένα από τα τέσσερα πηνία και ο άξονα κάνει ένα βήμα αντι-ωρολογιακά.

delay(2); }	Καθυστέρηση 2msec.

Πίνακας 2.9: Ο κώδικας προγραμματισμού για την επίτευξη της κίνησης μισού βήματος του άξονα του βηματικού κινητήρα αριστερόστροφα και δεξιόστροφα [25].

Κώδικας προγραμματισμού	Ανάλυση κώδικα
#define Stepper_Pin1 42 #define Stepper_Pin2 43 #define Stepper_Pin3 44 #define Stepper_Pin4 45	Δήλωση και αρχικοποίηση των ψηφιακών ακροδεκτών για των έλεγχο του βηματικού κινητήρα.
void setup() {	Η συνάρτηση setup() η οποία εκτελείται μία φορά κατά την ενεργοποίηση του Arduino.

pinMode(Stepper_Pin1, OUTPUT);
pinMode(Stepper_Pin2, OUTPUT);
pinMode(Stepper_Pin3, OUTPUT);
pinMode(Stepper_Pin4, OUTPUT);
	Καθορισμός των ψηφιακών ακροδεκτών ελέγχου της περιστροφής του βηματικού κινητήρα ως έξοδο.

digitalWrite(Stepper_Pin1, LOW);
digitalWrite(Stepper_Pin2, LOW);
digitalWrite(Stepper_Pin3, LOW);
digitalWrite(Stepper_Pin4, LOW);
}
	Ορισμός των ψηφιακών εξόδου 42, 43, 44, 45 σε χαμηλό δυναμικό δηλαδή 0Volt. Με αυτόν τον τρόπο μέσο της πλακέτας οδήγησης του βηματικού δεν διαρρέεται σε κανένα από τα τέσσερα πηνία ρεύμα.

void loop() {	Η συνάρτηση loop() η οποία επαναλαμβάνεται κάθε φορά μετά την ολοκλήρωσή της.
for(int i=0; i<=512; i++) {	Δομή επανάληψη for η οποία έχει έναν μετρητή i με τιμή 0 και κάθε φορά που ολοκληρώνεται η εκτέλεση του μπλοκ εντολών αυξάνεται κατά 1. Η αύξηση συνεχίζει έως ότου η συνθήκη γίνει ψευδής, δηλαδή μέχρι ο μετρητής i να φτάσει την τιμή 513. Η τιμή 512 καθορίζει το σύνολο των βημάτων. Στην συγκεκριμένη περίπτωση ο άξονα του βηματικού κινητήρα θα κάνει 4096 βήματα ωρολογιακά.

digitalWrite(Stepper_Pin1, HIGH);
digitalWrite(Stepper_Pin2, LOW);
digitalWrite(Stepper_Pin3, LOW);
digitalWrite(Stepper_Pin4, LOW);
	Ορισμός των ψηφιακών εξόδων 43, 44 & 45 σε χαμηλό δυναμικό δηλαδή 0Volt και της ψηφιακής εξόδου 42 σε υψηλό δυναμικό 5Volt. Έτσι στο πρώτο βήμα ενεργοποιείται ένα από τα τέσσερα πηνία και ο άξονα κάνει ένα βήμα ωρολογιακά.

delay(2);	Καθυστέρηση 2msec.

digitalWrite(Stepper_Pin1, HIGH);
digitalWrite(Stepper_Pin2, HIGH);
digitalWrite(Stepper_Pin3, LOW);
digitalWrite(Stepper_Pin4, LOW);
	Ορισμός των ψηφιακών εξόδων 44 & 45 σε χαμηλό δυναμικό δηλαδή 0Volt και των ψηφιακών εξόδων 42 & 43 σε υψηλό δυναμικό 5Volt. Έτσι στο δεύτερο βήμα ενεργοποιούνται δύο από τα τέσσερα πηνία και ο άξονα κάνει ένα βήμα ωρολογιακά.

delay(2);	Καθυστέρηση 2msec.

digitalWrite(Stepper_Pin1, LOW);
digitalWrite(Stepper_Pin2, HIGH);
digitalWrite(Stepper_Pin3, LOW);
digitalWrite(Stepper_Pin4, LOW);
	Ορισμός των ψηφιακών εξόδων 42, 44 & 45 σε χαμηλό δυναμικό δηλαδή 0Volt και της ψηφιακής εξόδου 43 σε υψηλό δυναμικό 5Volt. Έτσι στο τρίτο βήμα ενεργοποιείται ένα από τα τέσσερα πηνία και ο άξονα κάνει ένα βήμα ωρολογιακά.

delay(2);	Καθυστέρηση 2msec.

digitalWrite(Stepper_Pin1, LOW);
digitalWrite(Stepper_Pin2, HIGH);
digitalWrite(Stepper_Pin3, HIGH);
digitalWrite(Stepper_Pin4, LOW);
	Ορισμός των ψηφιακών εξόδων 42 & 45 σε χαμηλό δυναμικό δηλαδή 0Volt και των ψηφιακών εξόδων 43 & 44 σε υψηλό δυναμικό 5Volt. Έτσι στο τέταρτο βήμα ενεργοποιούνται δύο από τα τέσσερα πηνία και ο άξονα κάνει ένα βήμα ωρολογιακά.

delay(2);	Καθυστέρηση 2msec.

digitalWrite(Stepper_Pin1, LOW);
digitalWrite(Stepper_Pin2, LOW);
digitalWrite(Stepper_Pin3, HIGH);
digitalWrite(Stepper_Pin4, LOW);
	Ορισμός των ψηφιακών εξόδων 42, 43 & 45 σε χαμηλό δυναμικό δηλαδή 0Volt και της ψηφιακής εξόδου 44 σε υψηλό δυναμικό 5Volt. Έτσι στο πέμπτο βήμα ενεργοποιείται ένα από τα τέσσερα πηνία και ο άξονα κάνει ένα βήμα ωρολογιακά.

delay(2);	Καθυστέρηση 2msec.

digitalWrite(Stepper_Pin1, LOW);
digitalWrite(Stepper_Pin2, LOW);
digitalWrite(Stepper_Pin3, HIGH);
digitalWrite(Stepper_Pin4, HIGH);
	Ορισμός των ψηφιακών εξόδων 42 & 43 σε χαμηλό δυναμικό δηλαδή 0Volt και των ψηφιακών εξόδων 44 & 45 σε υψηλό δυναμικό 5Volt. Έτσι στο έκτο βήμα ενεργοποιούνται δύο από τα τέσσερα πηνία και ο άξονα κάνει ένα βήμα ωρολογιακά.

delay(2);	Καθυστέρηση 2msec

digitalWrite(Stepper_Pin1, LOW);
digitalWrite(Stepper_Pin2, LOW);
digitalWrite(Stepper_Pin3, LOW);
digitalWrite(Stepper_Pin4, HIGH);
	Ορισμός των ψηφιακών εξόδων 42, 43 & 44 σε χαμηλό δυναμικό δηλαδή 0Volt και της ψηφιακής εξόδου 45 σε υψηλό δυναμικό 5Volt. Έτσι στο έβδομο βήμα ενεργοποιείται ένα από τα τέσσερα πηνία

	και ο άξονα κάνει ένα βήμα ωρολογιακά.
delay(2);	Καθυστέρηση 2msec.

digitalWrite(Stepper_Pin1, HIGH);
digitalWrite(Stepper_Pin2, LOW);
digitalWrite(Stepper_Pin3, LOW);
digitalWrite(Stepper_Pin4, HIGH);
	Ορισμός των ψηφιακών εξόδων 43 & 44 σε χαμηλό δυναμικό δηλαδή 0Volt και των ψηφιακών εξόδων 42 & 45 σε υψηλό δυναμικό 5Volt. Έτσι στο όγδοο βήμα ενεργοποιούνται δύο από τα τέσσερα πηνία και ο άξονα κάνει ένα βήμα ωρολογιακά.

delay(2);	Καθυστέρηση 2msec
}	
for(int i=0; i<=512; i++) {	Δομή επανάληψη for η οποία έχει έναν μετρητή i με τιμή 0 και κάθε φορά που ολοκληρώνεται η εκτέλεση του μπλοκ εντολών αυξάνεται κατά 1. Η αύξηση συνεχίζει έως ότου η συνθήκη γίνει ψευδής, δηλαδή μέχρι ο μετρητής i να φτάσει την τιμή 513. Η τιμή 512 καθορίζει το σύνολο των βημάτων. Στην συγκεκριμένη περίπτωση ο άξονα του βηματικού κινητήρα θα κάνει 2048 βήματα αντι-ωρολογιακά.

digitalWrite(Stepper_Pin1, LOW);
digitalWrite(Stepper_Pin2, LOW);
digitalWrite(Stepper_Pin3, LOW);
digitalWrite(Stepper_Pin4, HIGH);
	Ορισμός των ψηφιακών εξόδων 42, 43 & 44 σε χαμηλό δυναμικό δηλαδή 0Volt και της ψηφιακής εξόδου 45 σε υψηλό δυναμικό 5Volt. Έτσι στο πρώτο βήμα ενεργοποιείται ένα από τα τέσσερα πηνία και ο άξονα κάνει ένα βήμα αντι-ωρολογιακά.

delay(2);	Καθυστέρηση 2msec.

digitalWrite(Stepper_Pin1, LOW);
digitalWrite(Stepper_Pin2, LOW);
digitalWrite(Stepper_Pin3, HIGH);
digitalWrite(Stepper_Pin4, HIGH);
	Ορισμός των ψηφιακών εξόδων 42 & 43 σε χαμηλό δυναμικό δηλαδή 0Volt και της ψηφιακής εξόδου 44 & 45 σε υψηλό δυναμικό 5Volt. Έτσι στο δεύτερο βήμα ενεργοποιούνται δύο από τα τέσσερα πηνία και ο άξονα κάνει ένα βήμα αντι-ωρολογιακά.

delay(2);	Καθυστέρηση 2msec.

digitalWrite(Stepper_Pin1, LOW);
digitalWrite(Stepper_Pin2, LOW);
digitalWrite(Stepper_Pin3, HIGH);
digitalWrite(Stepper_Pin4, LOW);
	Ορισμός των ψηφιακών εξόδων 42, 43 & 45 σε χαμηλό δυναμικό δηλαδή 0Volt και της ψηφιακής εξόδου 44 σε υψηλό δυναμικό 5Volt. Έτσι στο τρίτο βήμα ενεργοποιείται ένα από τα τέσσερα πηνία και ο άξονα κάνει ένα βήμα αντι-

	ωρολογιακά.
delay(2);	Καθυστέρηση 2msec.

digitalWrite(Stepper_Pin1, LOW);
digitalWrite(Stepper_Pin2, HIGH);
digitalWrite(Stepper_Pin3, HIGH);
digitalWrite(Stepper_Pin4, LOW);
	Ορισμός των ψηφιακών εξόδων 42 & 45 σε χαμηλό δυναμικό δηλαδή 0Volt και των ψηφιακών εξόδων 43 & 44 σε υψηλό δυναμικό 5Volt. Έτσι στο τέταρτο βήμα ενεργοποιούνται δύο από τα τέσσερα πηνία και ο άξονα κάνει ένα βήμα αντι-ωρολογιακά.

delay(2);	Καθυστέρηση 2msec.

digitalWrite(Stepper_Pin1, LOW);
digitalWrite(Stepper_Pin2, HIGH);
digitalWrite(Stepper_Pin3, LOW);
digitalWrite(Stepper_Pin4, LOW);
	Ορισμός των ψηφιακών εξόδων 42, 44 & 45 σε χαμηλό δυναμικό δηλαδή 0Volt και της ψηφιακής εξόδου 43 σε υψηλό δυναμικό 5Volt. Έτσι στο πέμπτο βήμα ενεργοποιείται ένα από τα τέσσερα πηνία και ο άξονα κάνει ένα βήμα αντι-ωρολογιακά.

delay(2);	Καθυστέρηση 2msec.

digitalWrite(Stepper_Pin1, HIGH);
digitalWrite(Stepper_Pin2, HIGH);
digitalWrite(Stepper_Pin3, LOW);
digitalWrite(Stepper_Pin4, LOW);
	Ορισμός των ψηφιακών εξόδων 44 & 45 σε χαμηλό δυναμικό δηλαδή 0Volt και των ψηφιακών εξόδων 42 & 43 σε υψηλό δυναμικό 5Volt. Έτσι στο έκτο βήμα ενεργοποιούνται δύο από τα τέσσερα πηνία και ο άξονα κάνει ένα βήμα αντι-ωρολογιακά.

delay(2);	Καθυστέρηση 2msec.

digitalWrite(Stepper_Pin1, HIGH);
digitalWrite(Stepper_Pin2, LOW);
digitalWrite(Stepper_Pin3, LOW);
digitalWrite(Stepper_Pin4, LOW);
	Ορισμός των ψηφιακών εξόδων 43, 44 & 45 σε χαμηλό δυναμικό δηλαδή 0Volt και της ψηφιακής εξόδου 42 σε υψηλό δυναμικό 5Volt. Έτσι στο έβδομο βήμα ενεργοποιείται ένα από τα τέσσερα πηνία και ο άξονα κάνει ένα βήμα αντι-ωρολογιακά.

delay(2);	Καθυστέρηση 2msec.

digitalWrite(Stepper_Pin1, HIGH);
digitalWrite(Stepper_Pin2, LOW);
digitalWrite(Stepper_Pin3, LOW);
digitalWrite(Stepper_Pin4, HIGH);
	Ορισμός των ψηφιακών εξόδων 43 & 44 σε χαμηλό δυναμικό δηλαδή 0Volt και των ψηφιακών εξόδων 42 & 45 σε υψηλό δυναμικό 5Volt. Έτσι στο όγδοο βήμα ενεργοποιούνται δύο από τα τέσσερα πηνία και ο άξονα κάνει ένα βήμα αντι-ωρολογιακά.

delay(2);	Καθυστέρηση 2msec.

}	

2.2.5 LCD 16 χαρακτήρων και 2 σειρών με ενσωματωμένο πληκτρολόγιο

Στην συγκεκριμένη εργασία, ένα από τα βασικότερα κομμάτια είναι η διεπαφή μεταξύ χρήστη και συστήματος. Με αυτόν τον τρόπο ο χρήστης μπορεί να επιλέξει τον τύπο κουρδίσματος και να εκκινήσει την διαδικασία κουρδίσματος, ενώ το σύστημα με την σειρά του καθοδηγεί τον χρήστη για το ποια χορδή πρέπει να διεγείρει καθώς και τον ειδοποιεί για την ολοκλήρωση του κουρδίσματος. Όπως ειπώθηκε προηγουμένως, για το σύστημα αυτομάτου κουρδίσματος των χορδών χρησιμοποιήθηκε μία LCD οθόνη 16x2 με ενσωματωμένο πληκτρολόγιο (Εικόνα Εικόνα 2.27) [124].

Εικόνα 2.27: Η LCD οθόνη 16x02 με ενσωματωμένο πληκτρολόγιο [180].

Αρχικά, ο συγκεκριμένος τύπος οθόνης χρησιμοποιεί την τεχνολογία υγρών κρυστάλλων (Liquid Crystal Display) και υποστηρίζει την εκτύπωση 16 χαρακτήρων σε κάθε μία από τις δύο γραμμές. Επιπλέον, περιλαμβάνει μπλε οπίσθιο φωτισμό, υψηλή αντίθεση, η οποία ρυθμίζεται μέσω ενός ποτενσιόμετρου και μεγάλη γωνία προβολής ώστε οι χαρακτήρες να είναι ευδιάκριτοι. Η τροφοδοσία της συγκεκριμένης οθόνης και του πληκτρολογίου είναι 5Volt και παρέχεται από το ίδιο το Arduino. Επίσης, η πλακέτα αυτή στο πίσω μέρος της περιλαμβάνει ακροδέκτες οι οποίοι χρησιμοποιούνται για να εφαρμοστεί επάνω σε πλακέτες Arduino UNO και Arduino Mega 2560. Ωστόσο, η συγκεκριμένη πλακέτα κατοχυρώνει της ψηφιακές θύρες (I/O) 5, 6, 7, 8 & 9 οι οποίες χρησιμοποιούνται για την εκτύπωση των χαρακτήρων στην 16x02 LCD οθόνη. Επίσης, η πλακέτα κατοχυρώνει την αναλογική θύρα εισόδου A0 η οποία χρησιμοποιείται για την αναγνώριση του πληκτρολογίου που περιλαμβάνει η πλακέτα. Πιο συγκεκριμένα, σύμφωνα με το ηλεκτρονικό σχέδιο της πλακέτας (Εικόνα Εικόνα 2.28) [140], τα πλήκτρα είναι συνδεδεμένα παράλληλα και ανάλογα με το ποιο πλήκτρο πιέζεται δημιουργείται διαφορετική πτώση τάσης εξαιτίας των αντιστάσεων που είναι συνδεδεμένες σε σειρά. Με αυτόν τον τρόπο μέσω της αναλογική θύρα του Arduino (A0) μπορεί να εντοπιστεί ποιο από τα πλήκτρα έχει πατηθεί. Επιπροσθέτως, η πλακέτα με την LCD οθόνη περιλαμβάνει ένα ακόμα πλήκτρο το οποίο είναι συνδεδεμένο με τον ακροδέκτη reset του Arduino. Με

αυτόν τον τρόπο μπορεί ο χρήστης να κάνει επανεκκίνηση στον μικροελεγκτή αν το επιθυμεί. Τέλος, επειδή η πλακέτα οθόνης καλύπτει τους ακροδέκτες του Arduino, η ίδια παρέχει πρόσβαση σε αυτούς μέσω θέσεων συγκόλλησης κλεμοσειρών. Επομένως, τα καλώδια αντί να συνδέονται κατευθείαν στους ακροδέκτες του Arduino συνδέονται στις επιθυμητές κλεμοσειρές [82].

Εν συνεχεία, ας δούμε τους ακροδέκτες και τον τρόπο λειτουργίας της LCD οθόνης 16x02. Πιο συγκεκριμένα, η LCD οθόνη έχει 16 ακροδέκτες οι οποίοι είναι:

- **VSS:** Ο ακροδέκτης αυτός συνδέεται στον ακροδέκτη της γείωση (GND) του Arduino.

- **VDD:** Ο ακροδέκτης αυτός συνδέεται στον ακροδέκτη παροχής των 5Volt του Arduino.

- **V0:** Ο ακροδέκτης αυτός συνδέεται μέσω ενός ποτενσιόμετρου στην γείωση (GND) του Arduino. Περιστρέφοντας το ποτενσιόμετρο, αλλάζει η ωμική αντίσταση και έτσι ρυθμίζεται η αντίθεση της οθόνης.

- **RS:** Η LCD οθόνη περιλαμβάνει δύο καταχωρητές, έναν για την αποθήκευση εντολών και έναν για την αποθήκευση δεδομένων. Ο ακροδέκτης αυτός συνδέεται σε μία ψηφιακή θύρα του Arduino και μέσω αυτής γίνεται η επιλογή του καταχωρητή. Με αυτόν τον τρόπο ο μικροελεγκτής σηματοδοτεί το αν η πληροφορία που στέλνει είναι εντολή ή δεδομένο.

- **RW:** Ο ακροδέκτης αυτός καθορίζει τον τύπο της ενέργειας που θα γίνει στον καταχωρητή, δηλαδή ανάγνωση ή εγγραφή. Επειδή της περισσότερες φορές γίνεται εγγραφή στους καταχωρητές συνήθως ο ακροδέκτης αυτός συνδέεται με την γείωση (GND) του Arduino.

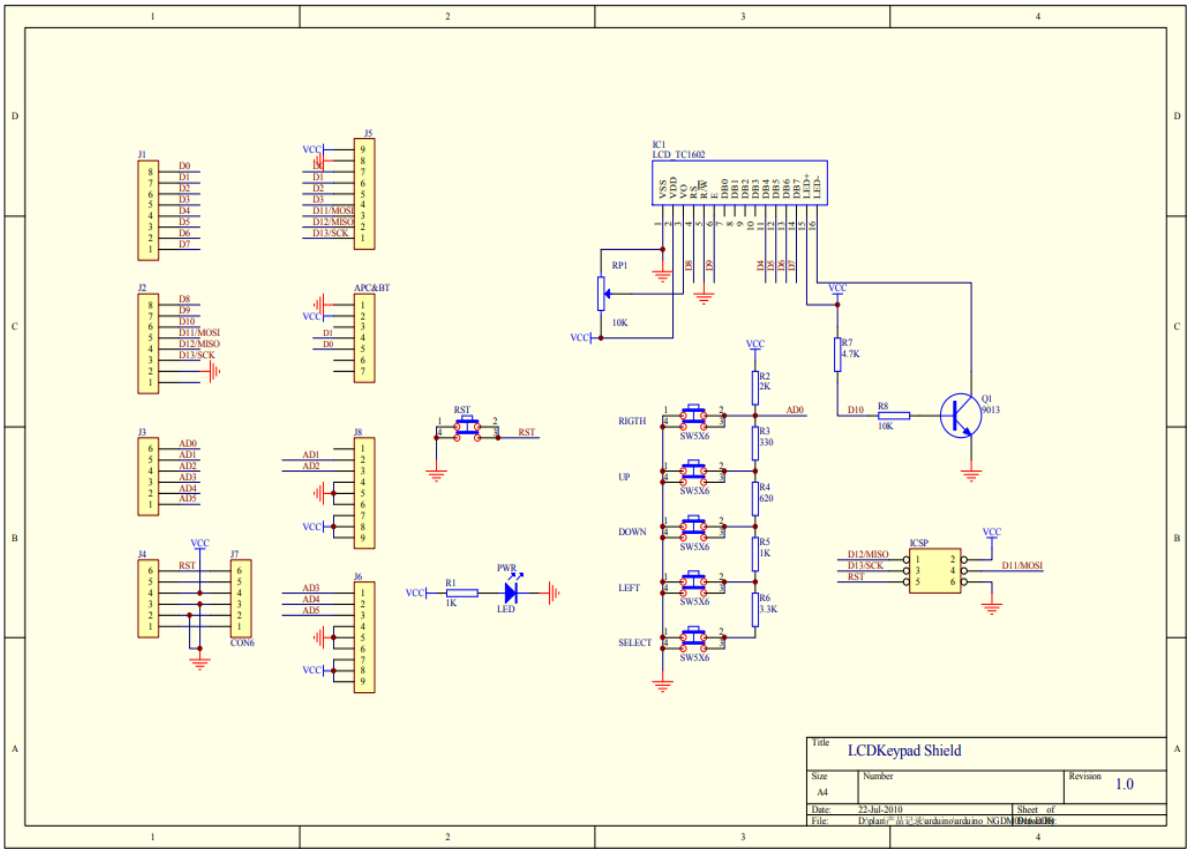
- **E:** Ο ακροδέκτης αυτός συνδέεται σε μία από τις ψηφιακές θύρες του Arduino. Σκοπός του είναι να μεταβάλλεται από την κατάσταση HIGH σε LOW κάθε φορά που υπάρχουν δεδομένα έτοιμα για εγγραφή στους καταχωρητές.

- **D0 - D7:** Μέσω αυτόν τον ακροδεκτών επιτυγχάνεται η παράλληλη μετάδοση δεδομένων 8bit. Πιο συγκεκριμένα, το Arduino υποστηρίζει δύο τρόπους αποστολής των δεδομένων στην LCD οθόνη. Ο ένας τρόπος χρησιμοποιεί και τους οκτώ ακροδέκτες της LCD οθόνης για την παράλληλη μετάδοση των οκτώ bit. Επομένως, με αυτόν τον τρόπο και οι οκτώ ακροδέκτες συνδέονται σε οκτώ ψηφιακές θύρες του Arduino. Ο δεύτερος τρόπος χρησιμοποιεί μόνο τους τέσσερις από τους οκτώ ακροδέκτες της LCD οθόνης και πιο συγκεκριμένα τους ακροδέκτες D4 – D7. Σε αυτή την περίπτωση μόνο οι τέσσερις ακροδέκτες συνδέονται σε τέσσερις ψηφιακές θύρες του Arduino και η αποστολή των οκτώ bit γίνεται σε δύο βήματα. Ο δεύτερος τρόπος είναι αυτός που χρησιμοποιείτε συχνότερα μιας και εξοικονομούνται ακροδέκτες στο Arduino.

- **A:** Ο ακροδέκτης αυτός έχει να κάνει με την άνοδο του LED στον οπίσθιο φωτισμό της LCD οθόνης. Ο ακροδέκτης αυτός μπορεί να συνδεθεί με τα 5Volt που παρέχει το Arduino για μόνιμη ενεργοποίηση του οπίσθιου φωτισμού της οθόνης. Από την άλλη, ο ακροδέκτης αυτός μπορεί να συνδεθεί με μία από τις ψηφιακές θύρες του Arduino. Με αυτό τον τρόπο ο οπίσθιος φωτισμός μπορεί να ενεργοποιείται όταν η ψηφιακή

θύρα γίνεται HIGH και να απενεργοποιείται όταν η ψηφιακή θύρα γίνεται LOW. Ωστόσο, αν είναι επιθυμητή η αυξομείωση του οπίσθιου φωτισμού της οθόνης, ο συγκεκριμένος ακροδέκτης θα μπορούσε να συνδεθεί με κάποια θύρα του Arduino η οποία υποστηρίζει διαμόρφωση πλάτος παλμού (PWM).

- **Κ:** Ο ακροδέκτης αυτός είναι συνδεδεμένος με την κάθοδο του LED οπίσθιου φωτισμού και συνδέεται με τον ακροδέκτη γείωσης (GND) του Arduino.



Εικόνα 2.28: Το ηλεκτρονικό σχέδιο της πλακέτας με την LCD οθόνη 16x02 και το ενσωματωμένο πληκτρολόγιο [181].

Ωστόσο, είναι σημαντικός ο τρόπος με τον οποίο γίνεται ο προγραμματισμός του Arduino για τον έλεγχο τόσο της LCD οθόνης 16x02 όσο και του ενσωματωμένου πληκτρολογίου (Πίνακας 2.10). Επίσης, ο προγραμματισμός της LCD οθόνης γίνεται μέσω βιβλιοθήκης και αποτελεί ανεξάρτητο κομμάτι από τον προγραμματισμό για το ενσωματωμένο πληκτρολόγιο. Τέλος, η LCD οθόνη περιλαμβάνει το ολοκληρωμένο κύκλωμα HD44780 το οποίο είναι συμβατό με την βιβλιοθήκη LiquidCrystal που χρησιμοποιείται στον συγκεκριμένο προγραμματισμό [82].

Πίνακας 2.10: Ο κώδικας προγραμματισμού για τον έλεγχο της LCD οθόνης 16x02 και του ενσωματωμένου πληκτρολογίου [26].

Κώδικας Προγραμματισμού	Ανάλυση Κώδικα
#include <LiquidCrystal.h>	Εισαγωγή της βιβλιοθήκης LiquidCrystal στο πρόγραμμα. Δημιουργία αντικειμένου τύπου LyquidCrystal με όνομα lcd και καθορισμό

LiquidCrystal lcd(8, 9, 4, 5, 6, 7);	των ακροδεκτών RS, E, D4, D5, D6 & D7 της LCD οθόνης με τις αντίστοιχες ψηφιακές θύρες 8, 9, 4, 5, 6 & 7 του Arduino.
int x;	Δημιουργία μεταβλητής τύπου Integer με όνομα x.
void setup() {	Η συνάρτηση setup() η οποία τρέχει το μπλοκ εντολών της μία φορά κατά την εκκίνηση του Arduino.
pinMode (A0, INPUT);	Καθορισμός της αναλογικής θύρας A0 ως είσοδο.
lcd.begin(16,2);	Κλήση της συνάρτησης begin(columns, rows) μέσω του αντικειμένου lcd για των καθορισμό του πλήθους των στηλών και των γραμμών της LCD οθόνης.
lcd.setCursor(0,0);	Τοποθέτηση του κέρσορα της LCD οθόνης στην πρώτης στήλη και στην πρώτη γραμμή.
lcd.print("Hello World!");	Εμφάνιση του μηνύματος «Hello World!» στην LCD οθόνη.
lcd.setCursor(0,1);	Τοποθέτηση του κέρσορα στην πρώτη στήλη της δεύτερης γραμμής.
lcd.print("Press Key:");	Εμφάνιση του μηνύματος «Press Key:» στην LCD οθόνη.
}	
void loop() {	Η συνάρτηση loop(), η οποία επαναλαμβάνει το μπλοκ εντολών της μετά από κάθε ολοκλήρωσή της.
x=analogRead(A0);	Ανάγνωση της τιμής της αναλογικής θύρας εισόδου A0 και αποθήκευση στην μεταβλητή τύπου integer x.
lcd.setCursor(10,1);	Τοποθέτηση του κέρσορα της LCD οθόνης στην ενδέκατη στήλη της δεύτερης γραμμής.
if (x<60) {	Αν η τιμή της integer μεταβλητής x είναι μικρότερη από 60, τότε:
	Εμφάνισε στην LCD οθόνη το μήνυμα

lcd.print("Right!");
 	
}
	«Right!», ως ένδειξη ότι πατήθηκε το πλήκτρο Right από το ενσωματωμένο πληκτρολόγιο.

else if(x<200) {	Αλλιώς αν η τιμή της integer μεταβλητής x είναι μικρότερη από 200, τότε:

lcd.print("Up!");
 	
}
	Εμφάνισε στην LCD οθόνη το μήνυμα «Up!», ως ένδειξη ότι πατήθηκε το πλήκτρο Up από το ενσωματωμένο πληκτρολόγιο.

else if(x<400) {	Αλλιώς αν η τιμή της integer μεταβλητής x είναι μικρότερη από 400, τότε:

lcd.print("Down!");
 	
}
	Εμφάνισε στην LCD οθόνη το μήνυμα «Down!», ως ένδειξη ότι πατήθηκε το πλήκτρο Down από το ενσωματωμένο πληκτρολόγιο.

else if(x<600) {	Αλλιώς αν η τιμή της integer μεταβλητής x είναι μικρότερη από 600, τότε:

lcd.print("Left!");
 	
}
	Εμφάνισε στην LCD οθόνη το μήνυμα «Left!», ως ένδειξη ότι πατήθηκε το πλήκτρο Left από το ενσωματωμένο πληκτρολόγιο.

else if(x<800)
{
lcd.print("Select!");
}
 	
}
	Αλλιώς αν η τιμή της integer μεταβλητής x είναι μικρότερη από 800, τότε:
	Εμφάνισε στην LCD οθόνη το μήνυμα «Select!», ως ένδειξη ότι πατήθηκε το πλήκτρο Select από το ενσωματωμένο πληκτρολόγιο.

2.2.6 **Τροφοδοσία του συστήματος και τριπολικός διακόπτης (ON – OFF – ON)**

Ένα εξίσου βασικό σημείο για την υλοποίηση του συγκεκριμένου συστήματος είναι η τροφοδοσία των υποσυστημάτων. Ουσιαστικά, στην συγκεκριμένη περίπτωση, το κάθε υποσύστημα είναι ανεξάρτητο μιας και η τροφοδοσία του καθ’ ενός είναι ξεχωριστή. Πιο συγκεκριμένα, το Arduino τροφοδοτείται από μία επαναφορτιζόμενη μπαταρία 9Volt της Varta, οι πλακέτα ενίσχυσης και μετατόπισης του σήματος από το μικρόφωνο επαφής τροφοδοτείτε από δύο επαναφορτιζόμενες μπαταρίες της Varta των 9Volt (Εικόνα Εικόνα 2.29α) συνδεδεμένες σε σειρά, ενώ οι βηματικοί κινητήρες και οι πλακέτες οδήγησής τους τροφοδοτούνται από δύο επαναφορτιζόμενες μπαταρίες των 3.7Volt, τύπου 18650 της Samsung (Εικόνα Εικόνα 2.29β) και είναι συνδεδεμένες σε σειρά [141] [142]. Οι επαναφορτιζόμενες μπαταρίες 9Volt της Varta έχουν χωρητικότητα 200mAh και διαθέτουν τεχνολογία NiMH. Από την άλλη, οι επαναφορτιζόμενες μπαταρίες 18650 της

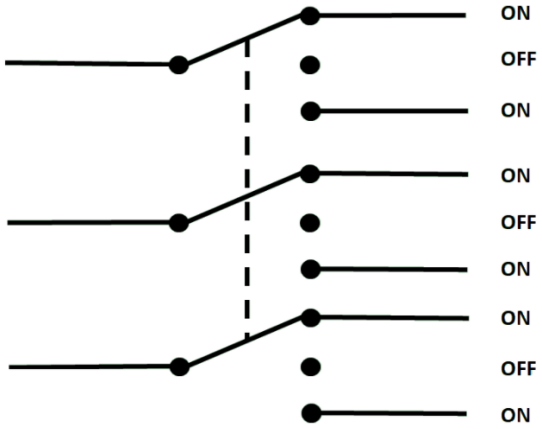
Samsung έχουν χωρητικότητα 3500mAh και διαθέτουν τεχνολογία Li-Ion. Επίσης, η χρήση των ανεξάρτητων τροφοδοσιών έχει το πλεονέκτημα της αύξησης των χρήσεων του συστήματος με μία μόνο φόρτιση.

α) β)

Εικόνα 2.29: α) Η μπαταρία των 9Volt της εταιρίας Varta, β) Η μπαταρία τύπου 18650 της Samsung των 3.7Volt [182].

Εν συνεχεία, για την ταυτόχρονη τροφοδοσία των υποσυστημάτων, χρησιμοποιείτε ένας τριπολικός διακόπτης τριών θέσεων (ON – OFF – ON) (Εικόνα Εικόνα 2.30). Ουσιαστικά, ο συγκεκριμένος διακόπτης έχει τρεις εισόδους (Triple Pole) και δύο εξόδους (Double Throw) καθώς και μία «νεκρή» θέση στην οποία δεν πραγματοποιείτε καμία επαφή (Εικόνα Εικόνα 2.31) [143]. Επίσης, ο συγκεκριμένος τύπος διακόπτη, έχει ανοχή επαφών 5A στα 28Volt (DC), περιλαμβάνει ακροδέκτες για συγκόλληση καλωδίων, η θερμοκρασία ορθής λειτουργίας κυμαίνεται από -35^{o}-80^{o}C και η μηχανική του αντοχή φτάνει τους 40000 κύκλους. Επιπλέον, με τον τριπολικό διακόπτη επιτυγχάνεται η ταυτόχρονη τροφοδοσία των τριών υποσυστημάτων (Arduino, οδηγοί και βηματικοί κινητήρες και πλακέτα ενίσχυσης και μετατόπισης του ήχου). Με αυτό τον τρόπο, το σύστημα αυτομάτου κουρδίσματος ενεργοποιείται μαζικά χωρίς να επιβαρύνεται μία μόνο πηγή τροφοδοσίας.

Εικόνα 2.30: Ο τριπολικός διακόπτης τριών θέσεων [183].



Εικόνα 2.31: Το ηλεκτρολογικό σχέδιο ενός τριπολικού διακόπτη με δύο εξόδους και μία «νεκρή» θέση [184].

2.3 Τρισδιάστατοι εκτυπωτές και τρισδιάστατη εκτύπωση

Ένα από τα σημαντικότερα μέρη αυτής της διπλωματικής εργασίας είναι η στήριξη όλου του συστήματος αυτομάτου κουρδίσματος επάνω στην κεφαλή της ηλεκτρικής κιθάρας. Σε αυτή την ενότητα λοιπόν, γίνεται μια αναφορά στα χαρακτηριστικά των τρισδιάστατων εκτυπωτών που χρησιμοποιήθηκαν για την εκπόνηση της διπλωματικής εργασίας. Επίσης, γίνεται επεξήγηση του τρόπου σχεδιασμού των βάσεων μέσα από σκίτσα των εξαρτημάτων. Ωστόσο, ο τρόπος με τον οποίο θα στηριχτεί το όλο σύστημα επάνω στην κεφαλή της ηλεκτρικής κιθάρας είναι καίριας σημασίας. Δηλαδή, με την ενσωμάτωση του συστήματος αυτομάτου κουρδίσματος επάνω στην κιθάρα θα πρέπει να παραμένει εύχρηστο και να μην εμποδίζει τον χρήστη κατά την χρήση του οργάνου.

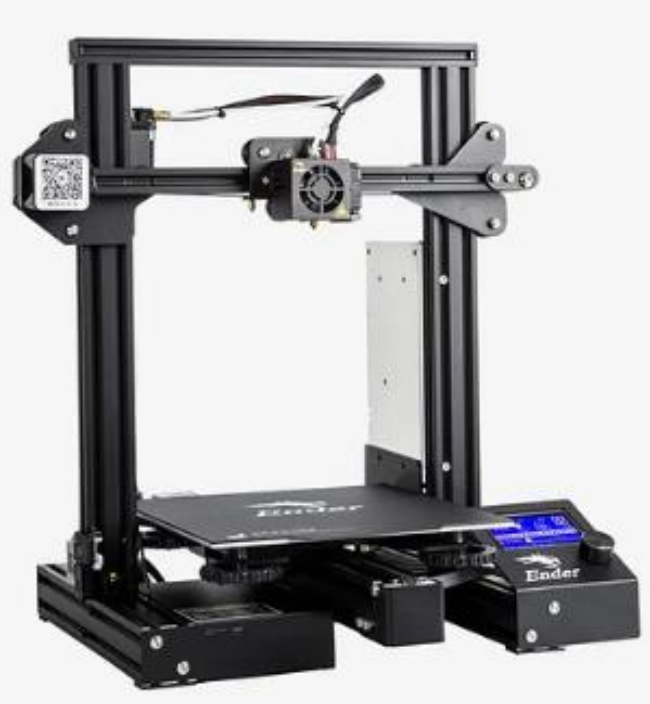
2.3.1 Τρισδιάστατοι εκτυπωτές (3D Printers)

Για την κατασκευή των βάσεων του συστήματος χρησιμοποιήθηκαν δύο τύποι τρισδιάστατων εκτυπωτών. Πιο συγκεκριμένα, ο πρώτος εκτυπωτής που χρησιμοποιήθηκε ήταν ο Creality Ender-3 Pro, ενώ ο δεύτερος ήταν ο Creality Ender-3 S1. Σε αυτό το σημείο πρέπει να σημειωθεί ότι επειδή ο Creality Ender-3 Pro βρισκόταν στο εργαστήριο του πανεπιστημίου και για λόγους εξοικονόμησης χρόνου αποκτήθηκε ο Creality Ender-3 S1. Πάραυτα, τα τρισδιάστατα σχέδια παραμένουν ίδια και στους δύο εκτυπωτές, το μόνο που αλλάζει είναι τα χαρακτηριστικά τους τα οποία αναλύονται στην συγκεκριμένη ενότητα.

2.3.1.1 Ο τρισδιάστατος εκτυπωτής Creality Ender-3 Pro

Αρχικά, ο τρισδιάστατος εκτυπωτής Creality Ender-3 Pro χρησιμοποιεί την τεχνική της μοντελοποίησης με συντηγμένη εναπόθεση (FDM) κάνοντας χρήση μίας κεφαλής Extruder (Εικόνα Εικόνα 2.32). Πιο συγκεκριμένα, ο Ender-3 Pro έχει διαστάσεις 440mm μήκος, 420mm πλάτος και 465mm ύψος ενώ το καθαρό βάρος του είναι περίπου 7 kg. Ο συγκεκριμένος τρισδιάστατος εκτυπωτής δύναται να εκτυπώσει κομμάτια τα οποία σχεδιαστικά φτάνουν μέχρι τα 220mm πλάτος, 220mm μήκος και 250mm ύψος. Επιπλέον, η ταχύτητα του Ender-3 Pro σε μία κανονική εκτύπωση κυμαίνεται από 30-60mm/sec. Γενικά όμως ο συγκεκριμένος εκτυπωτής έχει την δυνατότητα να φτάσει σε ταχύτητα

εκτύπωσης μέχρι και 180mm/sec. Επιπροσθέτως, η διάμετρος του μπεκ που έχει ο Ender-3 Pro κανονικά είναι 0.4mm ενώ σε μερικές περιπτώσεις, μπορεί να είναι 0.3mm ή 0.2mm. Επίσης, η ακρίβεια εκτύπωσης του συγκεκριμένου εκτυπωτή είναι ±0.1mm γεγονός το οποίο δίνει την δυνατότητα στον χρήστη να σχεδιάσει πολύ μικρές λεπτομέρειες στο τρισδιάστατο σχέδιο. Τα αρχεία τα οποία μπορεί να αναγνωρίσει ο Ender-3 Pro για να ξεκινήσει μία εκτύπωση πρέπει να έχουν την μορφή .STL ή .OBJ ή .AMF ή .GCODE και μπορούν να μεταφερθούν από τον υπολογιστή στον εκτυπωτή μέσω της χρήσης κάρτας SD ή μέσω διαδικτύου. Οι εφαρμογές που μπορούν να δημιουργήσουν τέτοιας μορφής αρχεία ονομάζονται λογισμικά τεμαχισμού διότι τεμαχίζουν το τρισδιάστατο σχέδιο σε επίπεδα εκτύπωσης. Τέτοια λογισμικά τεμαχισμού είναι το UltiMaker Cura, το Repetier-Host και το Simplify3D. Όσον αφορά τα υλικά που μπορεί να χρησιμοποιήσει ο συγκεκριμένος εκτυπωτής βρίσκονται σε μορφή νήματος και είναι το PLA, το ABS, το TPU, το ξύλο και ο χαλκός. Τέλος, ο εκτυπωτής αυτός παρέχει κάποια εργαλεία επεξεργασία του τρισδιάστατου κομματιού. Πιο συγκεκριμένα, παρέχει μία σπάτουλα για την αποκόλληση του κομματιού από την πλατφόρμα σχεδίασης, ένα κοφτάκι για την αφαίρεση των στηριγμάτων από το τρισδιάστατο κομμάτι και μία βελόνα καθαρισμού του εσωτερικού του Extruder από το υλικό [144].



Εικόνα 2.32: Ο τρισδιάστατος εκτυπωτής Creality Ender-3 Pro [185].

2.3.1.2 Ο τρισδιάστατος εκτυπωτής Creality Ender-3 S1

Εν συνεχεία, σε αυτή την ενότητα γίνεται αναφορά στα χαρακτηριστικά του δεύτερου τρισδιάστατου εκτυπωτή και πιο συγκεκριμένα του Creality Ender-3 S1. Πιο συγκεκριμένα, ο Ender-3 S1 χρησιμοποιεί και αυτός την τεχνική της μοντελοποίησης με συντηγμένη εναπόθεση (FDM) κάνοντας χρήση μίας κεφαλής Extruder (Εικόνα Εικόνα 2.33). Επίσης, οι διαστάσεις του συγκεκριμένου εκτυπωτή είναι μεγαλύτερες από του Ender-3 Pro μιας και το μήκος του είναι 487mm, το πλάτος του 453mm ενώ το ύψος του φτάνει τα 622mm. Ωστόσο, παρόλο που ο εκτυπωτής Ender-3 S1 είναι πιο ογκώδης από τον προηγούμενο, οι μέγιστες διαστάσεις εκτύπωσης ενός τρισδιάστατου σχεδίου είναι

σχεδόν ίδιες με εξαίρεση ύψος το οποίο είναι μεγαλύτερο σε αυτή την περίπτωση. Ουσιαστικά, ο συγκεκριμένος εκτυπωτής μπορεί να τυπώσει ένα τρισδιάστατο σχέδιο το οποίο φτάνει μέχρι τα 220mm μήκος, 220mm πλάτος και 270mm ύψος. Επιπλέον, το βάρος του Ender-3 S1 φτάνει μέχρι και τα 9.1Kg διότι εκτός του ότι είναι πιο ογκώδης χρησιμοποιεί κι' άλλα εξαρτήματα σε σχέση με τον Ender-3 Pro. Πιο συγκεκριμένα, πάνω στην κεφαλή Extruder περιλαμβάνετε ένας ανεμιστήρας ψύξης του πλαστικού που ακουμπά επάνω στην πλατφόρμα εκτύπωσης, καθώς και ένας αισθητήρας μέτρησης του επιπέδου της πλατφόρμας CR Touch. Επίσης, περιλαμβάνει έναν αισθητήρα ανίχνευσης υλικού και έναν δεύτερο βηματικό κινητήρα για τον καλύτερο έλεγχο του Z άξονα. Ουσιαστικά, με τον αισθητήρα μέτρησης του επιπέδου της πλατφόρμας CR Touch, ο εκτυπωτής συλλέγει μετρήσεις σε διάφορα σημεία της πλατφόρμας τα οποία βοηθούν το χρήστη να καταλάβει αν η πλατφόρμα θέλει ρύθμιση. Επίσης, με τον αισθητήρα CR Touch, η τρισδιάστατη εκτύπωση γίνεται με μεγαλύτερη ακρίβεια. Επιπροσθέτως, ο αισθητήρας υλικού που διαθέτει ο συγκεκριμένος εκτυπωτής έχει σκοπό την διακοπή της εκτύπωσης σε περίπτωση που είτε τελειώσει είτε κοπεί το νήμα υλικού. Επίσης, ο Ender-3 S1στη περίπτωση διακοπής της εκτύπωσης εξαιτίας της έλλειψης υλικού, παρέχει την δυνατότητα συνέχισης της εκτύπωσης από το σημείο που έγινε η διακοπή αφού πρώτα αντικατασταθεί το νήμα υλικού στον αισθητήρα. Τα υλικά που μπορεί να χρησιμοποιήσει ο συγκεκριμένος εκτυπωτής πρέπει να είναι σε μορφή νήματος διαμέτρου 1.75mm και να είναι είτε PLA είτε TPU είτε PETG είτε ABS [102].

Εικόνα 2.33: Ο τρισδιάστατος εκτυπωτής Creality Ender-3 S1 [186].

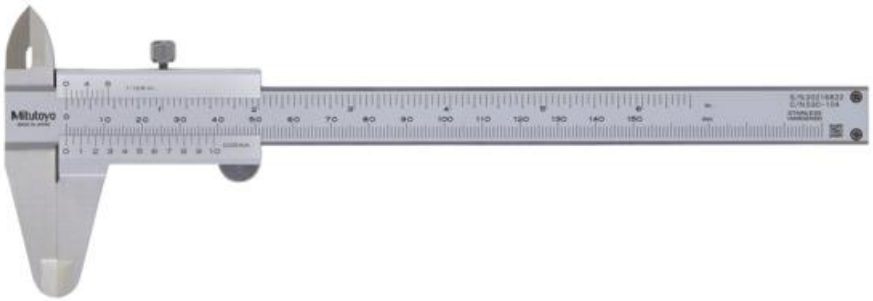
Εν συνεχεία, ο Ender-3 S1 μπορεί να καθορίσει το πάχος του κάθε στρώματος της τρισδιάστατης εκτύπωσης από 0.1mm έως 0.35mm, ενώ η διάμετρος του μπεκ του Extruder είναι 0.4mm και η ακρίβεια εκτύπωσης του εκτυπωτή φτάνει το ±0.1mm. Συνεπώς, με αυτή την ακρίβεια εκτύπωσης, ο χρήστης μπορεί να σχεδιάσει τρισδιάστατα σχέδια με μεγάλη λεπτομέρεια. Ακόμα, η τυπική ταχύτητα εκτύπωσης του συγκεκριμένου εκτυπωτή φτάνει μέχρι τα 110mm/sec, ενώ η μέγιστη ταχύτητα που μπορεί να φτάσει είναι 160mm/sec. Επίσης, τα τρισδιάστατα σχέδια πρέπει να έχουν μορφή .STL ή .OBJ ή .AMF ή .GCODE για να μπορέσει να ξεκινήσει την εκτύπωση ο τρισδιάστατος εκτυπωτής. Τα λογισμικά τεμαχισμού που μπορούν να δημιουργήσουν τέτοια αρχεία είναι το Creality Print, το Creality Slicer, το Ultimaker Cura και το Simplify3D. Ωστόσο, τα αρχεία μπορούν να μεταφερθούν στον εκτυπωτή είτε μέσω κάρτας SD είτε μέσω της

θύρας Type-C USB που διαθέτει ο εκτυπωτής. Τέλος, ο εκτυπωτής αυτός παρέχει κάποια εργαλεία, όπως μία σπάτουλα, ένα κοφτάκι και μία βελώνα. Πιο συγκεκριμένα, η σπάτουλα χρησιμοποιείται για την αποκόλληση του κομματιού από την πλατφόρμα και το κοφτάκι για την αφαίρεση των στηριγμάτων από το κομμάτι, ενώ η βελόνα χρησιμοποιείτε για τον καθαρισμό του εσωτερικού του Extruder από το υλικό [102].

2.3.2 **Τρισδιάστατη σχεδίαση**

Εν συνεχεία, είναι πολύ σημαντικός ο τρόπος με τον οποίο θα στηριχτεί όλο το σύστημα αυτομάτου κουρδίσματος επάνω στην κιθάρα. Πιο συγκεκριμένα, το σύστημα θα πρέπει να τοποθετηθεί με τέτοιον τρόπο που θα παραμένει εύχρηστο και δεν θα εμποδίζει τον χρήστη κατά την χρήση του οργάνου. Σε αυτή την ενότητα, εξηγείται βήμα-βήμα η λογική και ο τρόπος πίσω από την σχεδίαση των τρισδιάτατων βάσεων για την στήριξη όλου του συστήματος αυτομάτου κουρδίσματος επάνω στην κεφαλή της κιθάρας.

Καταρχήν, θα πρέπει να επιλεχτεί το κατάλληλο σημείο επάνω στην κεφαλή της κιθάρας το οποίο θα αντέχει το βάρος του συστήματος και δεν θα επηρεάζει τον χρήστη κατά το παίξιμο του οργάνου. Επίσης, το σημείο αυτό θα πρέπει να διαθέτει τον απαραίτητο χώρο για την τοποθέτηση ολόκληρου του συστήματος. Συνεπώς, μετά από αρκετή σκέψη, αποφασίστηκε όλο το σύστημα να τοποθετηθεί επάνω στα κλειδιά της κιθάρας. Πιο συγκεκριμένα, πάνω στο κάθε ένα κλειδί της κιθάρας θα τοποθετηθεί ένας βηματικός κινητήρας ο οποίος με την χρήση ενός αντάπτορα θα περιστρέφει το κλειδί. Επομένως, επειδή το σώμα του βηματικού κινητήρα πρέπει να παραμένει σταθερό θα πρέπει οι κινητήρες να συγκρατούνται μεταξύ τους και πάνω σε αυτούς θα στηρίζονται όλες οι υπόλοιπες βάσης. Επομένως, για την ακριβέστερη μέτρηση των διαστάσεων τόσο των εξαρτημάτων όσο και των διαφόρων σημείων πάνω στο όργανο και πάνω στις βάσεις, χρησιμοποιήθηκε ένα αναλογικό παχύμετρο (Εικόνα Εικόνα 2.34) [145].

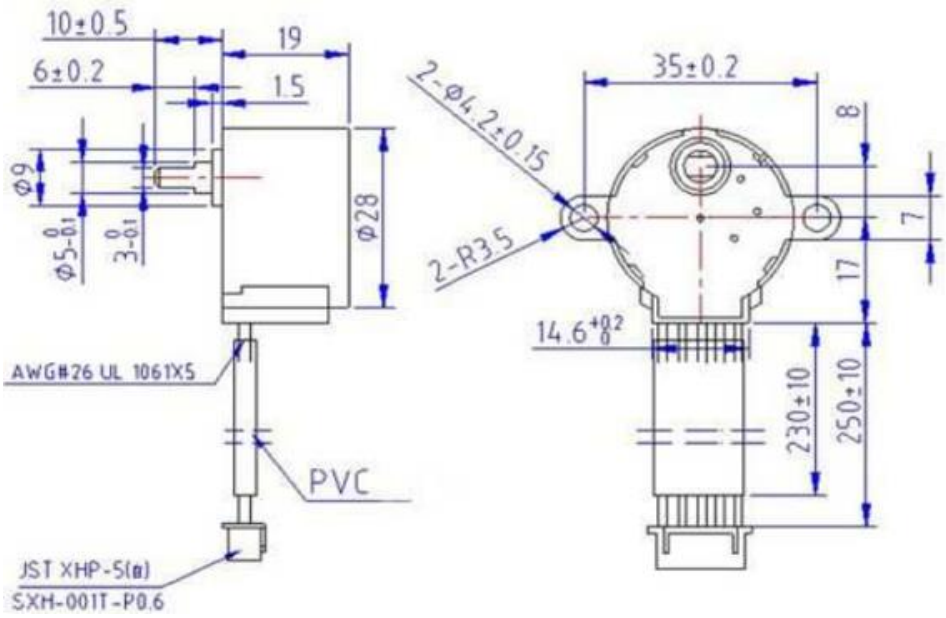


Εικόνα 2.34: Το αναλογικό παχύμετρο με το οποίο μετρήθηκαν διαστάσεις [187].

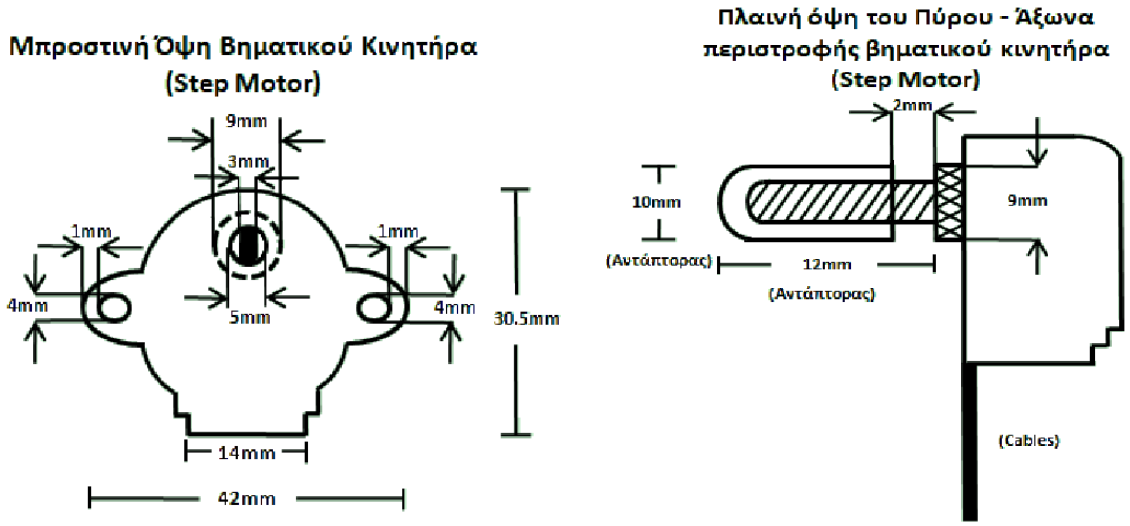
2.3.2.1 **Ο αντάπτορας μεταξύ κλειδιού και βηματικού κινητήρα**

Για την επίτευξη της σκέψης αυτής, πρώτα απ’ όλα σχεδιάστηκε ο αντάπτορας ο οποίος συνδέει τον άξονα του βηματικού κινητήρα με το κλειδί της κιθάρας. Σκοπός του αντάπτορα είναι να στηρίξει τον βηματικό κινητήρα και να μεταφέρει την περιστροφή του άξονά του στο αντίστοιχο κλειδί της κιθάρας. Επομένως, για την σχεδίαση του αντάπτορα πρέπει να ληφθούν υπ’ όψιν οι διαστάσεις τόσο του κλειδιού της κιθάρας όσο και του άξονα του βηματικού κινητήρα. Ωστόσο, ο κατασκευαστής του βηματικού κινητήρα 28BYJ-48 δίνει το αντίστοιχο σκίτσο με τις διαστάσεις του άξονα καθώς και τις

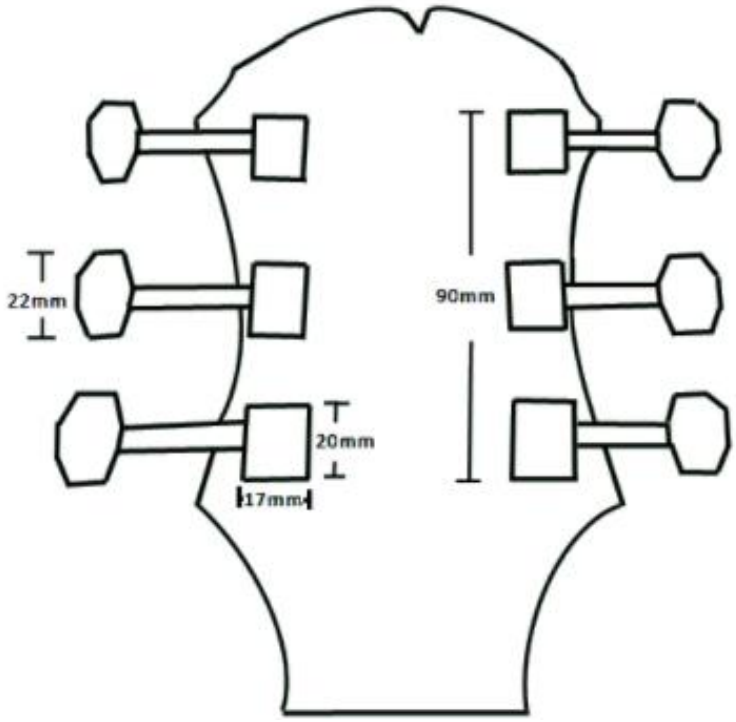
αποκλίσεις του (Εικόνα Εικόνα 2.35) [132]. Επομένως, για μεγαλύτερη ακρίβεια οι διαστάσεις του άξονα μετρήθηκαν με το παχύμετρο και δημιουργήθηκε ένα πρόχειρο σχέδιο στο Word το οποίο περιλαμβάνει τις μετρούμενες διαστάσεις του βηματικού κινητήρα και του άξονά του (Εικόνα Εικόνα 2.36). Επίσης, δημιουργήθηκε άλλο ένα πρόχειρο σχέδιο στο Word το οποίο περιλαμβάνει τις διαστάσεις των κλειδιών επάνω στην κιθάρα (Εικόνα Εικόνα 2.37). Οι διαστάσεις αυτές μετρήθηκαν με την χρήση του παχυμέτρου.



Εικόνα 2.35: Το σκίτσο του κατασκευαστή των βηματικών κινητήρων 28BYJ-48 με τις διαστάσεις του άξονα [188].

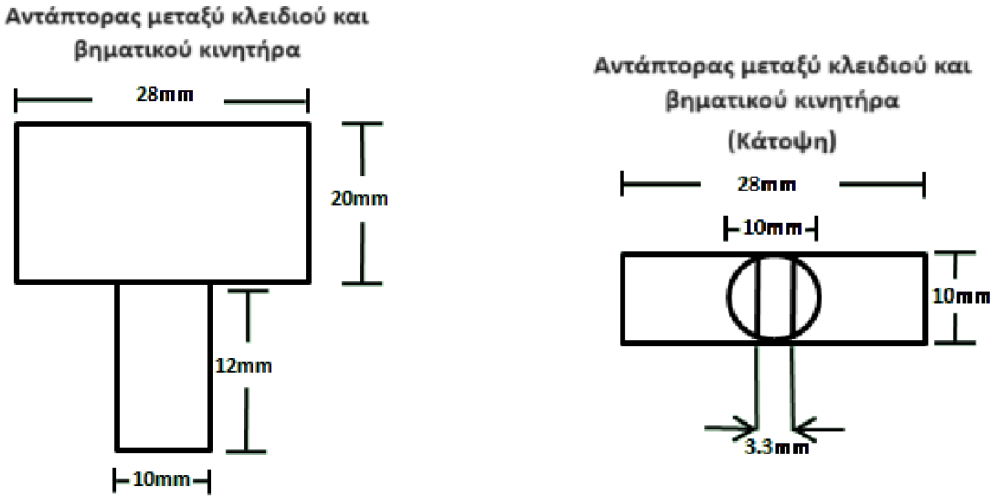


Εικόνα 2.36: Το πρόχειρο σχέδιο με τις διαστάσεις του βηματικού κινητήρα και του άξονά του [189].

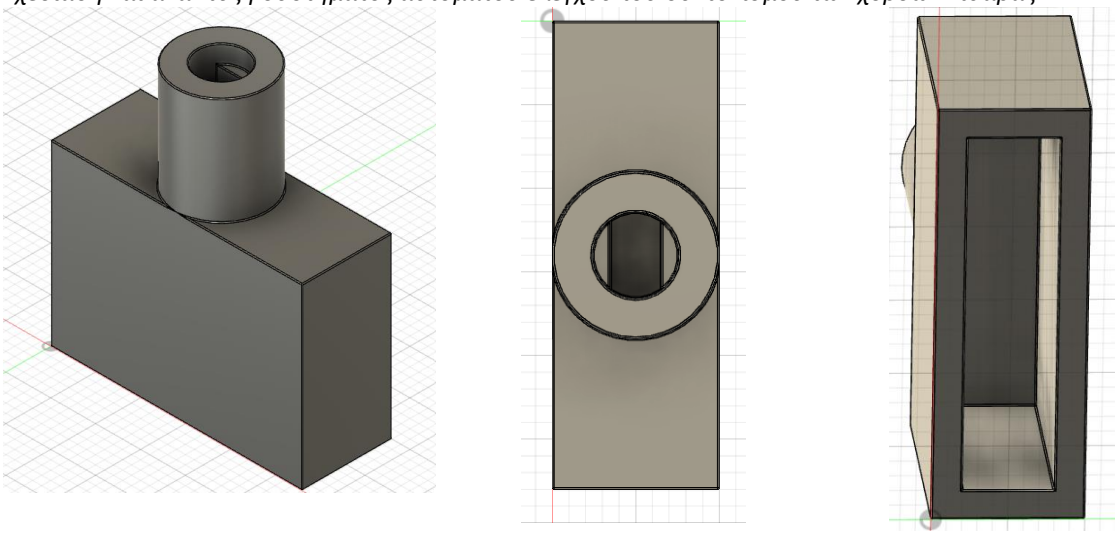


Εικόνα 2.37: Το πρόχειρο σχέδιο με τις διαστάσεις των κλειδιών της κιθάρας [190].

Επομένως, σύμφωνα με τις μετρούμενες διαστάσεις του άξονα αλλά και των κλειδιών της κιθάρας σχεδιάστηκε πρόχειρα στο Word η μορφή που θα έχει ο αντάπτορας (Εικόνα Εικόνα 2.38). Ο σχεδιασμός αυτός, επιτρέπει την εφαρμογή του αντάπτορα τόσο στον άξονα του βηματικού κινητήρα όσο και στα κλειδιά της κιθάρας. Επίσης, με τον συγκεκριμένο σχεδιασμό προσφέρεται στήριξη του κάθε βηματικού κινητήρα επάνω στο αντίστοιχο κλειδί και το σημαντικότερο είναι ότι όλοι οι βηματικοί κινητήρες βρίσκονται στο ίδιο ύψος. Με αυτόν τον τρόπο, η σύνδεση των βηματικών κινητήρων μεταξύ τους, όπως παρουσιάζεται στην επόμενη ενότητα, γίνεται πιο εύκολα. Εν συνεχεία, αφού αποφασίστηκαν οι διαστάσεις του αντάπτορα, τότε ο τελικός σχεδιασμός του έγινε στο Autodesk Fusion (Εικόνα Εικόνα 2.39). Πιο συγκεκριμένα, ο σχεδιασμός του αντάπτορα στα σημεία εφαρμογής με το κλειδί και με τον άξονα του βηματικού κινητήρα έγινε με τέτοια ακρίβεια, ώστε να συγκρατούνται σφικτά χωρίς την χρήση βίδας.

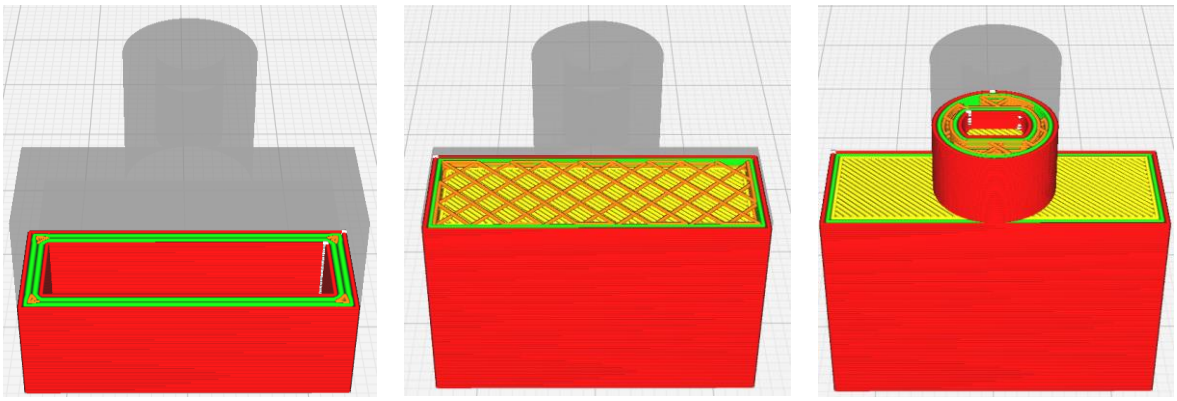


Εικόνα 2.38: Το πρόχειρο σχέδιο του αντάπτορα το οποίο πραγματοποιήθηκε σύμφωνα με τις μετρούμενες διαστάσεις [191].



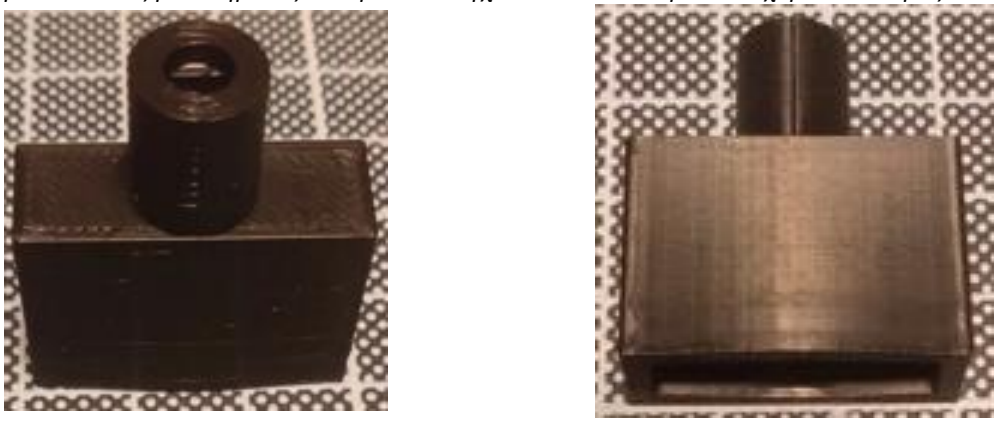
Εικόνα 2.39: Ο τρισδιάστατος σχεδιασμός του αντάπτορα στο Autodesk Fusion [192].

Έπειτα, αφού ολοκληρώθηκε η τρισδιάστατη σχεδίαση του αντάπτορα, έγινε εξαγωγή του σχεδίου σε ένα αρχείο με μορφή .STL. Με αυτόν τον τρόπο το τρισδιάστατο σχέδιο μπορεί να εισαχθεί στο λογισμικό τεμαχισμού UltiMaker Cura για περεταίρω επεξεργασία. Επειδή, ο συγκεκριμένος αντάπτορας θα δέχεται ροπές και δυνάμεις αντίστασης επιλέχθηκε ένα σχετικά πυκνό γέμισμα ανάμεσα στα τοιχώματα και μεταξύ των στρωμάτων (Εικόνα Εικόνα 2.40). Με αυτόν τον τρόπο, ο αντάπτορας δεν κινδυνεύει να παραμορφωθεί αλλά ούτε και να σπάσει κατά την περιστροφή των κλειδιών.



Εικόνα 2.40: Τα διάφορα στρώματα και ο τρόπος γεμίσματος μεταξύ των τοιχωμάτων του αντάπτορα στο λογισμικό τεμαχισμού UltiMaker Cura [193].

Έπειτα, μετά την ολοκλήρωση της επεξεργασίας στο πρόγραμμα τεμαχισμού έγινε εξαγωγή του τρισδιάστατου σχεδίου σε ένα αρχείο με μορφή .GCODE. Το αρχείο αυτό αποθηκεύτηκε στην κάρτα SD ώστε να εισαχθεί στην συνέχεια στον τρισδιάστατο εκτυπωτή και να ξεκινήσει η τρισδιάστατη εκτύπωση. Ωστόσο, για την εκτύπωση της συγκεκριμένης μορφής αντάπτορα χρησιμοποιήθηκε βιοδιασπώμενο πλαστικό υλικό και πιο συγκεκριμένα PLA μαύρου χρώματος σε μορφή νήματος (Εικόνα Εικόνα 2.41). Τέλος, ο αντάπτορας τοποθετήθηκε στον άξονα του βηματικού κινητήρα και στην συνέχεια επάνω στο κλειδί της κιθάρας (Εικόνα Εικόνα 2.42).



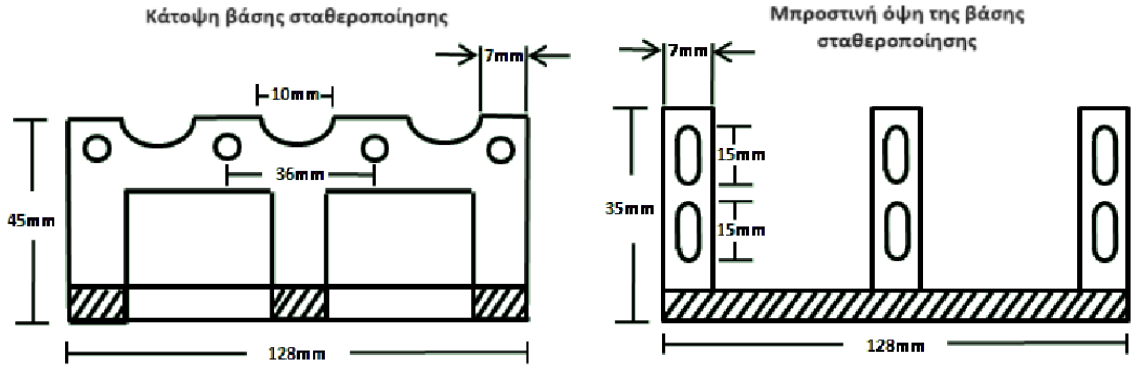
Εικόνα 2.41: Ο αντάπτορας μετά την ολοκλήρωση της τρισδιάστατης εκτύπωσης [194].

Εικόνα 2.42: Εφαρμογή του βηματικού κινητήρα επάνω στο κλειδί της κιθάρας με την χρήση του αντάπτορα [195].

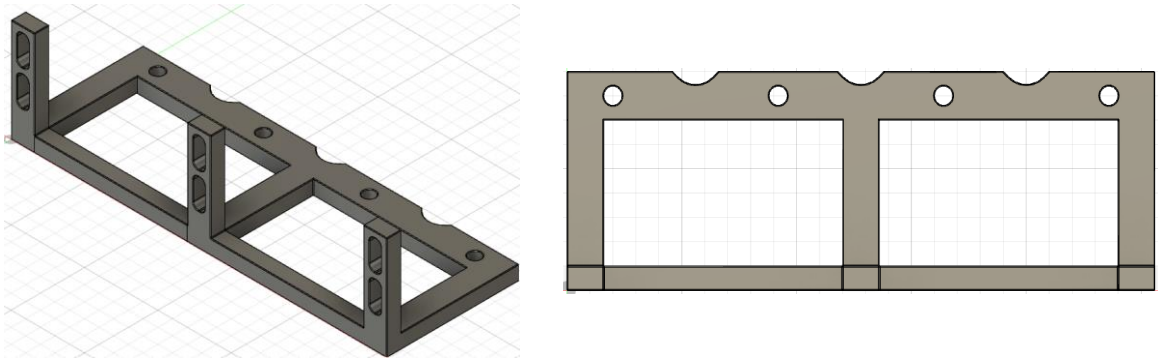
2.3.2.2 Οι βάσεις για την σύνδεση και σταθεροποίηση των βηματικών κινητήρων, καθώς και για την στήριξη των υπόλοιπων βάσεων

Μετά την δημιουργία του αντάπτορα μεταξύ κλειδιού και βηματικού κινητήρα, έπρεπε να δημιουργηθεί ένας τύπος βάσης ο οποίος αφενός θα κρατάει σταθερούς τους βηματικούς κινητήρες, και αφετέρου θα αποτελεί στήριξη για τις βάσεις των υπολοίπων εξαρτημάτων. Επίσης, ο συγκεκριμένος τύπος βάσης θα πρέπει να είναι αρκετά ανθεκτικός και να μην λυγίζει με το βάρος των υπολοίπων βάσεων και εξαρτημάτων. Επιπλέον, η βάση θα πρέπει να έχει τέτοιες διαστάσεις ώστε να μην εμποδίζει τον χρήστη κατά την χρήση του οργάνου. Ακόμα, η βάση αυτή θα πρέπει να δημιουργηθεί εις διπλούν διότι, η κιθάρα στην οποία θα ενσωματωθεί το σύστημα έχει τρία κλειδιά στο επάνω μέρος της κεφαλής και άλλα τρία κλειδιά στο κάτω μέρος της κεφαλής. Συνεπώς, υπάρχουν τρεις βηματικοί κινητήρες από την μία πλευρά και άλλοι τρεις από την άλλοι που χρειάζονται σταθεροποίηση.

Επομένως, μετά από αρκετή σκέψη και σύμφωνα πάντα με τις μετρήσεις των διαστάσεων των βηματικών κινητήρων σχεδιάστηκε πρόχειρα στο Word η μορφή της βάσης σταθεροποίησης των βηματικών κινητήρων και στήριξης των υπολοίπων βάσεων (Εικόνα Εικόνα 2.43). Σύμφωνα με την μορφή της βάσης αυτής, ο κάθε ένας βηματικός κινητήρας συνδέεται με τον επόμενο πάνω στην βάση και συγκρατούνται με την χρήση βίδας, ροδέλας και παξιμαδιού τύπου M3. Επίσης, στην βάση αυτή έχει σχεδιαστεί μια ημικυκλική εγκοπή ώστε να εφαρμόζεται το χείλος που έχει ο βηματικός κινητήρας στον άξονά του. Ακόμα, στην βάση έχουν σχεδιαστεί τρία άκρα με δύο εγκοπές το κάθε ένα, με σκοπό την στήριξη των υπολοίπων βάσεων και των αντίστοιχων εξαρτημάτων τους επάνω στην κεφαλή της κιθάρας. Στην συνέχεια, πραγματοποιήθηκε η σχεδίαση της τρισδιάστατης βάσης σύνδεσης και σταθεροποίησης στο Autodesk Fusion σύμφωνα με το πρόχειρο σχέδιο (Εικόνα Εικόνα 2.44). Επίσης, στο Autodesk Fusion η βάση σχεδιάστηκε να έχει ένα ικανοποιητικό πάχος ώστε να αντέχει το βάρος όλων των υπολοίπων βάσεων χωρίς να υπάρχει κίνδυνος να σπάσει. Ακόμα, έχουν σχεδιαστεί ενώσεις επάνω στην βάση με σκοπό το βάρος από τις βάσεις που θα σταθεροποιηθούν επάνω της στην πορεία, να κατανέμεται με τέτοιον τρόπο που δεν θα υπάρχει κίνδυνος παραμόρφωσης. Επιπροσθέτως, η σχεδίαση αυτή επιτυγχάνει αισθητική και οικονομία σε κόστος, διότι δεν απαιτείται μεγάλη κατανάλωση υλικού κατά την εκτύπωση.

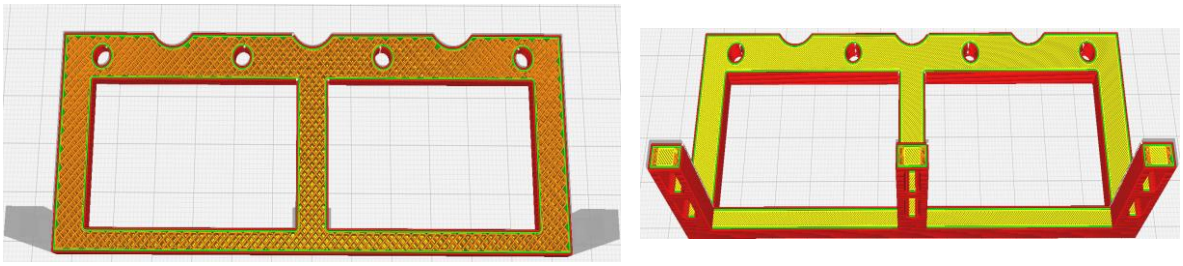


Εικόνα 2.43: Το πρόχειρο σχέδιο της βάσης σταθεροποίησης των βηματικών κινητήρων και στήριξης των υπολοίπων εξαρτημάτων [196].



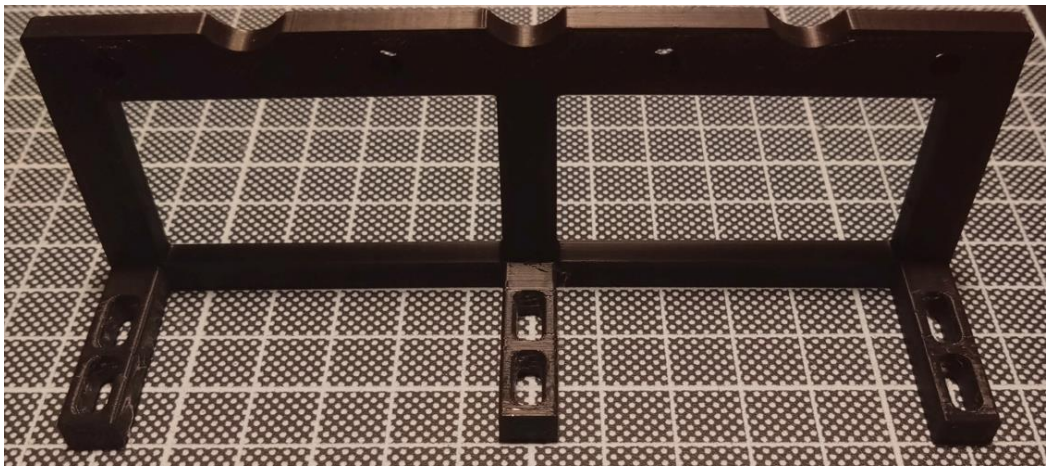
Εικόνα 2.44: Το τρισδιάστατο σχέδιο της βάσης σταθεροποίησης των βηματικών κινητήρων και στήριξης των υπόλοιπων βάσεων στο Autodesk Fusion [197].

Αφού ολοκληρώθηκε και η σχεδίαση στο Autodesk Fusion έγινε εξαγωγή του τρισδιάστατου σχεδίου σε μορφή .STL. Έπειτα, το αρχείο αυτό εισήχθη στο λογισμικό τεμαχισμού UltiMaker Cura για περεταίρω επεξεργασία. Πιο συγκεκριμένα, κατά την επεξεργασία του τρισδιάστατου σχεδίου στο λογισμικό τεμαχισμού, επιλέχθηκε ένα αρκετά πυκνό γέμισμα μεταξύ των στρωμάτων ώστε η βάση να είναι αρκετά στιβαρή και να μην παραμορφώνεται από το βάρος των υπολοίπων βάσεων και των εξαρτημάτων (Εικόνα Εικόνα 2.45).

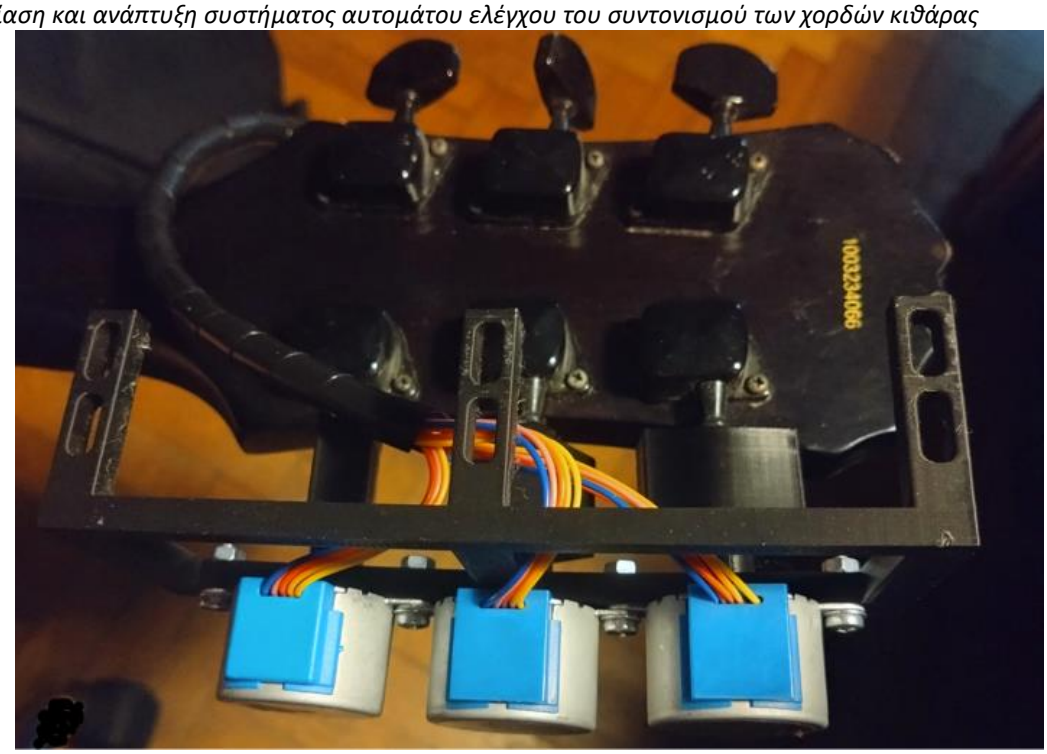


Εικόνα 2.45: Τα διάφορα στρώματα και ο τρόπος γεμίσματος μεταξύ των στρωμάτων της βάσης σταθεροποίησης των βηματικών κινητήρων και στήριξης των υπόλοιπων βάσεων στο λογισμικό τεμαχισμού UltiMaker Cura [198].

Τέλος, αφού ολοκληρώθηκε η επεξεργασία στο λογισμικό τεμαχισμού UltiMaker Cura, έγινε εξαγωγή του τρισδιάστατου σχεδίου σε ένα αρχείο με μορφή .GCODE. Το αρχείο αυτό αποθηκεύτηκε στην κάρτα SD για να εισαχθεί στην συνέχεια στον τρισδιάστατο εκτυπωτή και να ξεκινήσει η εκτύπωση (Εικόνα Εικόνα 2.46). Για την εκτύπωση των δύο αυτών βάσεων χρησιμοποιήθηκε βιοδιασπώμενο πλαστικό και πιο συγκεκριμένα μαύρο PLA σε μορφή νήματος. Μετά την ολοκλήρωση της εκτύπωσης των δύο βάσεων, βιδώθηκαν επάνω οι βηματικοί κινητήρες με την χρήση βιδών, ροδελών και παξιμαδιών τύπου M3 και έπειτα τοποθετήθηκαν όλα μαζί επάνω στην κεφαλή της κιθάρας (Εικόνα Εικόνα 2.47).



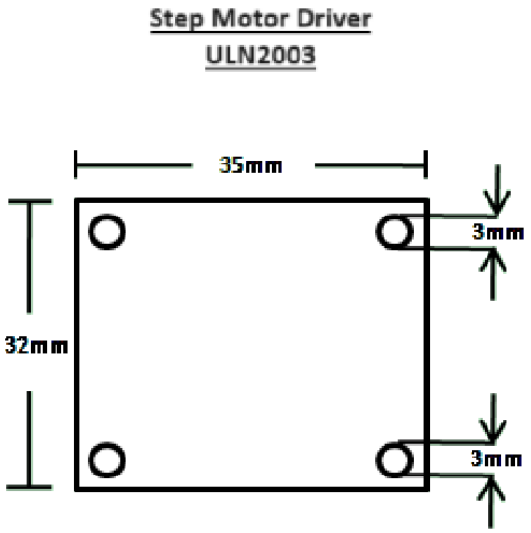
Εικόνα 2.46: Η εκτυπωμένη βάση σταθεροποίησης των βηματικών κινητήρων και στήριξης των υπολοίπων βάσεων [199].



Εικόνα 2.47: Η βάση σταθεροποίησης των βηματικών κινητήρων και στήριξης των υπολοίπων βάσεων τοποθετημένη επάνω στην κεφαλή της κιθάρας [200].

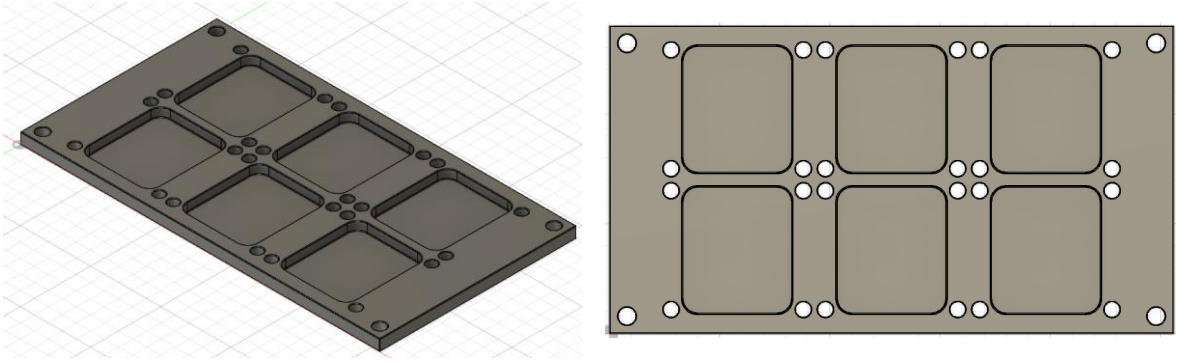
2.3.2.3 Η βάση των πλακετών οδήγησης των βηματικών κινητήρων και οι σύνδεσμοι στήριξης

Σε αυτό το σημείο γίνεται ο σχεδιασμός της βάσης των πλακετών οδήγησης των βηματικών κινητήρων. Πιο συγκεκριμένα, η βάση αυτή πρέπει να έχει αρκετό χώρο για να χωρέσει τις έξι πλακέτες οδήγησης ULN2003. Επιπλέον, ο σχεδιασμός της βάσης πρέπει να γίνει με τέτοιον τρόπο ώστε να μην προσθέτει βάρος στην κεφαλή της κιθάρας και να μην επηρεάζει τον χρήστη κατά την χρήση του οργάνου. Επομένως, το πρώτο βήμα είναι η μέτρηση των διαστάσεων των πλακετών οδήγησης των βηματικών κινητήρων και η δημιουργία ενός πρόχειρου σχεδίου στο Word (Εικόνα Εικόνα 2.48).

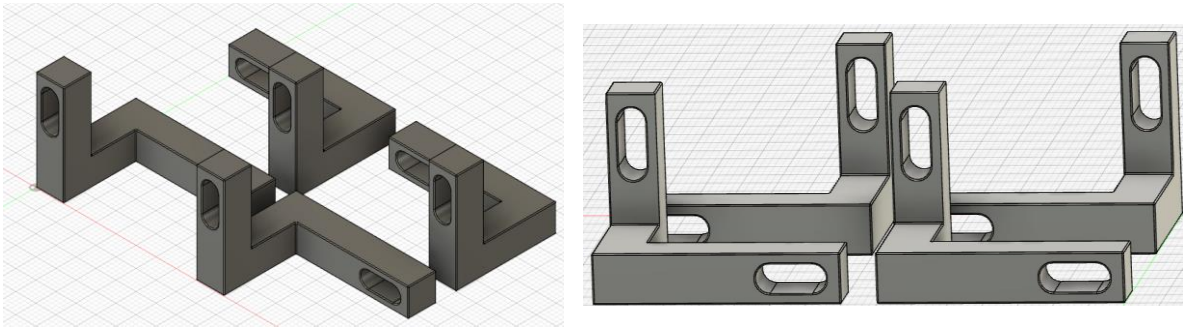


Εικόνα 2.48: Το πρόχειρο σχέδιο με τις διαστάσεις της πλακέτας οδήγησης των βηματικών κινητήρων [201].

Στην συνέχεια, με βάση τις μετρούμενες διαστάσεις της πλακέτας ULN2003, σχεδιάστηκε απευθείας στο Autodesk Fusion η βάση για τις πλακέτες οδήγησης των βηματικών κινητήρων ώστε να εξοικονομηθεί χρόνος. Επίσης, η βάση αυτή περιλαμβάνει ξεχωριστές θέσης υποδοχής για κάθε μία πλακέτα οδήγησης ULN2003 (Εικόνα Εικόνα 2.49). Με αυτόν τον τρόπο μειώνεται κατά πολύ το βάρος της βάσης και παραμένει στιβαρή. Ωστόσο, πρέπει να βρεθεί ένας τρόπος με τον οποίο θα συνδεθεί η βάση με τις πλακέτες ULN2003 με την βάση σταθεροποίησης των βηματικών κινητήρων. Γι' αυτό τον λόγο, σχεδιάστηκαν στο Autodesk Fusion δύο ζευγάρια συνδέσμων τα οποία είναι μικρά σε μέγεθος και έχουν ένα ικανοποιητικό πάχος ώστε να μπορέσουν να αντέξουν το βάρος της βάσης χωρίς να παραμορφωθούν (Εικόνα Εικόνα 2.50). Σε αυτό το σημείο αξίζει να σημειωθεί ότι για την εύρεση των διαστάσεων των συνδέσμων χρησιμοποιήθηκε το παχύμετρο,

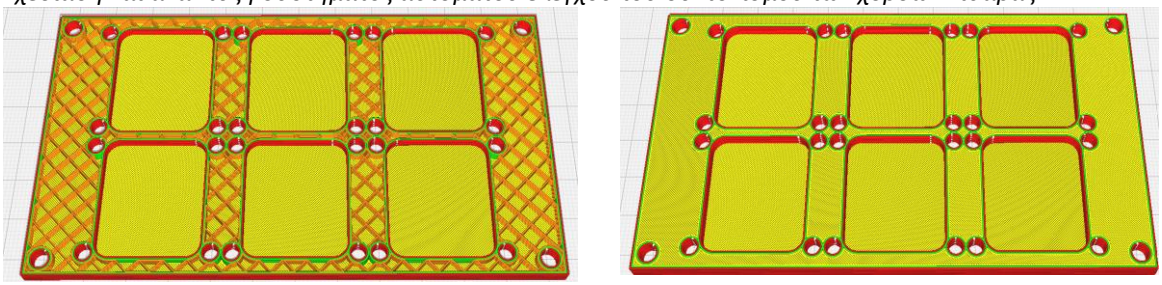


Εικόνα 2.49: Το τρισδιάστατο σχέδιο της βάσης για τις πλακέτες οδήγησης των βηματικών κινητήρων ULN2003 στο Autodesk Fusion [202].

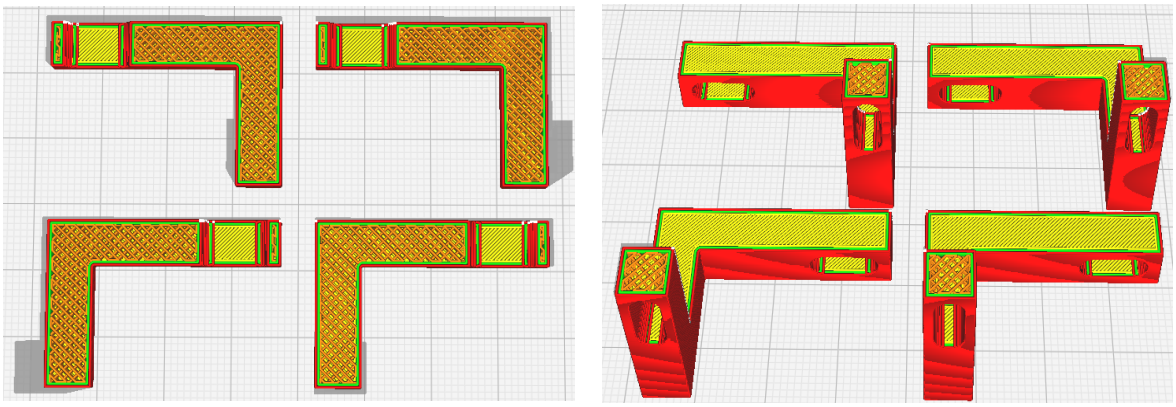


Εικόνα 2.50: Το τρισδιάστατο σχέδιο των συνδέσμων στήριξης της βάσης με τις πλακέτες οδήγησης των βηματικών κινητήρων ULN2003 στο Autodesk Fusion [203].

Έπειτα, με την ολοκλήρωση του τρισδιάστατου σχεδιασμού της βάσης και των συνδέσμων, έγινε εξαγωγή των δύο αυτών σχεδίων σε αρχεία υπό την μορφή .STL ώστε να εισαχθούν για περαιτέρω επεξεργασία στο λογισμικό τεμαχισμού UltiMaker Cura. Πιο συγκεκριμένα, η βάση για της πλακέτες ULN2003 επιλέχθηκε να έχει ένα κανονικό γέμισμα ώστε να μην προσθέτει βάρος στην κεφαλή του οργάνου εξαιτίας του μεγέθους της (Εικόνα Εικόνα 2.51). Από την άλλη, οι σύνδεσμοι επιλέχθηκαν να έχουν ένα αρκετά πυκνό γέμισμα μεταξύ των στρωμάτων ώστε να είναι στιβαροί και να μπορούν να αντέχουν το βάρος της βάσης με τις πλακέτες οδήγησης των βηματικών κινητήρων χωρίς να παραμορφώνονται (Εικόνα 2.52).

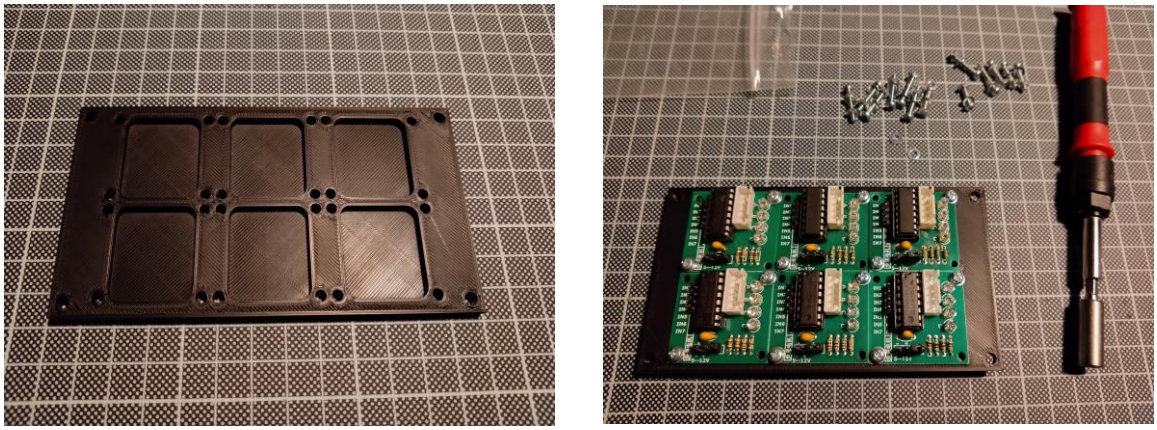


Εικόνα 2.51: Τα διάφορα στρώματα και ο τρόπος γεμίσματος μεταξύ των στρωμάτων της βάσης στήριξης των πλακετών οδήγησης των βηματικών κινητήρων ULN2003 στο λογισμικό τεμαχισμού UltiMaker Cura [204].

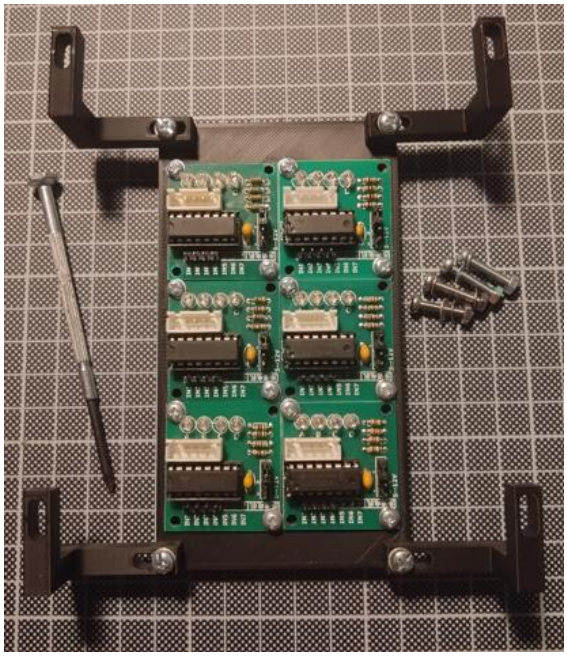


Εικόνα 2.52: Τα διάφορα στρώματα και ο τρόπος γεμίσματος μεταξύ των στρωμάτων των συνδέσμων στήριξης της βάσης με τις πλακέτες οδήγησης ULN2003 στο λογισμικό τεμαχισμού UltiMaker Cura [205].

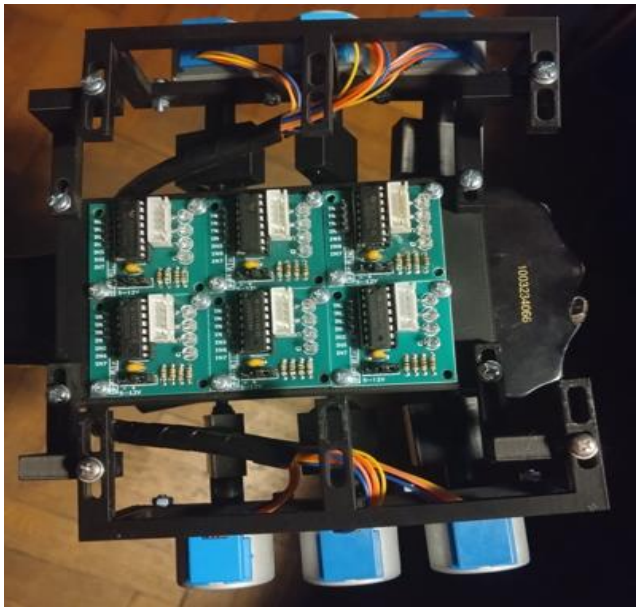
Ύστερα, αφού ολοκληρώθηκε η επεξεργασία των τρισδιάστατων σχεδίων της βάσης στήριξης των πλακετών οδήγησης ULN2003 και των συνδέσμων στο λογισμικό τεμαχισμού UltiMaker Cura, έγινε εξαγωγή των σχεδίων σε αρχεία υπό την μορφή .GCODE. Μετά, τα αρχεία αυτά αποθηκεύτηκαν στην κάρτα SD του εκτυπωτή ώστε να πραγματοποιηθούν οι τρισδιάστατες εκτυπώσεις. Πρώτα, πραγματοποιήθηκε η εκτύπωση της βάσης στήριξης των πλακετών οδήγησης ULN2003 (Εικόνα 2.53) και μετά οι σύνδεσμοι στήριξης της βάσης (Εικόνα 2.54). Ακόμα, όσον αφορά τις τρισδιάστατες εκτυπώσεις, πραγματοποιήθηκαν με την χρήση βιοδιασπώμενου πλαστικού και πιο συγκεκριμένα μαύρου PLA σε μορφή νήματος. Ωστόσο, τόσο οι πλακέτες οδήγησης ULN2003 όσο και οι σύνδεσμοι σταθεροποιήθηκαν με την χρήση βιδών, ροδελών και παξιμαδιών τύπου M3. Τέλος, έγινε τοποθέτηση της βάσης στήριξης των οδηγών ULN2003 στην βάση σταθεροποίησης των βηματικών κινητήρων μέσω της χρήσης των συνδέσμων (Εικόνα 2.55).



Εικόνα 2.53: Η βάση στήριξης των πλακετών οδήγησης των βηματικών κινητήρων ULN2003 [206].



Εικόνα 2.54: Η βάση στήριξης των πλακετών ULN2003 μαζί με τους συνδέσμους [207].

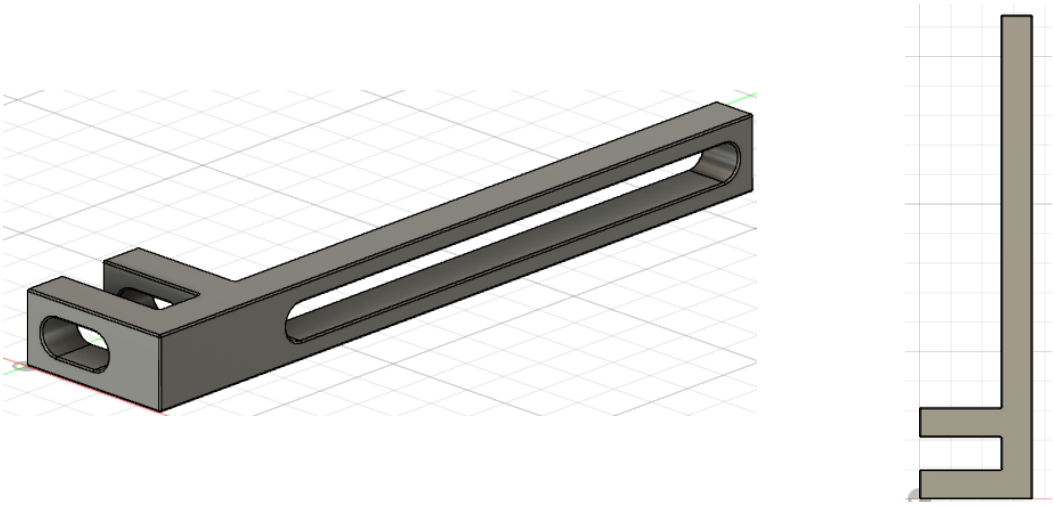


Εικόνα 2.55: Η βάση στήριξης των πλακετών οδήγησης των βηματικών κινητήρων ULN2003 στην κεφαλή της κιθάρας [208].

2.3.2.4 Οι αντάπτορες στήριξης των υπολοίπων βάσεων επάνω στην κεφαλή της κιθάρας

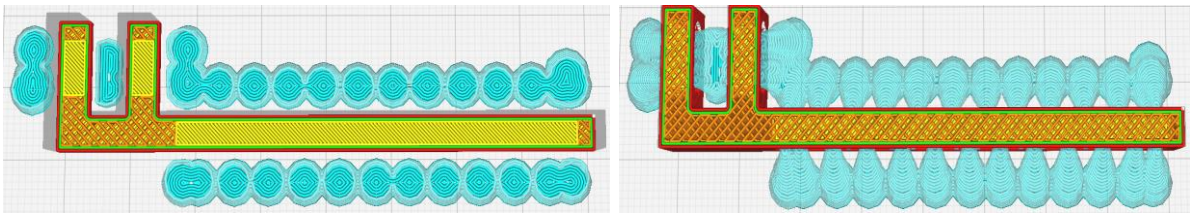
Στο σημείο αυτό, έπρεπε να βρεθεί ένας τρόπος με τον οποίο θα στηριχτούν όλες οι επόμενες βάσεις επάνω στην κεφαλή της κιθάρας. Αρχικά, το μόνο σημείο στο οποίο μπορούν να στηριχτούν οι επόμενες βάσης είναι επάνω στην βάση σταθεροποίησης των βηματικών κινητήρων. Όμως, επειδή οι επόμενες βάσεις θα πρέπει να στηριχτούν η μία πάνω από την άλλη με συγκεκριμένη σειρά, θα πρέπει να δημιουργηθεί ένας αντάπτορας στήριξης. Γι' αυτό λοιπόν, σε αυτή την ενότητα, αναλύετε η λογική πίσω από την σχεδίαση του αντάπτορα στήριξης των υπολοίπων βάσεων.

Ξεκινώντας, θα πρέπει ο αντάπτορας στήριξης να εφαρμόζεται επάνω στην βάση σταθεροποίησης των βηματικών κινητήρων και να έχει υποδοχή για τους συνδέσμους των επόμενων βάσεων. Επίσης, ο ίδιος ο αντάπτορας στήριξης θα πρέπει να έχει σύνδεσμο για να μπορέσει να σταθεροποιηθεί επάνω στην βάση σταθεροποίησης των βηματικών κινητήρων καθώς και υποδοχή για βίδα, ροδέλα και παξιμάδι τύπου M3. Επομένως, σύμφωνα με τις διαστάσεις από την βάση σταθεροποίησης των βηματικών κινητήρων σχεδιάστηκε κατευθείαν στο Autodesk Fusion ο αντάπτορας για την στήριξη των υπολοίπων βάσεων επάνω στην κεφαλή της κιθάρας (Εικόνα 2.56).



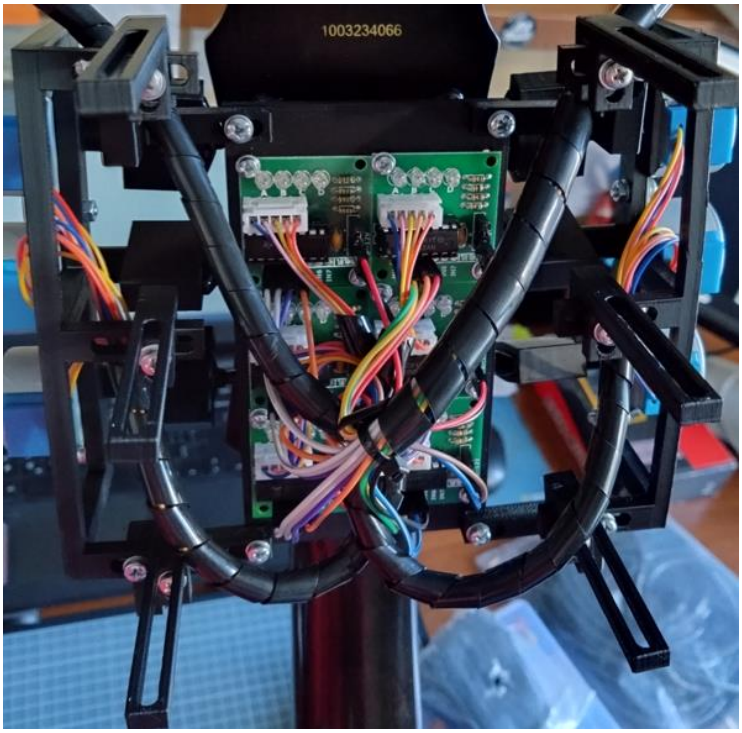
Εικόνα 2.56: Το τρισδιάστατο σχέδιο του αντάπτορα στήριξης των υπολοίπων βάσεων επάνω στην κεφαλή της κιθάρας στο Autodesk Fusion [209].

Έπειτα, αφού ολοκληρώθηκε ο τρισδιάστατος σχεδιασμός του αντάπτορα στήριξης για τις υπόλοιπες βάσεις έγινε εξαγωγή του σχεδίου σε αρχείο με μορφή .STL. Με αυτό το αρχείο, το τρισδιάστατο σχέδιο μπορεί να εισαχθεί στο λογισμικό τεμαχισμού UltimaMaker Cura και να επεξεργαστεί περαιτέρω. Πιο συγκεκριμένα, επειδή οι αντάπτορες αυτοί είναι σχετικά λεπτοί, θα πρέπει το γέμισμα μεταξύ των στρωμάτων να είναι αρκετά πυκνό ώστε να μπορούν να αντέχουν το βάρος των υπολοίπων βάσεων χωρίς να παραμορφώνονται. Επίσης, εξαιτίας της γεωμετρίας του αντάπτορα στήριξης των βάσεων, υπάρχουν σημεία τα οποία θα πέσουν κατά την διάρκεια της εκτύπωσης εξαιτίας της βαρύτητας. Γι αυτό λοιπόν θα πρέπει να δημιουργηθεί υποστήριξη των σημείων αυτών κατά την διάρκεια της εκτύπωσης. Επομένως, μέσω του λογισμικού τεμαχισμού UltiMaker Cura επιλέχθηκε η δενδροειδής υποστήριξη των σημείων αυτών διότι καταναλώνει λιγότερο υλικό και μπορεί να αφαιρεθεί πιο εύκολα χωρίς να προκληθεί φθορά στον αντάπτορα στήριξης (Εικόνα 2.57).



Εικόνα 2.57: Τα διάφορα στρώματα και ο τρόπος γεμίσματος μεταξύ των στρωμάτων για τους αντάπτορες στήριξης των υπολοίπων βάσεων στην κεφαλή της κιθάρας μέσω του λογισμικού τεμαχισμού UltiMaker Cura [210].

Στην συνέχεια, αφού ολοκληρώθηκε η επεξεργασία στο λογισμικό τεμαχισμού UltiMaker Cura έγινε εξαγωγή του σχεδίου σε ένα αρχείο της μορφής .GCODE και αποθηκεύτηκε στην κάρτα SD του εκτυπωτή. Με αυτό τον τρόπο μπορεί ο τρισδιάστατος εκτυπωτής να ξεκινήσει την τρισδιάστατη εκτύπωση των ανταπτόρων στήριξης των υπολοίπων βάσεων. Η τρισδιάστατη εκτύπωση των ανταπτόρων επιτεύχθηκε με την χρήση βιοδιασπώμενου πλαστικού και πιο συγκεκριμένα μαύρου PLA σε μορφή νήματος. Τέλος, αφού ολοκληρώθηκε η τρισδιάστατη εκτύπωση, οι αντάπτορες βιδώθηκαν επάνω στην βάση σταθεροποίησης των βηματικών κινητήρων με την χρήση βιδών ροδελών και παξιμαδιών τύπου M3 (Εικόνα 2.58).



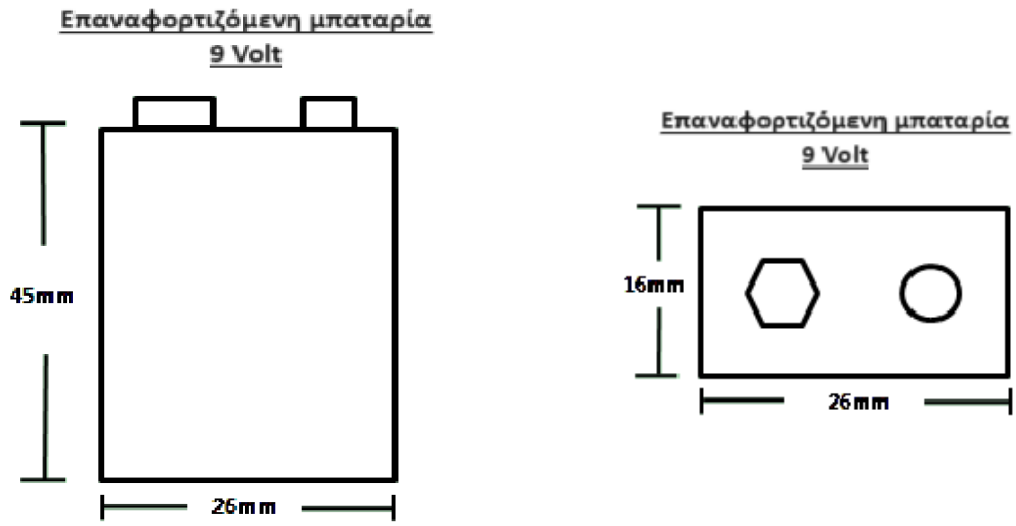
Εικόνα 2.58: Οι αντάπτορες στήριξης των υπολοίπων βάσεων επάνω στις βάσεις σταθεροποίησης των βηματικών κινητήρων [211].

2.3.2.5 Η βάση στήριξης των μπαταριών για την τροφοδοσία του συστήματος και οι σύνδεσμοι σταθεροποίησης επάνω στους αντάπτορες

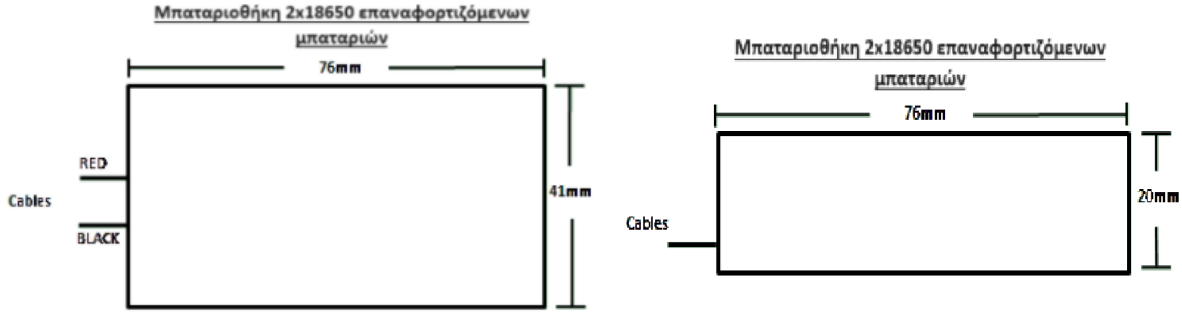
Σε αυτό το σημείο αξίζει να σημειωθεί ότι είναι αρκετά σημαντική η σειρά με την οποία θα τοποθετηθούν οι επόμενες βάσεις επάνω στην κεφαλή της κιθάρας. Πιο συγκεκριμένα, η βάση για την στήριξη των μπαταριών θα πρέπει να τοποθετηθεί όσο το δυνατόν πιο κοντά στην κεφαλή. Με αυτόν τον τρόπο το κέντρο βάρος του συστήματος θα βρίσκεται κοντά στην κεφαλή της κιθάρας και δεν θα επηρεάζει την αίσθηση του χρήστη κατά την χρήση του οργάνου. Επίσης, οι μπαταρίες θα πρέπει να βρίσκονται κοντά στα υπό-συστήματα που χρειάζονται τροφοδοσία, ώστε να ελαττωθούν τα μήκη των καλωδίων τροφοδοσίας. Συνεπώς, με βάση την λογική αυτή, πάνω στους αντάπτορες στήριξης θα τοποθετηθεί πρώτα η βάση στήριξης των μπαταριών με την χρήση των απαραίτητων συνδέσμων σταθεροποίησης.

Αρχικά, για την σχεδίαση της βάσης στήριξης των μπαταριών θα πρέπει πρώτα να μετρηθούν οι διαστάσεις των μπαταριών και της μπαταριοθήκης. Επομένως, με την χρήση του παχυμέτρου, μετρήθηκαν οι διαστάσεις των μπαταριών των 9 Volt και της

μπαταριοθήκης για την υποδοχή των δύο μπαταριών τύπου 18650. Έπειτα, σχεδιάστηκε ένα πρόχειρο σχέδιο στο Word με τις διαστάσεις των μπαταριών των 9Volt (Εικόνα 2.59) και της μπαταριοθήκης των μπαταριών 18650 (Εικόνα 2.60). Σύμφωνα λοιπόν με τις διαστάσεις που μετρήθηκαν αποφασίστηκε οι μπαταρίες και η μπαταριοθήκη να τοποθετηθούν στην ίδια βάση. Επίσης, η μπαταριοθήκη διαθέτει τρύπες για να βιδωθεί επάνω στην βάση γεγονός που πρέπει να ληφθεί υπ' όψιν κατά την σχεδίαση. Για τις επαναφορτιζόμενες μπαταρίες των 9Volt όμως θα πρέπει να βρεθεί ένας άλλος τρόπος στήριξης επάνω στην βάση διότι δεν έχουν υποδοχή για βίδα. Γι' αυτό λοιπόν, θα πρέπει επάνω στην βάση να δημιουργηθεί μια υποδοχή για κάθε μία μπαταρία των 9Volt, η οποία θα έχει μεγάλη ακρίβεια στις διαστάσεις ώστε η μπαταρία να κρατιέται σφικτά. Οι διαστάσεις σε αυτό το σημείο παίζουν σημαντικό ρόλο διότι δεν πρέπει κατά την τοποθέτηση της μπαταρίας να σπάσει η υποδοχή από το ζόρισμα. Συνεπώς, μετά από αρκετή σκέψη, σχεδιάστηκε στο Autodesk Fusion η βάση για την στήριξη των μπαταριών των 9Volt και της μπαταριοθήκης σύμφωνα πάντα με τις μετρούμενες διαστάσεις (Εικόνα 2.61). Στο τρισδιάστατο σχέδιο της βάσης, έχει δημιουργηθεί βάθρο υποδοχής για δύο μπαταρίες των 9Volt και της μπαταριοθήκης των 2x18650 μπαταριών. Με αυτόν τον τρόπο δημιουργείται καλύτερη στήριξη των μπαταριών, οικονομία του υλικού και του χρόνου κατά την εκτύπωση και μείωση του βάρους της βάσης.



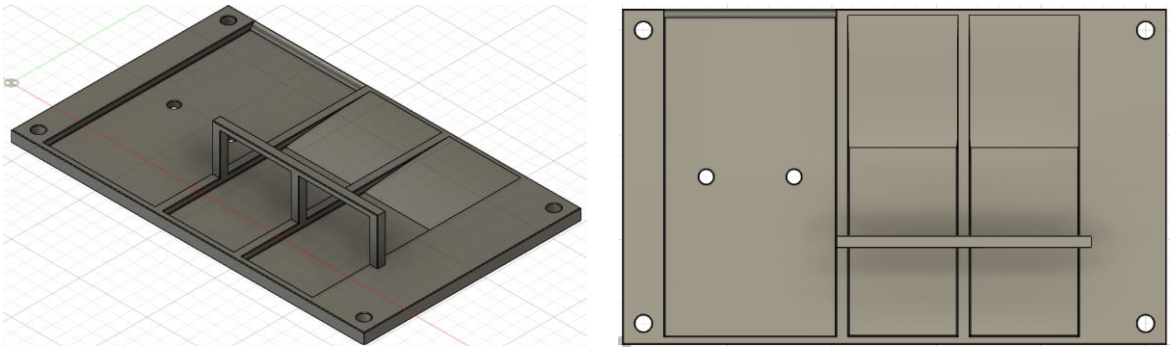
Εικόνα 2.59: Το πρόχειρο σχέδιο με τις διστάσεις των μπαταριών των 9 Volt [212].



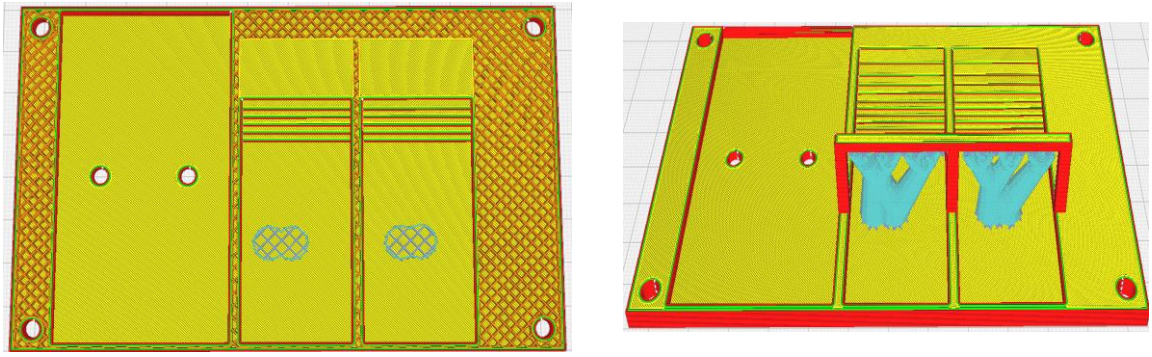
Εικόνα 2.60: Το πρόχειρο σχέδιο με τις διαστάσεις της μπαταριοθήκης για την υποδοχή των μπαταριών τύπου 18650 [213].

Στην συνέχεια, αφού ολοκληρώθηκε η τρισδιάστατη σχεδίαση της βάσης των μπαταριών στο Autodesk Fusion, έγινε εξαγωγή του σχεδίου σε ένα αρχείο της μορφής .STL. Με αυτόν τον τρόπο, το σχέδιο μπορεί να εισαχθεί στο λογισμικό τεμαχισμού

UltiMaker Cura για περαιτέρω επεξεργασία. Πιο συγκεκριμένα, στο λογισμικό τεμαχισμού δεν επιλέχτηκε πυκνό γέμισμα ενδιάμεσα στα στρώματα της συγκεκριμένης βάσης (Εικόνα 2.62). Με αυτόν τον τρόπο επιτυγχάνεται μείωση του βάρους, καθώς και μείωση του χρόνου και του υλικού εκτύπωσης της βάσης. Πάραυτα, το γέμισμα ενδιάμεσα στα στρώματα της βάσης είναι αρκετά πυκνό ώστε να μην παραμορφώνεται από το βάρος των μπαταριών. Επίσης, για την συγκεκριμένη βάση χρησιμοποιήθηκε η τεχνική υποστήριξης στις υποδοχές των δύο επαναφορτιζόμενων μπαταριών των 9Volt, ώστε να μην πέσει το κομμάτι αυτό κατά την διάρκεια της εκτύπωσης εξαιτίας της βαρύτητας. Ουσιαστικά, επιλέχθηκε το γέμισμα της μορφής δέντρου, διότι παρέχει καλύτερη στήριξη και μπορεί να αφαιρεθεί πιο εύκολα.



Εικόνα 2.61: Το τρισδιάστατο σχέδιο της βάσης για την στήριξη των μπαταριών στο Autodesk Fusion [214].

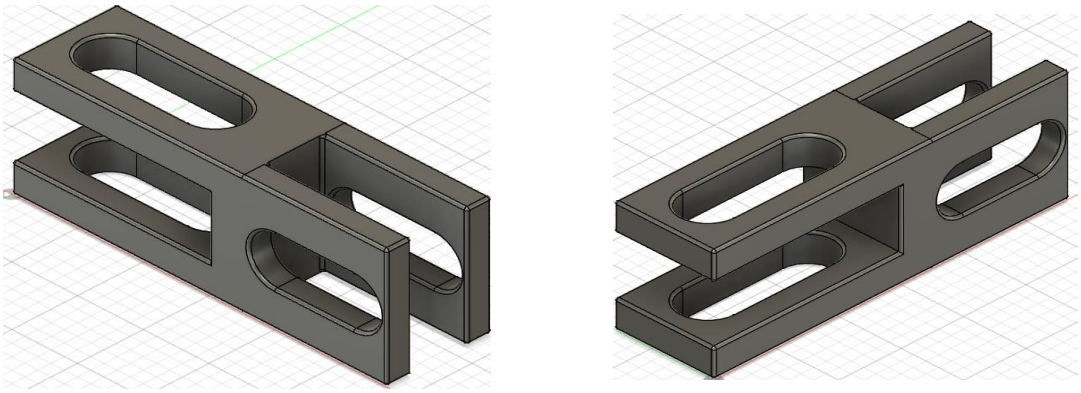


Εικόνα 2.62: Τα στρώματα, το γέμισμα ενδιάμεσα στα στρώματα και ο δεντροειδής τρόπος υποστήριξης των υποδοχών για την βάση στήριξης των μπαταριών στο λογισμικό τεμαχισμού UltiMaker Cura [215].

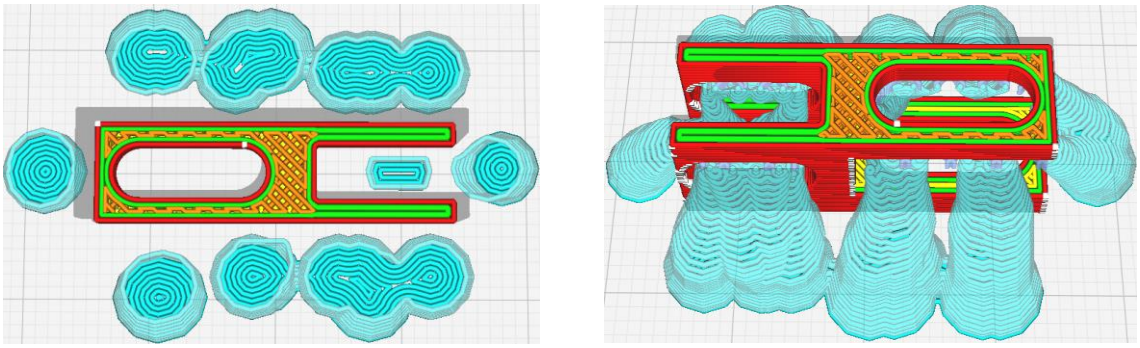
Έπειτα, έπρεπε να βρεθεί ένας τρόπος σχεδίασης των συνδέσμων σταθεροποίησης της βάσης επάνω στους αντάπτορες στήριξης. Με βάση λοιπόν τις διαστάσεις της βάσης στήριξης των μπαταριών και των ανταπτόρων σχεδιάστηκε στο Autodesk Fusion η τρισδιάστατη μορφή των συνδέσμων στήριξης της βάσης (Εικόνα 2.63). Όπως φαίνεται και στο τρισδιάστατο σχέδιο, οι σύνδεσμοι αυτοί είναι αρκετά λεπτή σαν κατασκευή. Με αυτόν τον τρόπο επιτυγχάνεται η ελάττωση του βάρους του συνδέσμου μιας και πρέπει να δημιουργηθούν τέσσερα κομμάτια για την σταθεροποίησης της βάσης επάνω στον αντάπτορα. Μετά την ολοκλήρωση της τρισδιάστατης σχεδίασης του συνδέσμου σταθεροποίησης της βάσης στήριξης των μπαταριών, έγινε εξαγωγή του σχεδίου σε ένα αρχείο της μορφής .STL. Με αυτόν τον τρόπο, το σχέδιο μπορεί να εισαχθεί στο λογισμικό τεμαχισμού UltiMaker Cura για περεταίρω επεξεργασία.

Στην συνέχεια, είναι ιδιαίτερα σημαντική η επεξεργασία του συνδέσμου σταθεροποίησης της βάσης στήριξης των μπαταριών στο λογισμικό τεμαχισμού UltiMaker

Cura. Πιο συγκεκριμένα, εξαιτίας της λεπτής κατασκευής του συνδέσμου θα πρέπει το γέμισμα μεταξύ των στρωμάτων να είναι αρκετά πυκνό. Με αυτόν τον τρόπο, θα αποφευχθεί η οποιαδήποτε παραμόρφωση του συνδέσμου από το βάρος της βάσης και των μπαταριών. Επίσης, θα πρέπει να προστεθεί υποστήριξη σε κάποια σημεία του συνδέσμου κατά την διάρκεια της εκτύπωσης ώστε να μην πέσουν εξαιτίας της βαρύτητας (Εικόνα 2.64). Ωστόσο, στα σημεία αυτά χρησιμοποιήθηκε η δενδροειδής υποστήριξη, διότι μπορεί να αφαιρεθεί πιο εύκολα και δεν καταναλώνει ιδιαίτερη ποσότητα υλικού.



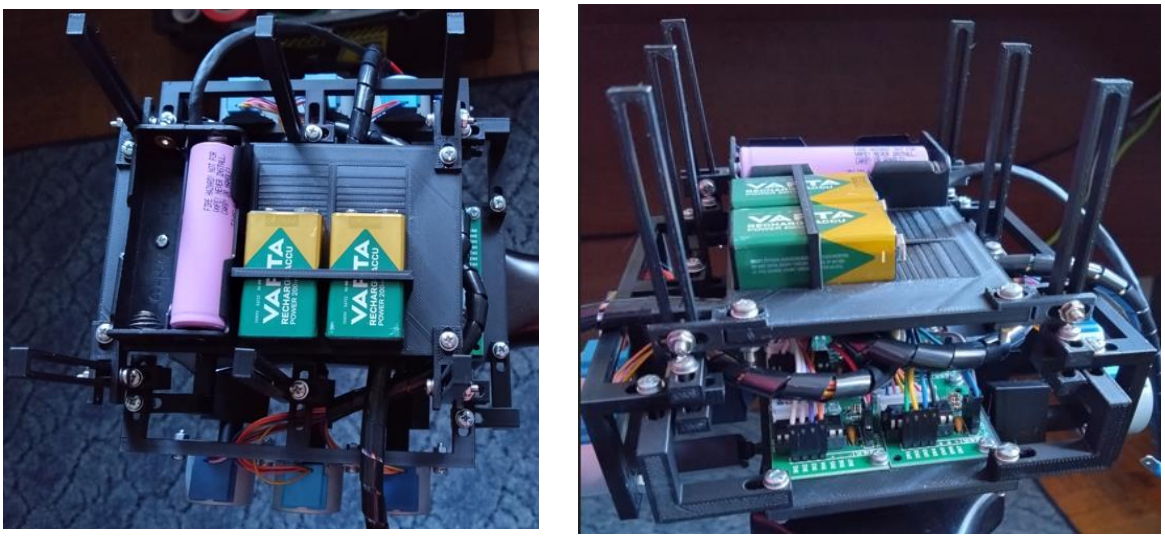
Εικόνα 2.63: Το τρισδιάστατο σχέδιο των συνδέσμων σταθεροποίησης της βάσης στήριξης των μπαταριών στο Autodesk Fusion [216].



Εικόνα 2.64: Τα στρώματα, ο τρόπος γεμίσματος μεταξύ των στρωμάτων και η υποστήριξη των ευάλωτων από την βαρύτητα σημείων του συνδέσμου σταθεροποίησης της βάσης στήριξης των μπαταριών στο λογισμικό τεμαχισμού UltiMaker Cura [217].

Ύστερα, με την ολοκλήρωση της επεξεργασίας της βάσης στήριξης των μπαταριών και των συνδέσμων σταθεροποίησης στο λογισμικό τεμαχισμού UltiMaker Cura, έγινε εξαγωγή των σχεδίων σε αρχεία με την μορφή .GCODE και αποθηκεύτηκαν στην κάρτα SD. Με αυτό τον τρόπο ο τρισδιάστατος εκτυπωτής μπορεί να ξεκινήσει την εκτύπωση της βάσης στήριξης των μπαταριών και των συνδέσμων σταθεροποίησης (Εικόνα 2.65). Για την εκτύπωση της βάσης και των συνδέσμων χρησιμοποιήθηκε βιοδιασπώμενο πλαστικό μαύρου χρώματος υπό την μορφή νήματος και πιο συγκεκριμένα PLA. Τέλος, με την ολοκλήρωση της εκτύπωσης οι σύνδεσμοι βιδώθηκαν επάνω στην βάση και εν συνεχεία επάνω στους αντάπτορες στήριξης με την χρήση βιδών, ροδελών και παξιμαδιών τύπου M3 (Εικόνα 2.66).

Εικόνα 2.65: Η βάση στήριξης των μπαταριών και οι σύνδεσμοι σταθεροποίησης της επάνω στους αντάπτορες [218].



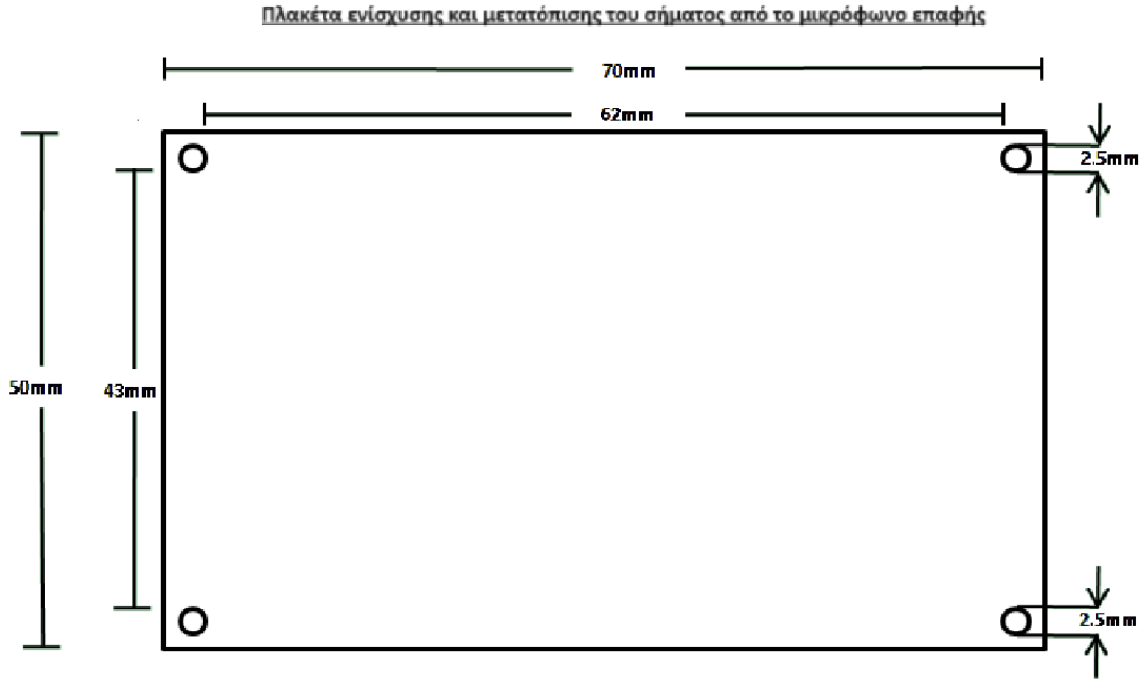
Εικόνα 2.66: Η βάση στήριξης των μπαταριών βιδωμένη επάνω στους αντάπτορες στήριξης στην κεφαλή της κιθάρας [219].

2.3.2.6 Η βάση στήριξης της πλακέτας ενίσχυσης και μετατόπισης του σήματος από το μικρόφωνο επαφής και οι σύνδεσμοι σταθεροποίησης της βάσης

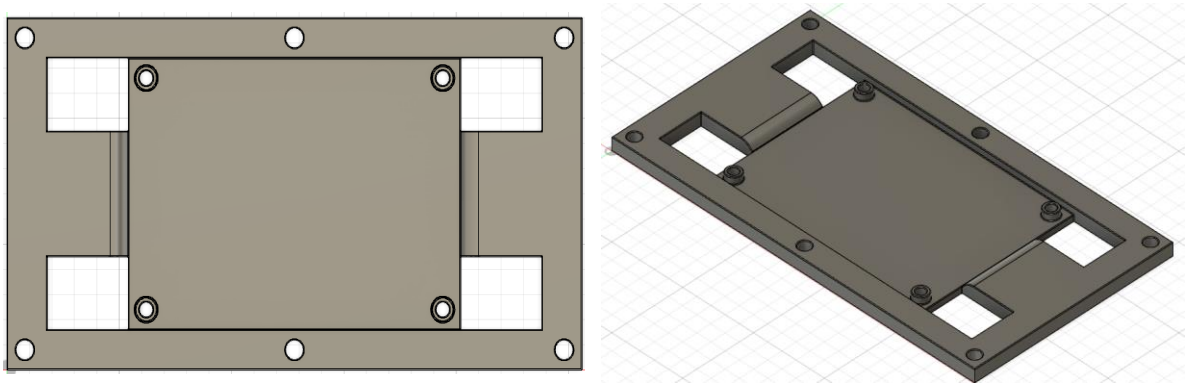
Η επόμενη βάση που θα πρέπει να τοποθετηθεί επάνω στους αντάπτορες περιέχει την πλακέτα ενίσχυσης και μετατόπισης του σήματος από το μικρόφωνο επαφής. Ουσιαστικά, η βάση αυτή επιλέχθηκε να τοποθετηθεί σε αυτό το σημείο για τον λόγο ότι η πλακέτα θα πρέπει να βρίσκεται κοντά στις μπαταρίες από τις οποίες τροφοδοτείτε και κοντά στο Arduino Mega2560 στο οποίο καταλήγει το ενισχυμένο και μετατοπισμένο σήμα. Σε αυτή λοιπόν την ενότητα, αναλύεται η λογική πίσω από τον τρόπο σχεδίασης της βάσης, πάνω στην οποία θα στηριχτεί η πλακέτα ενίσχυσης και μετατόπισης του σήματος από το μικρόφωνο επαφής, καθώς και των συνδέσμων με τους οποίους θα σταθεροποιηθεί επάνω στους αντάπτορες στήριξης.

Αρχικά, για την ανάπτυξη του τρισδιάστατου σχεδίου της βάσης στήριξης της πλακέτα ενίσχυσης και μετατόπισης του σήματος, θα πρέπει πρώτα να μετρηθούν οι διαστάσεις της με το παχύμετρο. Με βάση λοιπόν τις μετρούμενες διαστάσεις, δημιουργήθηκε στο Word ένα πρόχειρο σχέδιο της πλακέτας (Εικόνα 2.67). Λαμβάνοντας υπ' όψιν τις μετρούμενες διαστάσεις και το πρόχειρο σχέδιο, δημιουργήθηκε το

τρισδιάστατο σχέδιο της βάσης στήριξης της πλακέτας ενίσχυσης και μετατόπισης της πλακέτας στο Autodesk Fusion (Εικόνα 2.68). Κατά την σχεδίαση της συγκεκριμένης βάσης δημιουργήθηκε μια υποδοχή για την πλακέτα ώστε να σταθεροποιείτε καλύτερα καθώς και για την μείωση του βάρους της βάσης. Επίσης, επειδή το βάρος της πλακέτας είναι ελάχιστο, η βάση δεν κινδυνεύει να παραμορφωθεί. Γι' αυτό τον λόγο καθώς και για περαιτέρω μείωση του περιττού βάρους της βάσης, δημιουργήθηκαν κενά σε κάποια σημεία της βάσης. Επιπροσθέτως, επιτυγχάνεται εξοικονόμηση του χρόνου της τρισδιάστατης εκτύπωσης καθώς και του υλικού που θα χρησιμοποιηθεί. Με λίγα λόγια, η βάση αυτή προσφέρει στοιβαρότητα, αισθητική και εξοικονόμηση χρόνου και κόστους κατασκευής.



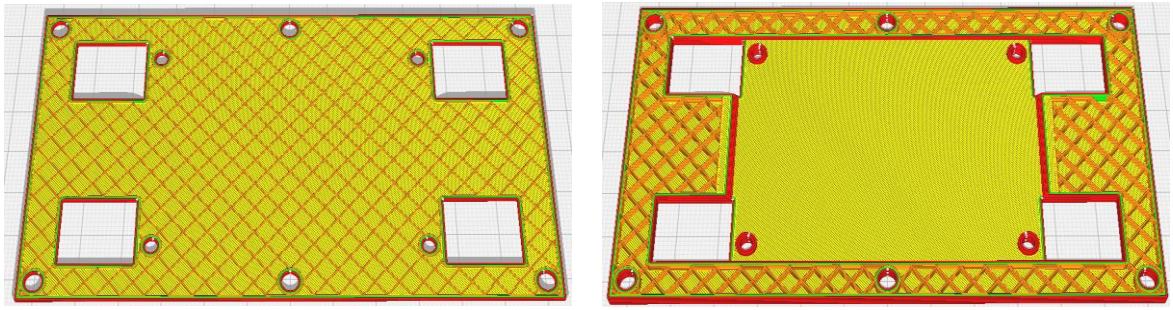
Εικόνα 2.67: Το πρόχειρο σχέδιο της πλακέτας με τις μετρούμενες διαστάσεις [220].



Εικόνα 2.68: Το τρισδιάστατο σχέδιο της βάσης για την στήριξη της πλακέτας ενίσχυσης και μετατόπισης του σήματος από το μικρόφωνο επαφής στο Autodesk Fusion [221].

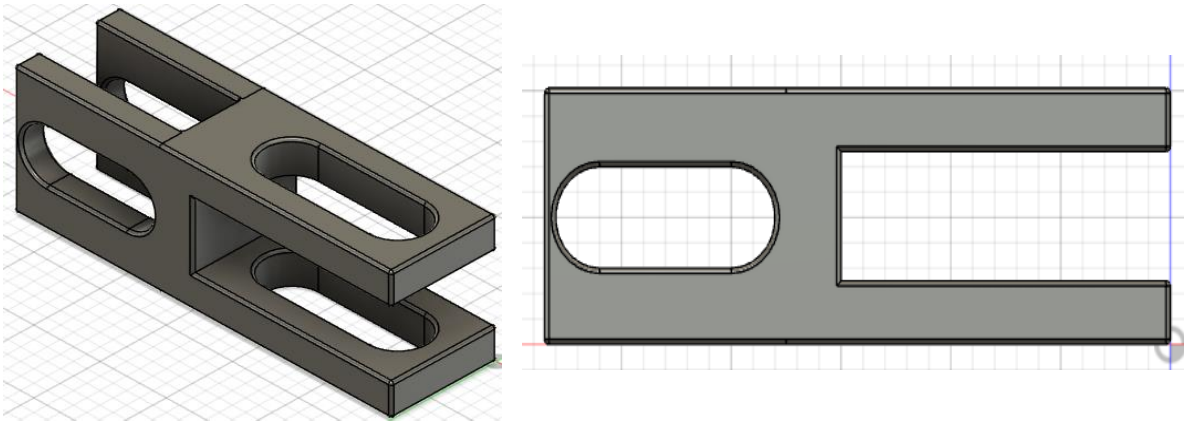
Εν συνεχεία, μετά την ολοκλήρωση της σχεδίασης της τρισδιάστατης βάσης για την στήριξη της πλακέτας ενίσχυσης και μετατόπισης του σήματος από το μικρόφωνο επαφής στο Autodesk Fusion, πραγματοποιήθηκε εξαγωγή του σχεδίου σε αρχείο με μορφή .STL. Με αυτόν τον τρόπο το σχέδιο της βάσης μπορεί να εισαχθεί στο λογισμικό τεμαχισμού

UltiMaker Cura για περαιτέρω επεξεργασία των στρωμάτων. Πιο συγκεκριμένα, η βάση αυτή δεν δέχεται κάποιο ιδιαίτερο βάρος, μιας και η πλακέτα είναι αρκετά ελαφριά και δεν υπάρχει κίνδυνος παραμόρφωσης. Επομένως, επιλέχθηκε ένα αραιό γέμισμα μεταξύ των στρωμάτων για την βάση στήριξης της πλακέτας ενίσχυσης και μετατόπισης του σήματος από το μικρόφωνο επαφής (Εικόνα 2.69). Με αυτόν τον τρόπο εξοικονομείται όχι μόνο υλικό και κατά συνέπεια το κόστος, αλλά και ο χρόνος εκτύπωσης.



Εικόνα 2.69: Τα στρώματα και ο τρόπος γεμίσματος μεταξύ των στρωμάτων της βάσης στήριξης της πλακέτας ενίσχυσης και μετατόπισης του σήματος από το μικρόφωνο επαφής στο λογισμικό τεμαχισμού UltiMaker Cura [222].

Πάραυτα, πρέπει να βρεθεί ένας τρόπος με τον οποίον θα στηριχτεί η βάση με την πλακέτα ενίσχυσης και μετατόπισης του σήματος από το μικρόφωνο επαφής επάνω στους αντάπτορες στήριξης. Αυτό μπορεί να επιτευχθεί με την χρήση συνδέσμων σταθεροποίησης όπως έγινε και στην προηγούμενη βάση. Συνεπώς, σύμφωνα με τις διαστάσεις της βάσης στήριξης της πλακέτας και των διαστάσεων από τους αντάπτορες στήριξης, σχεδιάστηκε στο Autodesk Fusion ένας αντίστοιχος σύνδεσμος σταθεροποίησης ο οποίος είναι παρόμοιος με τους συνδέσμους της προηγούμενης βάσης για την στήριξη των μπαταριών (Εικόνα 2.70). Ουσιαστικά, οι σύνδεσμοι σταθεροποίησης αυτοί έχουν σχεδιαστεί με τέτοιον τρόπο ώστε να θηλυκώνουν από την μία πλευρά στους αντάπτορες στήριξης και από την άλλη πλευρά στην βάση στήριξης της πλακέτας. Επίσης, έχουν δημιουργηθεί υποδοχές για βίδες τύπου M3 ώστε να σταθεροποιούν την βάση στους αντάπτορες στήριξης καλύτερα.

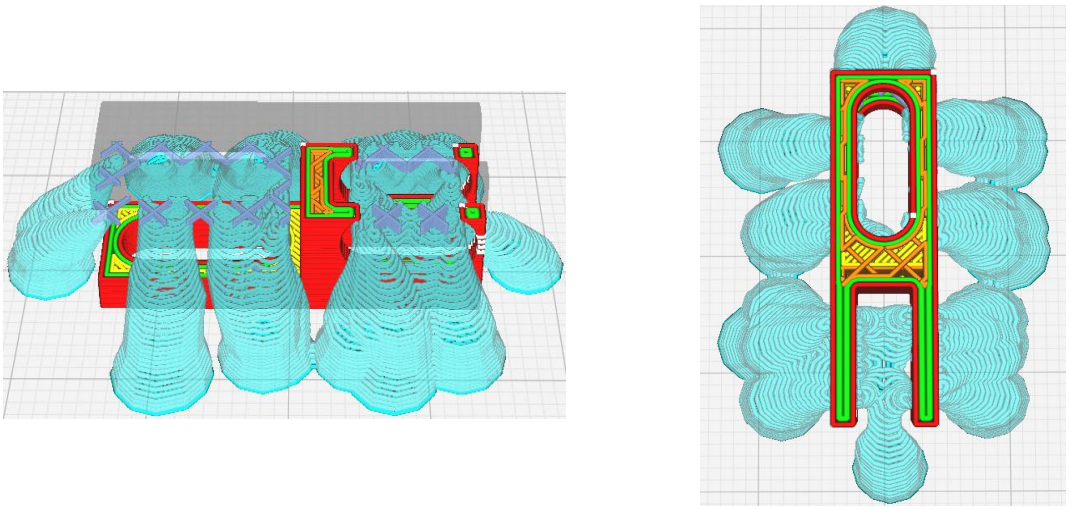


Εικόνα 2.70: Το τρισδιάστατο σχέδιο των συνδέσμων σταθεροποίησης της βάσης στήριξης της πλακέτας στο Autodesk Fusion [223].

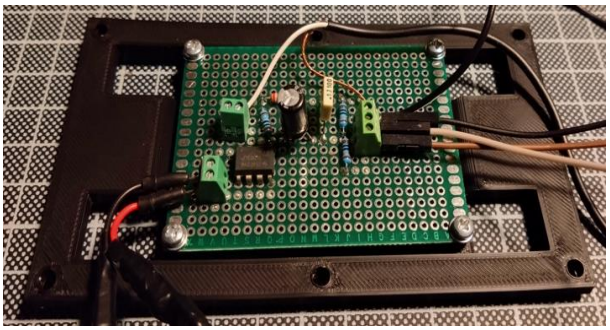
Μετά την ολοκλήρωση της τρισδιάστατης σχεδίασης του συνδέσμου σταθεροποίησης της βάσης επάνω στους αντάπτορες στήριξης, έγινε εξαγωγή του σχεδίου σε ένα αρχείο της μορφής .STL. Στην συνέχεια, το αρχείο αυτό εισήχθη στο λογισμικό τεμαχισμού UltiMaker Cura για περαιτέρω επεξεργασία. Πιο συγκεκριμένα, το συνολικό

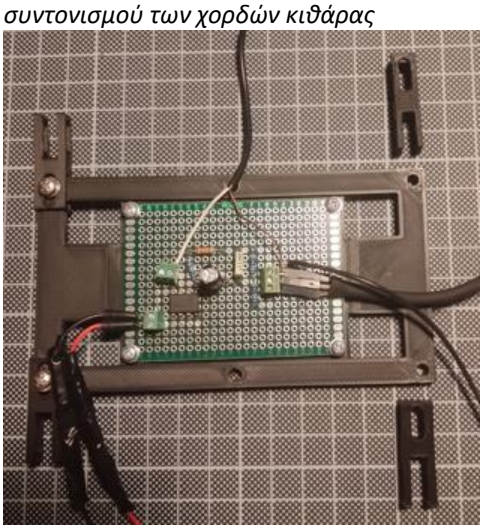
βάρος της βάσης μαζί με την πλακέτα είναι αρκετά μικρό, οπότε το γέμισμα ενδιάμεσα στα στρώματα δεν χρειάζεται να είναι πολύ πυκνό. Επίσης, εξαιτίας της γεωμετρίας του συνδέσμου σταθεροποίησης της βάσης, θα πρέπει να χρησιμοποιηθεί υποστήριξη σε κάποια σημεία. Πιο συγκεκριμένα, για τον σύνδεσμο σταθεροποίησης χρησιμοποιήθηκε υποστήριξη δενδροειδής μορφής, διότι καταναλώνεται λιγότερο υλικό και μπορεί να αφαιρεθεί πιο εύκολα χωρίς να γίνει κάποια ζημιά (Εικόνα 2.71).

Έπειτα, με την ολοκλήρωση της επεξεργασίας της βάσης στήριξης της πλακέτας και του συνδέσμου σταθεροποίησης στο λογισμικό τεμαχισμού UltiMaker Cura, έγινε εξαγωγή των τρισδιάστατων σχεδίων σε αρχεία με την μορφή .GCODE. Τα αρχεία αυτά, αποθηκεύτηκαν στην κάρτα SD ώστε να μπορέσει να ξεκινήσει την εκτύπωση της βάσης και του συνδέσμου σταθεροποίησης ο τρισδιάστατος εκτυπωτής (Εικόνα 2.72). Ωστόσο, για την εκτύπωση της βάσης αλλά και των συνδέσμων σταθεροποίησης, χρησιμοποιήθηκε βιοδιασπώμενο πλαστικό και πιο συγκεκριμένα PLA μαύρου χρώματος σε μορφή νήματος. Επίσης, θα πρέπει να εκτυπωθούν τέσσερα τεμάχια συνδέσμων για να μπορέσει να σταθεροποιηθεί η βάση επάνω στους αντάπτορες στήριξης. Τέλος, για την στήριξη της πλακέτας πάνω στην βάση και των συνδέσμων σταθεροποίησης τόσο στην βάση όσο και στους αντάπτορες στήριξης, χρησιμοποιήθηκαν βίδες, ροδέλες και παξιμάδια τύπου M3. Συνεπώς, με αυτόν τον τρόπο επιτεύχθηκε η τοποθέτηση της πλακέτας ενίσχυσης και μετατόπισης του σήματος από το μικρόφωνο επαφής στην κεφαλή της κιθάρας (Εικόνα 2.73).

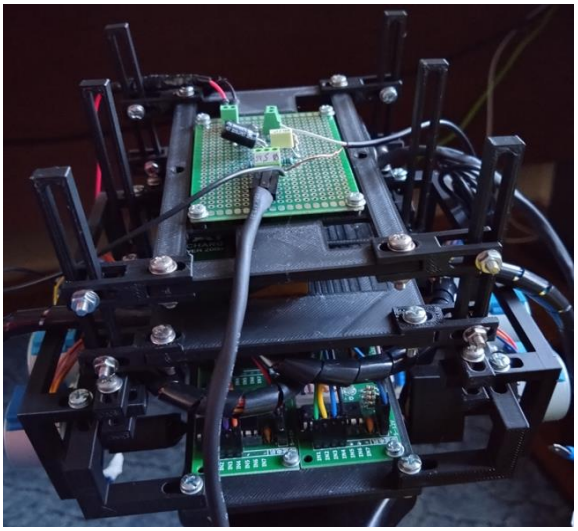


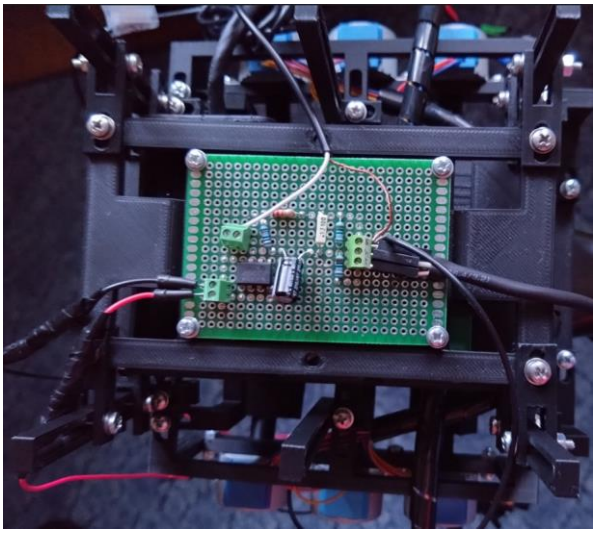
Εικόνα 2.71: Το γέμισμα ενδιάμεσα στα στρώματα και ο τύπος υποστήριξης των ευάλωτων από την βαρύτητα σημείων του συνδέσμου σταθεροποίησης για την βάση στήριξης της πλακέτας στο λογισμικό τεμαχισμού UltiMaker Cura [224].





Εικόνα 2.72: Η βάση στήριξης της πλακέτας ενίσχυσης και μετατόπισης του σήματος από το μικρόφωνο επαφής και οι σύνδεσμοι σταθεροποίησης της βάσης [225].





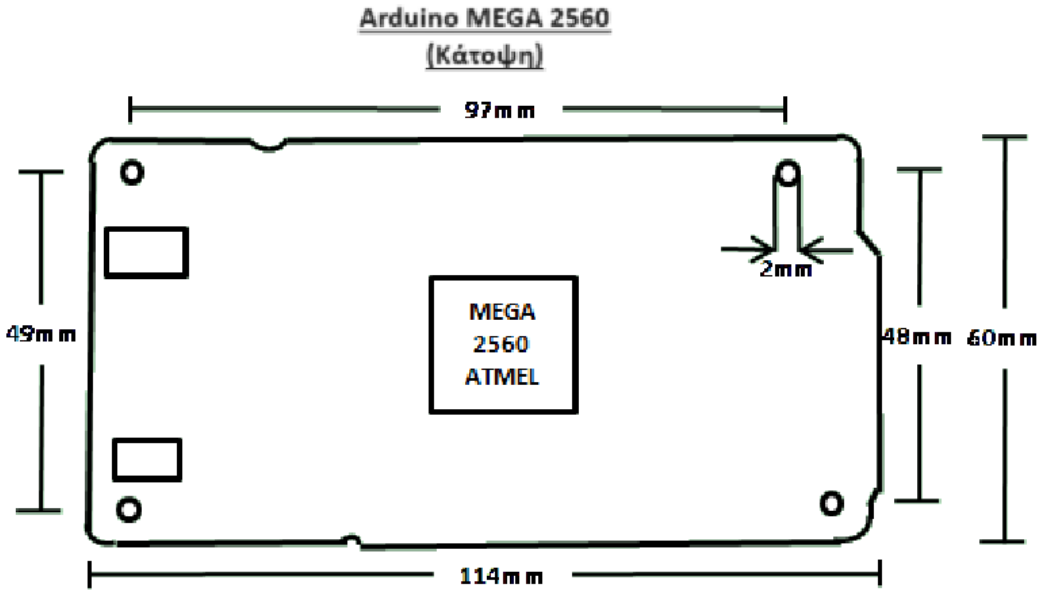
Εικόνα 2.73: Η βάση στήριξης της πλακέτας ενίσχυσης και μετατόπισης του σήματος από το μικρόφωνο επαφής μαζί με τους συνδέσμους τοποθετημένη στην κεφαλή της κιθάρας [226].

2.3.2.7 Η βάση στήριξης του Arduino Mega 2560 και οι σύνδεσμοι σταθεροποίησης της στους αντάπτορες στήριξης

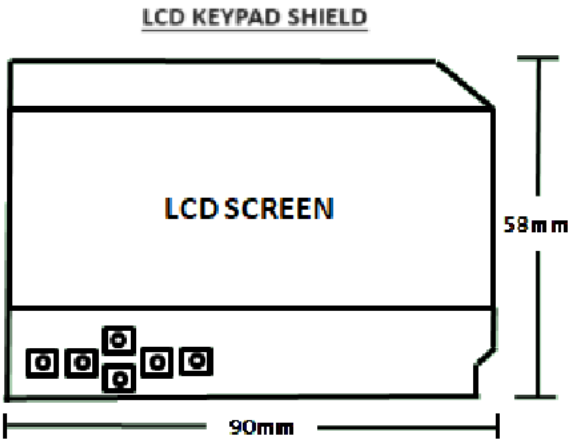
Ένα στάδιο πριν την ολοκλήρωση των εκτυπώσεων, βρίσκεται η σχεδίαση και η εκτύπωση της βάσης στήριξης του Arduino Mega 2560 και των συνδέσμων σταθεροποίησης της. Σε αυτήν την ενότητα, αναλύεται η λογική πίσω από την σχεδίαση της βάσης για την στήριξη του Arduino Mega 2560 επάνω στην κεφαλή της κιθάρας. Σε αυτό το σημείο πρέπει να δοθεί ιδιαίτερη προσοχή, διότι επάνω στο Arduino Mega 2560 τοποθετείται η πλακέτα της LCD οθόνη με το πληκτρολόγιό της η οποία αποτελεί την διεπαφή μεταξύ συστήματος και χρήστη.

Αρχικά, όπως και σε όλες τις προηγούμενες περιπτώσεις, θα πρέπει να μετρηθούν οι διαστάσεις τόσο του Arduino Mega 2560 όσο και της πλακέτας της LCD οθόνης με το πληκτρολόγιο. Με αυτόν τον τρόπο, μπορεί κανείς να σκεφτεί καλύτερα τον τρόπο σχεδίασης της βάσης καθώς και των σημείων στήριξης. Επομένως, σύμφωνα με τις διαστάσεις που μετρήθηκαν με την χρήση του παχυμέτρου, δημιουργήθηκαν στο Word δύο πρόχειρα σχέδια τόσο για το Arduino Mega 2560 (Εικόνα 2.74), όσο και για την

πλακέτα της LCD οθόνης με το πληκτρολόγιο (Εικόνα 2.75). Ένα ακόμα γεγονός που πρέπει να ληφθεί υπ' όψιν είναι ότι η πλακέτα της LCD οθόνης με το πληκτρολόγιο εφαρμόζει σφικτά επάνω στους ακροδέκτες του Arduino Mega 2560. Συνεπώς, η βάση που θα σχεδιαστεί θα πρέπει να στηρίζει το Arduino Mega 2560.

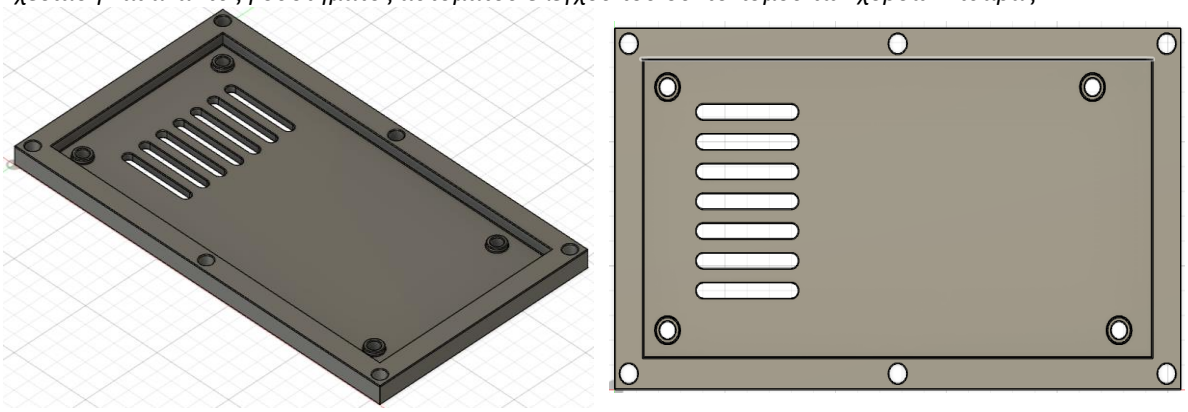


Εικόνα 2.74: Το πρόχειρο σχέδιο με τις διαστάσεις του Arduino Mega 2560 [227].



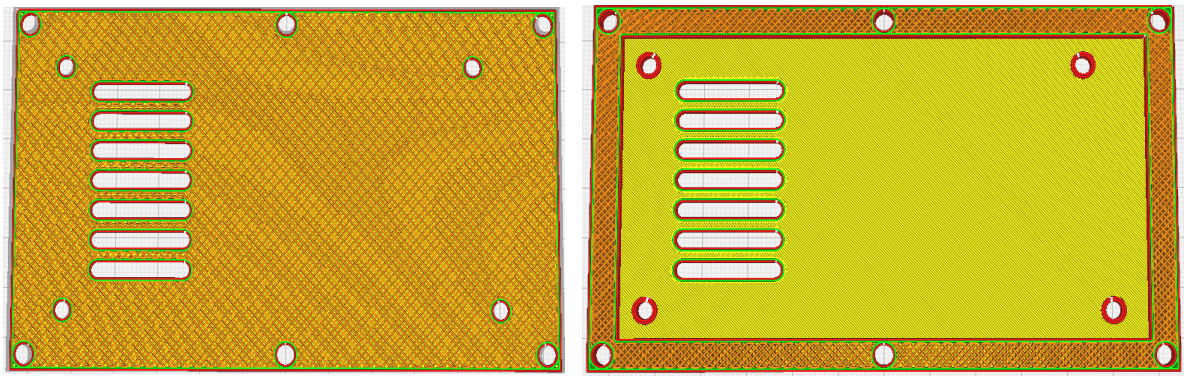
Εικόνα 2.75: Το πρόχειρο σχέδιο με τις διαστάσεις της πλακέτας της LCD οθόνης με το πληκτρολόγιο [228].

Σύμφωνα, λοιπόν με τα όσα αναφέρθηκαν μέχρι στιγμής και με βάση τις διαστάσεις από τα πρόχειρα σχέδια, πραγματοποιήθηκε η σχεδίαση της τρισδιάστατης μορφής της βάσης στήριξης του Arduino Mega 2560 στο Autodesk Fusion (Εικόνα 2.76). Πιο συγκεκριμένα, έχει δημιουργηθεί μία υποδοχή για το Arduino Mega 2560 καθώς και εγκοπές για την εισαγωγή αέρα στο οπίσθιο μέρος του. Επίσης, έχουν σχεδιαστεί τρύπες με τις απαραίτητες διαστάσεις ώστε να σταθεροποιηθεί το Arduino Mega 2560 επάνω στην βάση, αλλά και τρύπες για την προσθήκη των συνδέσμων. Ωστόσο, ο σχεδιασμός αυτός προσφέρει αισθητική και οικονομία τόσο σε χρόνο εκτύπωσης, όσο και σε κόστος. Αυτό επιτυγχάνεται, εξαιτίας της υποδοχής και των εγκοπών που έχουν δημιουργηθεί διότι δεν χρειάζεται να καλυφθεί με υλικό κατά την εκτύπωση μία μεγάλη επιφάνεια.



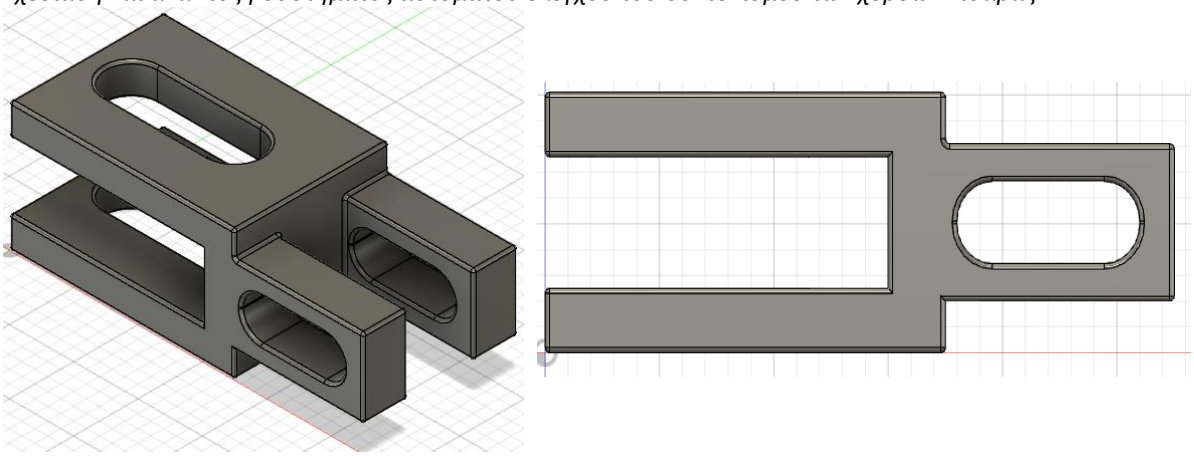
Εικόνα 2.76: Το τρισδιάστατο σχέδιο της βάσης στήριξης του Arduino Mega 2560 στο Autodesk Fusion [229].

Μετά την ολοκλήρωση της τρισδιάστατης σχεδίασης της βάσης για την στήριξη του Arduino, έγινε εξαγωγή του σχεδίου σε ένα αρχείο της μορφής .STL ώστε να εισαχθεί στο λογισμικό τεμαχισμού UltiMaker Cura για περαιτέρω επεξεργασία. Πιο συγκεκριμένα, επειδή η βάση αυτή θα πρέπει να στηρίξει το βάρος του Arduino Mega 2560 και της πλακέτας με την LCD οθόνη και το πληκτρολόγιο θα πρέπει να επιλεχθεί ένα αρκετά πυκνό γέμισμα (Εικόνα 2.77). Με αυτόν τον τρόπο, δεν θα υπάρχει κίνδυνος παραμόρφωσης της βάσης από το συνολικό βάρος του Arduino και της πλακέτας με την LCD οθόνη και το ενσωματωμένο πληκτρολόγιο.



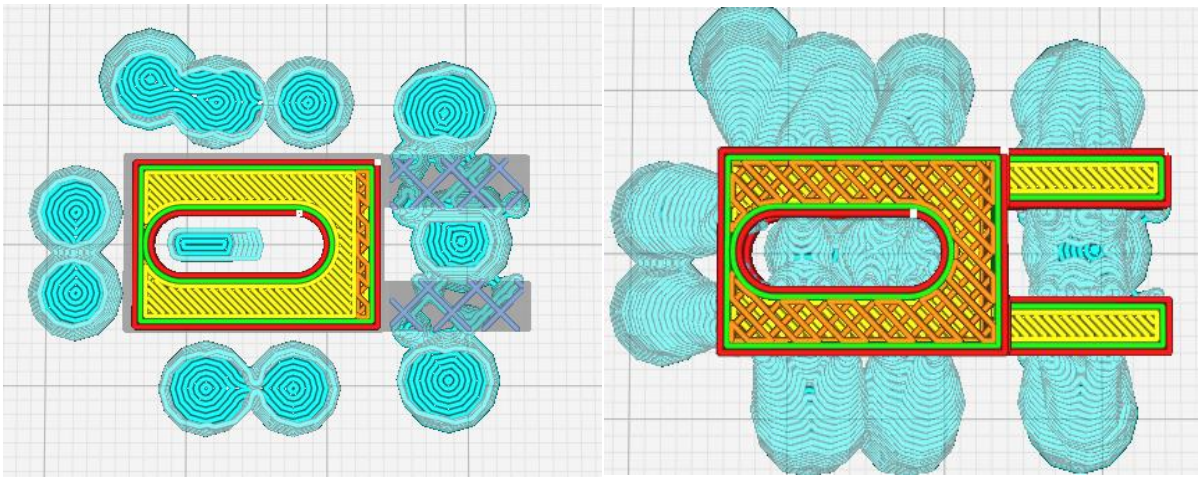
Εικόνα 2.77: Τα στρώματα και το γέμισμα ενδιάμεσα στα στρώματα της βάσης στήριξης του Arduino Mega 2560 στο λογισμικό τεμαχισμού UltiMaker Cura [230].

Ωστόσο, θα πρέπει να βρεθεί ένας τρόπος σταθεροποίησης της βάσης στήριξης του Arduino επάνω στους αντάπτορες στήριξης των βάσεων. Ουσιαστικά, θα πρέπει να σχεδιαστεί ένας αντίστοιχος σύνδεσμος σταθεροποίησης όπως έγινε για τις προηγούμενες βάσης. Συνεπώς, σύμφωνα με τις διαστάσεις της βάσης στήριξης του Arduino Mega 2560 και με τις διαστάσεις του αντάπτορα στήριξης, δημιουργήθηκε στο Autodesk Fusion το τρισδιάστατο σχέδιο του συνδέσμου σταθεροποίησης (Εικόνα 2.78). Ο σχεδιασμός του συνδέσμου σταθεροποίησης έγινε με τέτοιον τρόπο ώστε να δείχνει λεπτός για να μην προσθέτει ιδιαίτερο βάρος στους αντάπτορες στήριξης. Με αυτόν τον τρόπο εξοικονομείται χρόνος εκτύπωσης και κόστος μιας και χρησιμοποιείτε λιγότερο υλικό. Επίσης, στον σύνδεσμο έχουν δημιουργηθεί εγκοπές ώστε να μπορέσει να βιδωθεί επάνω στην βάση στήριξης του Arduino και εν συνεχεία επάνω στους αντάπτορες στήριξης των βάσεων.



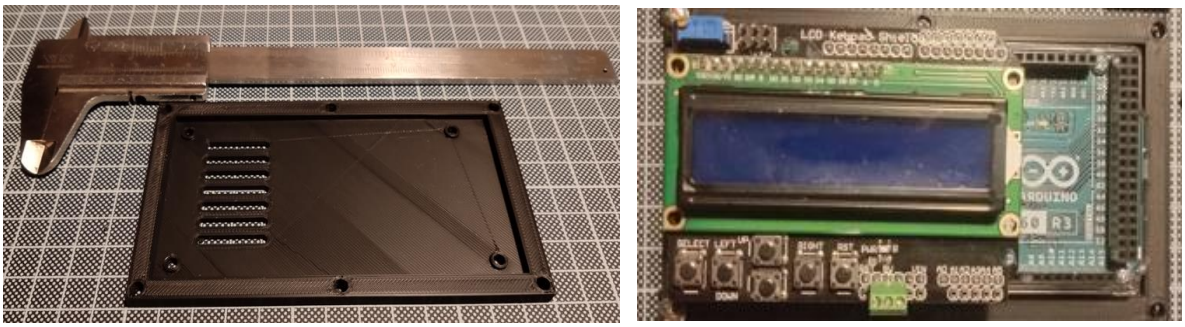
Εικόνα 2.78: Το τρισδιάστατο σχέδιο του συνδέσμου σταθεροποίησης της βάσης στήριξης του Arduino Mega 2560 στο Autodesk Fusion [231].

Έπειτα, αφού ολοκληρώθηκε η τρισδιάστατη σχεδίαση του συνδέσμου σταθεροποίησης της βάσης στο Autodesk Fusion, έγινε εξαγωγή του σχεδίου σε ένα αρχείο της μορφής .STL. Εν συνεχεία, το αρχείο αυτό εισήχθη στο λογισμικό τεμαχισμού UltiMaker Cura για περεταίρω επεξεργασία. Πιο συγκεκριμένα, επειδή ο σχεδιασμός του συνδέσμου σταθεροποίησης είναι σχετικά λεπτός, θα πρέπει το γέμισμα μεταξύ των στρωμάτων να είναι αρκετά πυκνό. Με αυτόν τον τρόπο, οι τέσσερις σύνδεσμοι σταθεροποίησης θα είναι αρκετά στιβαροί και θα μπορέσουν να αντέξουν το βάρος της βάσης στήριξης του Arduino Mega 2560 και της πλακέτας με την LCD οθόνη και το ενσωματωμένο πληκτρολόγιο χωρίς να παραμορφωθούν. Επίσης, εξαιτίας της περίπλοκης γεωμετρίας του συνδέσμου σταθεροποίησης θα πρέπει μέσω του λογισμικού τεμαχισμού UltiMaker Cura να δημιουργηθούν υποστηρίξεις σε μερικά σημεία κατά την διάρκεια της εκτύπωσης (Εικόνα 2.79). Έτσι, δεν υπάρχει κίνδυνος πτώσης και κατά συνέπεια καταστροφής του συνδέσμου σταθεροποίησης κατά την διάρκεια της εκτύπωσης εξαιτίας της βαρύτητας. Συνεπώς, για τον συγκεκριμένο σύνδεσμο σταθεροποίησης χρησιμοποιήθηκε ο δενδροειδής τύπος υποστήριξης ο οποίος δεν καταναλώνει αρκετό υλικό κατά την εκτύπωση και μπορεί να αφαιρεθεί εύκολα χωρίς να φθαρεί ο σύνδεσμος σταθεροποίησης.

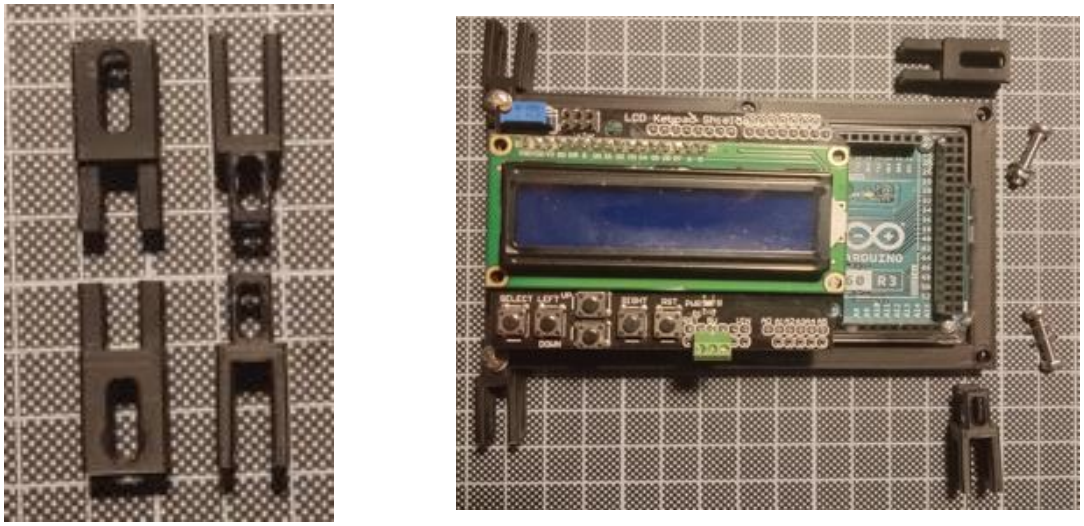


Εικόνα 2.79: Τα στρώματα, το γέμισμα μεταξύ των στρωμάτων και η δενδροειδής υποστήριξης κάποιων σημείων του συνδέσμου στο λογισμικό τεμαχισμού UltiMaker Cura [232].

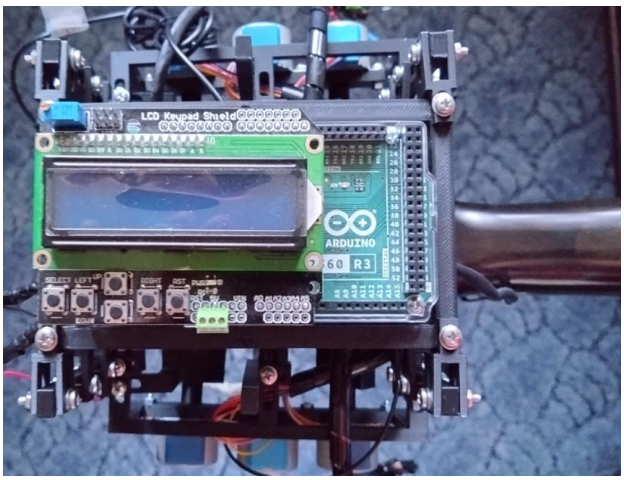
Εν’ τέλη, με την ολοκλήρωση της επεξεργασίας στο λογισμικό τεμαχισμού, UltiMaker Cura, τόσο της βάσης στήριξης του Arduino όσο και του συνδέσμου σταθεροποίησης της βάσης στους αντάπτορες στήριξης, έγινε εξαγωγή των δύο αυτών σχεδίων σε αρχεία με την μορφή .GCODE. Τα αρχεία αυτά με την σειρά τους αποθηκεύτηκαν στην κάρτα SD ώστε να εισαχθούν στον εκτυπωτή και αυτός με την σειρά του να ξεκινήσει τις τρισδιάστατες εκτυπώσεις. Πρώτα πραγματοποιήθηκε η τρισδιάστατη εκτύπωση της βάσης για την στήριξη του Arduino Mega 2560 (Εικόνα 2.80) και έπειτα η εκτύπωση των τεσσάρων συνδέσμων σταθεροποίησης της βάσης (Εικόνα 2.81). Όσον αφορά τις εκτυπώσεις, χρησιμοποιήθηκε βιοδιασπώμενο πλαστικό και πιο συγκεκριμένα μαύρο PLA σε μορφή νήματος. Τέλος, η στήριξη του Arduino Mega 2560 επάνω στην βάση επιτεύχθηκε με την χρήση βιδών, ροδελών και παξιμαδιών τύπου M2. Ωστόσο, η βάση σταθεροποιήθηκε με την σειρά της επάνω στους αντάπτορες στήριξης μέσω των συνδέσμων και αυτό επιτεύχθηκε με την χρήση βιδών, ροδελών και παξιμαδιών τύπου M3 (Εικόνα 2.82).

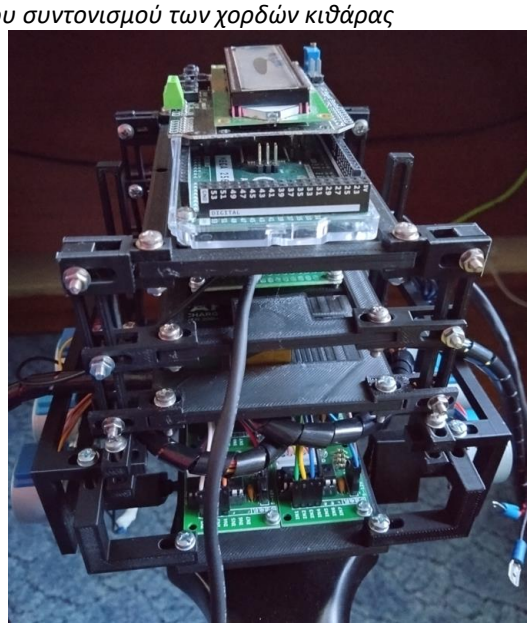


Εικόνα 2.80: Η βάση στήριξης του Arduino Mega 2560 μετά την ολοκλήρωση της τρισδιάστατης εκτύπωσης [233].



Εικόνα 2.81: Οι σύνδεσμοι για την σταθεροποίηση της βάσης στήριξης του Arduino επάνω στους αντάπτορες στήριξης των βάσεων [234].





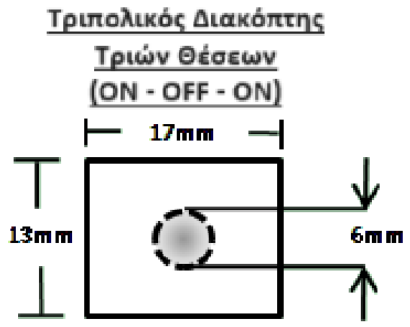
Εικόνα 2.82: Η βάση στήριξης του Arduino Mega 2560 σταθεροποιημένη επάνω στους αντάπτορες στήριξης των βάσεων [235].

2.3.2.8 Η βάση στήριξης του τριπολικού διακόπτη τροφοδοσίας του συστήματος και της μπαταρίας τροφοδοσίας του Arduino Mega 2560

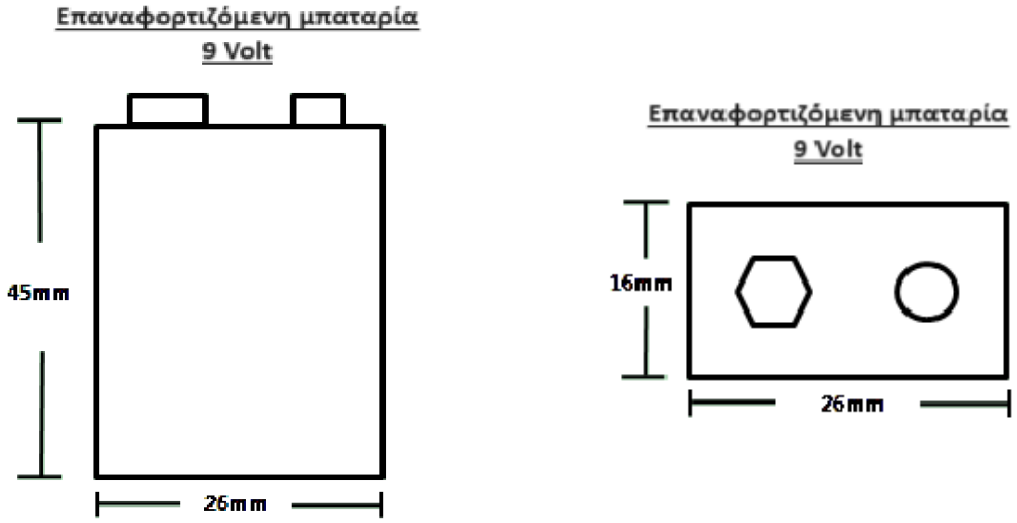
Η τελευταία βάση που σχεδιάστηκε και εκτυπώθηκε αποσκοπεί στην στήριξη του διακόπτη τροφοδοσίας του συστήματος και της μπαταρίας για την τροφοδοσία του Arduino Mega 2560. Πιο συγκεκριμένα, επειδή δεν υπήρχε χώρος επάνω στην βάση στήριξης των μπαταριών έπρεπε να βρεθεί ένας τρόπος να τοποθετηθεί και αυτή η επαναφορτιζόμενη μπαταρία των 9Volt επάνω στην κεφαλή της κιθάρας. Σε αυτή λοιπόν την ενότητα, αναλύεται η λογική πίσω από την σχεδίαση της βάσης στήριξης τόσο του διακόπτη τροφοδοσίας του συστήματος όσο και της μπαταρίας για την τροφοδοσία του Arduino Mega 2560.

Αρχικά, όπως πραγματοποιήθηκε και στις προηγούμενες βάσης, θα πρέπει να μετρηθούν οι διαστάσεις του τριπολικού διακόπτη με την χρήση του παχυμέτρου. Αφού μετρήθηκαν η διαστάσεις, δημιουργήθηκε στο Word ένα πρόχειρο σχέδιο του τριπολικού διακόπτη (Εικόνα 2.83). Ωστόσο, για την τρισδιάστατη σχεδίαση της βάσης θα χρειαστούν και οι διαστάσεις της μπαταρίας μιας και θα στηριχτεί και αυτή επάνω στην ίδια βάση. Όσον αφορά τις διαστάσεις της επαναφορτιζόμενης μπαταρίας έχουν αναφερθεί σε προηγούμενη ενότητα. Όμως, για λόγους πληρότητας εμφανίζεται και σε αυτήν την ενότητα το πρόχειρο σχέδιο στο Word, το οποίο περιλαμβάνει τις διαστάσεις της επαναφορτιζόμενης μπαταρίας (Εικόνα 2.84). Συνεπώς, θα πρέπει να σχεδιαστεί μία βάση η οποία θα στηρίζει και τον τριπολικό διακόπτη αλλά και την επαναφορτιζόμενη μπαταρία των 9Volt. Πάραυτα, θα πρέπει να βρεθεί το κατάλληλο σημείο στο οποίο θα τοποθετηθεί η συγκεκριμένη βάση επάνω στην κεφαλή της κιθάρας. Ουσιαστικά, το σημείο σταθεροποίησης της βάσης αποτελεί έναν εξίσου σημαντικό παράγοντα από τον οποίο εξαρτάται η μορφή που θα έχει η βάση. Έπειτα από αρκετή σκέψη, αποφασίστηκε η μία πλευρά της βάσης να στηριχτεί επάνω στον μεσαίο αντάπτορα στήριξης βάσεων και η άλλη πλευρά επάνω στην βάση στήριξης του Arduino Mega 2560. Με αυτόν τον τρόπο προσφέρεται σταθερότητα στην βάση στήριξης του τριπολικού διακόπτη και της

επαναφορτιζόμενης μπαταρίας των 9Volt, αλλά και στην βάση στήριξης του Arduino μιας και αποκτά άλλο ένα σημείο σταθεροποίησης.

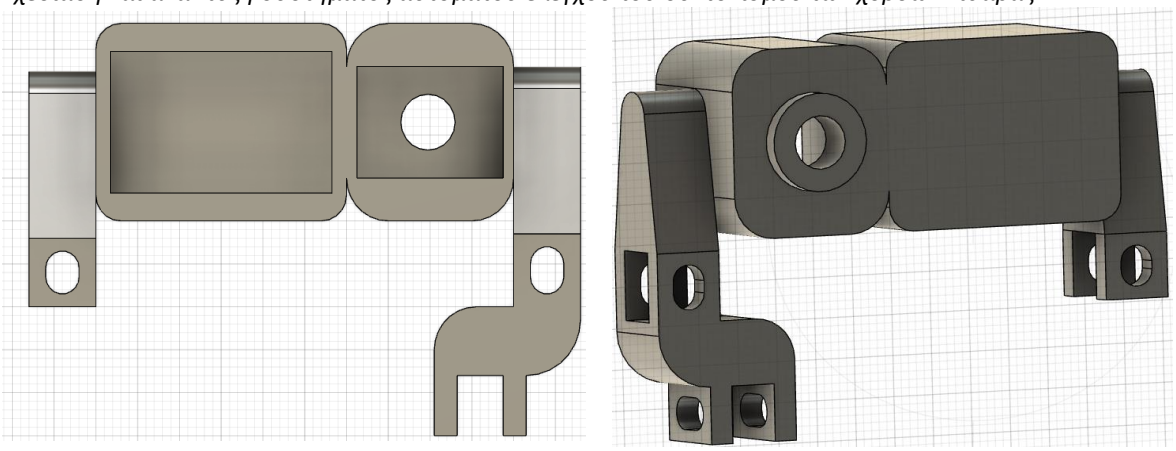


Εικόνα 2.83: Το πρόχειρο σχέδιο με τις διαστάσεις του τριπολικού διακόπτη [236].



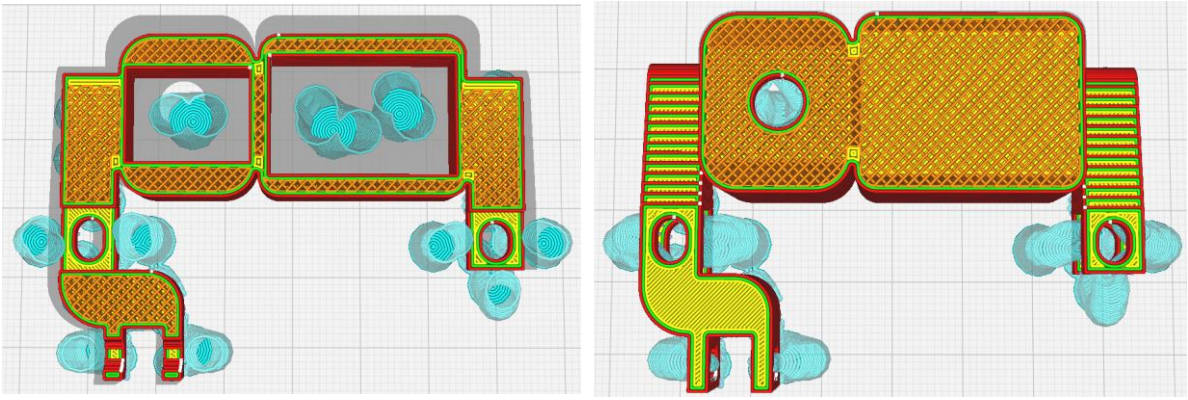
Εικόνα 2.84: Το πρόχειρο σχέδιο με τις διαστάσεις της επαναφορτιζόμενης μπαταρίας των 9Volt [237].

Εν συνεχεία, σύμφωνα με τις διαστάσεις του τριπολικού διακόπτη, της επαναφορτιζόμενης μπαταρίας, της βάσης στήριξης του Arduino και του αντάπτορα σταθεροποίησης των βάσεων, σχεδιάστηκε στο Autodesk Fusion η τρισδιάστατη μορφή της βάσης (Εικόνα 2.85). Όπως φαίνεται και στο τρισδιάστατο σχέδιο, η βάση αυτή έχει δύο υποδοχές, μία για τον τριπολικό διακόπτη και μία για την επαναφορτιζόμενη μπαταρία των 9 Volt. Οι υποδοχές αυτές έχουν δημιουργηθεί με τέτοια ακρίβεια ώστε τόσο ο τριπολικός διακόπτης όσο και η επαναφορτιζόμενη μπαταρία να σφηνώνονται επάνω στην βάση. Επίσης, έχει δημιουργηθεί μία τρύπα μέσα από την οποία διέρχεται ο λεβιές τριπολικού διακόπτη. Ωστόσο, η βάση του λεβιέ έχει βόλτες και μπορεί να σταθεροποιηθεί καλύτερα επάνω στην βάση με την χρήση του κυκλικού παξιμαδιού που περιέχει. Επιπροσθέτως, στα σημεία σταθεροποίησης της βάσης έχουν δημιουργηθεί τρύπες για την υποδοχή βιδών τύπου M3 για όταν τοποθετηθεί η βάση επάνω στην κεφαλή της κιθάρας. Ακόμα, στην συγκεκριμένη βάση έχει δημιουργηθεί μία υποδοχή εντός της οποίας θα θηλυκώσει ο αντάπτορας στήριξης των βάσεων προσφέροντας έτσι καλύτερη σταθεροποίηση της βάσης επάνω στην κεφαλή.



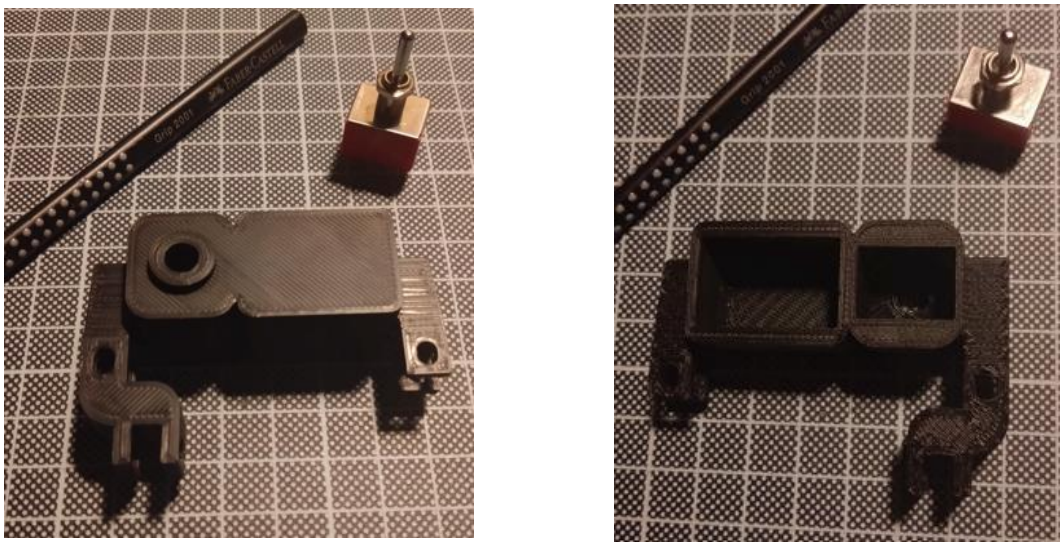
Εικόνα 2.85: Το τρισδιάστατο σχέδιο της βάσης στήριξης του τριπολικού διακόπτη και της επαναφορτιζόμενης μπαταρίας των 9 Volt [238].

Στην συνέχεια, αφού ολοκληρώθηκε η τρισδιάστατη σχεδίαση της βάση στήριξης του τριπολικού διακόπτη και της επαναφορτιζόμενης μπαταρίας, πραγματοποιήθηκε εξαγωγή του σχεδίου σε ένα αρχείο της μορφής .STL. Έπειτα το αρχείο αυτό εισήχθη στο λογισμικό τεμαχισμού UltiMaker Cura ώστε το σχέδιο να επεξεργαστεί περεταίρω. Η συγκεκριμένη βάση, θα πρέπει να στηρίξει το βάρος της επαναφορτιζόμενης μπαταρίας και στου τριπολικού διακόπτη, καθώς θα πρέπει να είναι ανθεκτική στις δυνάμεις που θα δέχεται όταν ο χρήστης θα ενεργοποιεί και θα απενεργοποιεί το σύστημα. Συνεπώς, μιας και το τρισδιάστατο σχέδιο της βάσης έχει κάποια σημεία τα οποία είναι αρκετά παχιά και κάποια άλλα πιο λεπτά, θα πρέπει να δημιουργηθεί ένα γέμισμα μεταξύ των στρωμάτων σχετικά πυκνό. Με αυτόν τον τρόπο η βάση θα είναι πιο ανθεκτική και δεν θα υπάρχει κίνδυνος παραμόρφωσης από το βάρος και από τις δυνάμεις που θα παράγονται όταν ο χρήστης ενεργοποιεί και απενεργοποιεί το σύστημα. Επίσης, η μορφή της βάσης έχει μία σχετικά περίεργη γεωμετρική μορφή που σημαίνει ότι κατά την διάρκεια της εκτύπωσης κάποια σημεία θα πέσουν εξαιτίας της βαρύτητας. Για την αντιμετώπιση του προβλήματος αυτού θα πρέπει μέσα από το λογισμικό τεμαχισμού να δημιουργηθεί υποστήριξη σε αυτά τα σημεία. Συνεπώς, μέσω του UltiMaker Cura πραγματοποιήθηκε υποστήριξη των σημείων αυτόν με την χρήση της δενδροειδούς μορφής (Εικόνα 2.86). Ουσιαστικά, αυτού του είδους η υποστήριξη προσφέρει οικονομία όσον αφορά το κόστος, διότι δεν καταναλώνει μεγάλη ποσότητα υλικού καθώς επίσης, μπορεί να αφαιρεθεί εύκολα χωρίς να γίνει κάποια φθορά στην βάση.

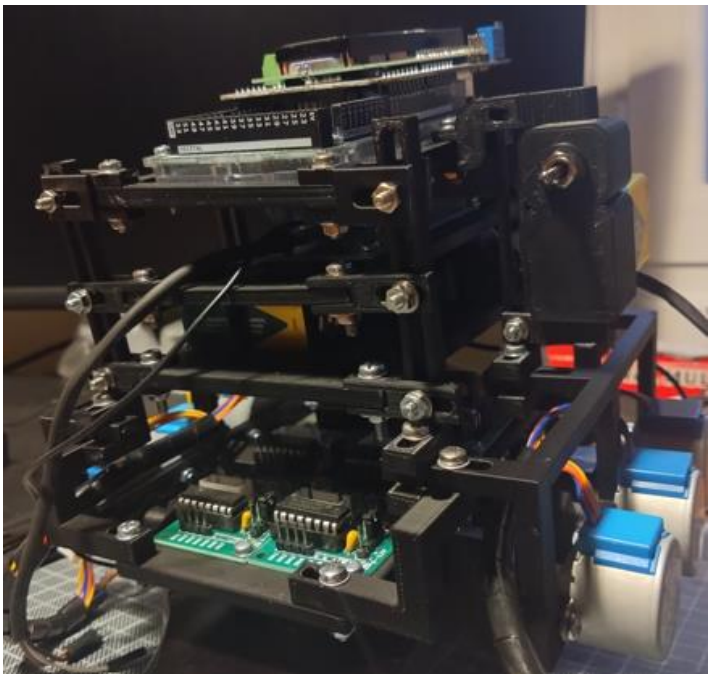


Εικόνα 2.86: Τα στρώματα, το γέμισμα μεταξύ των στρωμάτων και η δενδροειδής υποστήριξη της βάσης στήριξης του τριπολικού διακόπτη και της επαναφορτιζόμενης μπαταρίας στο λογισμικό τεμαχισμού UltiMaker Cura [239].

Αφότου ολοκληρώθηκε η επεξεργασία της βάσης στήριξης στο λογισμικό τεμαχισμού UltiMaker Cura, έγινε εξαγωγή του σχεδίου σε ένα αρχείο της μορφής .GCODE. Το αρχείο αυτό αποθηκεύτηκε στην κάρτα SD η οποία με την σειρά της εισήχθη στον τρισδιάστατο εκτυπωτή ώστε να πραγματοποιηθεί η εκτύπωση της βάσης στήριξης του τριπολικού διακόπτη και της επαναφορτιζόμενης μπαταρίας. Για την εκτύπωση της συγκεκριμένης βάσης, χρησιμοποιήθηκε βιοδιασπώμενο πλαστικό και πιο συγκεκριμένα PLA μαύρου χρώματος σε μορφή νήματος. Μετά την ολοκλήρωση της εκτύπωσης τοποθετήθηκε στην βάση ο τριπολικός διακόπτης και η επαναφορτιζόμενη μπαταρία των 9 Volt (Εικόνα 2.87). Τέλος, η τοποθέτηση του συστήματος επάνω στην κεφαλή ολοκληρώθηκε με την σταθεροποίησης της συγκεκριμένης βάσης επάνω στον αντάπτορα στήριξης και στην πλακέτα στήριξης του Arduino Mega 2560 με την χρήση βιδών, ροδελών και παξιμαδιών τύπου M3 (Εικόνα 2.88).



Εικόνα 2.87: Η βάση στήριξης του τριπολικού διακόπτη και της επαναφορτιζόμενης μπαταρίας των 9 Volt μετά την ολοκλήρωση της τρισδιάστατης εκτύπωσης [240].



Εικόνα 2.88: Η σταθεροποίηση της βάσης στήριξης του τριπολικού διακόπτη επάνω στην κεφαλή της κιθάρας [241].

3 ΚΕΦΑΛΑΙΟ 3ο : Επεξήγηση της λειτουργίας του συστήματος

Σε αυτό το κεφάλαιο αναλύεται ένα εξίσου σημαντικό κομμάτι για την κατανόηση του συστήματος αυτομάτου ελέγχου για το κούρδισμα των χορδών της κιθάρας. Ουσιαστικά, σε αυτό το κεφάλαιο επεξηγείται ο τρόπος λειτουργίας του συγκεκριμένου συστήματος καθώς και η σειρά με την οποία πρέπει να πραγματοποιηθούν τα γεγονότα ώστε να επιτευχθεί το κούρδισμα των χορδών της κιθάρας. Επομένως, για να γίνει κατανοητός ο τρόπος λειτουργίας του συστήματος αυτομάτου ελέγχου για το κούρδισμα των χορδών της κιθάρας, θα πρέπει να γίνει αντιληπτός τόσο ο κώδικας προγραμματισμού όσο και η σειρά με την οποία πρέπει να πραγματοποιηθούν τα γεγονότα από τον χρήστη.

3.1 Ανάλυση κώδικα προγραμματισμού

Σε αυτήν την ενότητα, πραγματοποιείται η ανάλυση του κώδικα προγραμματισμού ο οποίος αποτελεί αναπόσπαστο μέρος του συστήματος μιας και βρίσκεται αποθηκευμένος στην μνήμη του μικροελεγκτή. Ο κώδικας προγραμματισμού είναι ένα από τα σημαντικότερα κομμάτια του συγκεκριμένου συστήματος μιας και ελέγχει αλλά και καθορίζει την σειρά με την οποία θα λειτουργήσει το κάθε ένα υποσύστημα. Επιπλέον, ο κώδικας προγραμματισμού είναι υπεύθυνος για την λήψη του σήματος από το μικρόφωνο επαφής, την δειγματοληψία του και τον υπολογισμό της θεμελιώδης συχνότητας της χορδής. Επιπροσθέτως, ένα ακόμα σημαντικό πλεονέκτημα που πρέπει να έχει ο κώδικας, είναι ότι θα πρέπει να είναι φιλικός ως προς τον χρήστη και να συνεργάζονται αρμονικά κατά την λειτουργία του συστήματος. Πιο συγκεκριμένα, θα πρέπει ο κώδικας να προλαμβάνει τυχόν λάθη του χρήστη εμφανίζοντας μηνύματα στην LCD οθόνη τα οποία θα τον καθοδηγούν για το τι πρέπει να κάνει. Ουσιαστικά, το κομμάτι αυτό αποτελεί μια δικλείδα ασφαλείας τόσο ως προς τον χρήστη όσο και ως προς το ίδιο το σύστημα. Επιπλέον, ο κώδικας προγραμματισμού για την συγκεκριμένη διπλωματική εργασία, αναπτύχθηκε σε γλώσσα Wiring C η οποία αποτελεί μία παραλλαγή της γλώσσας προγραμματισμού C++. Ακόμα, ο συγγραφή του κώδικα προγραμματισμού πραγματοποιήθηκε στο Aruino IDE, το οποίο αποτελεί ένα ολοκληρωμένο περιβάλλον ανάπτυξης προγραμμάτων για πλακέτες Arduino. Η ανάλυση του κώδικα προγραμματισμού στο συγκεκριμένο κεφάλαιο γίνεται σε κομμάτια ώστε να είναι πιο εύκολο να κατανοηθεί από τον αναγνώστη. Επίσης, ο κώδικας προγραμματισμού της συγκεκριμένης εργασίας βρίσκεται ολόκληρος στο Παράρτημα Β μαζί με τα απαραίτητα σχόλια δίπλα από κάθε εντολή ώστε να είναι πιο εύκολο ως προς την κατανόησή του.

3.1.1 Ανάλυση των μεταβλητών, των σταθερών και της βιβλιοθήκης (Κώδικας προγραμματισμού Μέρος 1ο)

Αρχικά, στο συγκεκριμένο κώδικα προγραμματισμού γίνεται πρώτα απ’ όλα η κλήση των βιβλιοθηκών, ο καθορισμός σταθερών και η δήλωση και η αρχικοποίηση των μεταβλητών (Πίνακας 3.1). Στο πρώτο μέρος του κώδικα προγραμματισμού, γίνεται πρώτα κλήση της βιβλιοθήκης «LiquidCrystal.h» η οποία αποσκοπεί στην διευκόλυνση του ελέγχου της LCD οθόνης με την χρήση πιο απλών εντολών. Για την επίτευξη της κλήσης της βιβλιοθήκης, θα πρέπει να χρησιμοποιηθεί η εντολή (#include) και στην συνέχεια το όνομα της βιβλιοθήκης μέσα σε (< >).

Έπειτα, δημιουργούνται κάποιες σταθερές οι οποίες έχουν σκοπό να διευκολύνουν την εύρεση των ακροδεκτών ελέγχου των πλακετών οδήγησης των βηματικών κινητήρων.

Ουσιαστικά, με αυτόν τον τρόπο δεν χρειάζεται κάποιος να θυμάται τον αριθμό των ακροδεκτών του Arduino που χρησιμοποιεί, αλλά αρκεί να χρησιμοποιεί το όνομα που έχει δώσει στον κάθε ένα ακροδέκτη. Με αυτόν τον τρόπο αποφεύγονται τα λάθη και ο κώδικας προγραμματισμού αναπτύσσεται με μεγαλύτερη ευκολία. Ωστόσο, η δημιουργία των σταθερών επιτυγχάνεται με την χρήση της εντολής (#define) και στην συνέχεια ολοκληρώνεται γράφοντας το όνομα της σταθεράς και τέλος τον επιθυμητό αριθμό που θα αντιστοιχεί στην συγκεκριμένη σταθερά.

Εν συνεχεία, γίνονται δηλώσεις και αρχικοποιήσεις μεταβλητών τύπου boolean, character (char), byte, integer (int), unsigned integer (unsigned int) και float. Κάθε μία μεταβλητή από αυτές χρησιμοποιείται σε συγκεκριμένο κομμάτι του κώδικα προγραμματισμού και σκοπός τους είναι η αποθήκευση δεδομένων. Πιο συγκεκριμένα, οι μεταβλητές τύπου Boolean στο συγκεκριμένο κώδικα προγραμματισμού χρησιμοποιούνται ως σημαίες (flags), δηλαδή καθορίζουν το πότε έχει πραγματοποιηθεί ένα γεγονός και πότε όχι. Η μεταβλητή τύπου χαρακτήρα (char) αποσκοπεί στην αποθήκευση ενός μόνο στοιχείου είτε είναι αριθμός είτε είναι γράμμα υπό την μορφή χαρακτήρα ASCII. Έπειτα, στις μεταβλητές τύπου byte, μπορούν να αποθηκευτούν ακέραιοι αριθμοί μήκους ενός byte, Στο συγκεκριμένο πρόγραμμα κάποιες μεταβλητές τύπου byte χρησιμοποιούνται ως σημαίες (flags) ώστε να καθορίζουν πότε έχει πραγματοποιηθεί κάποιο γεγονός και πότε όχι, ενώ άλλες για την αποθήκευση δεδομένων από τον αναλογικό-ψηφιακό μετατροπέα και δεδομένων που προκύπτουν κατά την διάρκεια των υπολογισμών της θεμελιώδης συχνότητας ταλάντωσης της χορδής. Εν συνεχεία, χρησιμοποιούνται και κάποιες μεταβλητές τύπου integer στις οποίες μπορούν να αποθηκευτούν ακέραιοι αριθμοί μέχρι 2 byte ακόμα και με αρνητικό πρόσημο. Πιο συγκεκριμένα μία μεταβλητή τύπου integer μπορεί να αποθηκεύσει μία τιμή από -32768 έως 32767. Στο συγκεκριμένο πρόγραμμα, κάποιες μεταβλητές τύπου integer χρησιμοποιούνται ως μετρητές ενώ σε κάποιες άλλες αποθηκεύονται δεδομένα που χρησιμοποιούνται κατά την διάρκεια του υπολογισμού της θεμελιώδης συχνότητας. Επιπροσθέτως, στον συγκεκριμένο κώδικα προγραμματισμού χρησιμοποιούνται μεταβλητές τύπου unsigned integer οι οποίες χρησιμοποιούνται για την αποθήκευση μη προσημασμένων αριθμών και πιο συγκεκριμένα ακεραίων αριθμών από 0 έως 65535. Οι μεταβλητές τύπου unsigned integer στο συγκεκριμένο πρόγραμμα, χρησιμοποιούνται για την αποθήκευση χρονικών δεδομένων που προκύπτουν κατά την διάρκεια του υπολογισμού της θεμελιώδης συχνότητας ταλάντωσης της χορδής. Τα χρονικά δεδομένα πρέπει να είναι θετικά και γι’ αυτό τον λόγο οι συγκεκριμένες μεταβλητές είναι τύπου unsigned integer. Για την δημιουργία οποιουδήποτε μεταβλητής, θα πρέπει πρώτα να γραφτεί ο τύπος της μεταβλητής (πχ. float ή int ή char) και έπειτα το όνομα της μεταβλητής. Στην περίπτωση που η μεταβλητή είναι τύπου integer ή float ή byte ή unsigned integer και πρέπει να αρχικοποιηθεί, τότε θα πρέπει μετά το όνομα να προστεθεί το σύμβολο ίσον (=) και έπειτα η επιθυμητή τιμή. Στην περίπτωση όμως που η μεταβλητή είναι τύπου χαρακτήρα (char), η αρχικοποίηση γίνεται με την προσθήκη του συμβόλου ίσον (=) μετά το όνομα και έπειτα ακολουθεί ο χαρακτήρας ανάμεσα σε μονά εισαγωγικά (‘_’). Ωστόσο, αν είναι επιθυμητή η δήλωση πολλών μεταβλητών, τότε θα πρέπει αμέσως μετά το πρώτο όνομα ή μετά την ολοκλήρωση της αρχικοποίησης να τοποθετηθεί κώμα (,) και μετά το επόμενο όνομα. Αυτό συνεχίζεται μέχρι να γραφτεί και το τελευταίο όνομα ή να γίνει η τελική αρχικοποίηση και μετά τοποθετείται ερωτηματικό (;) ώστε να ολοκληρωθούν οι δηλώσεις και οι αρχικοποιήσεις των μεταβλητών.

Το πρώτο μέρος του κώδικα προγραμματισμού για το σύστημα αυτομάτου κουρδίσματος των χορδών, ολοκληρώνεται με την δημιουργία ενός αντικειμένου όπου αρχικοποιούνται κάποιο ακροδέκτες του Arduino. Επίσης, με την δημιουργία του αντικειμένου παρέχεται πρόσβαση στις συναρτήσεις της βιβλιοθήκης που έχει εισαχθεί στην αρχή. Στην συγκεκριμένη περίπτωση, χρησιμοποιείται η βιβλιοθήκη LiquidCrystal η οποία παρέχει συναρτήσεις για τον έλεγχο των LCD οθονών. Επομένως, για την δημιουργία ενός αντικειμένου, θα πρέπει πρώτα να γραφτεί το όνομα της βιβλιοθήκης, στην συνέχεια το όνομα του αντικειμένου και έπειτα οι ακροδέκτες πάνω στους οποίους έχει συνδεθεί η LCD οθόνη μέσα σε παρενθέσεις και χωρισμένοι με κώμα (,). Για την ολοκλήρωση της δημιουργίας του αντικειμένου πρέπει να γίνει χρήση του ερωτηματικού (;) στο τέλος. Στην προκειμένη περίπτωση, δημιουργείτε ένα αντικείμενο για την βιβλιοθήκη LiquidCrystal με όνομα lcd και οι ψηφιακοί ακροδέκτες που χρησιμοποιεί η LCD οθόνη είναι οι (4, 5, 6, 7, 8 & 9). Τέλος, με το όνομα του αντικειμένου lcd μπορεί να γίνει κλήση οποιασδήποτε συνάρτησης της βιβλιοθήκης.

Πίνακας 3.1: Το κομμάτι του κώδικα προγραμματισμού με την κλήση της βιβλιοθήκης, τον καθορισμό των σταθερών, την δήλωση και την αρχικοποίηση των μεταβλητών [27].

Κώδικας Προγραμματισμού (Μέρος 1ο)
#include <LiquidCrystal.h> //Liquid crystal library #define eStepper_Pin1 42 //define 1st pin for e string step motor #define eStepper_Pin2 43 //define 2nd pin for e string step motor #define eStepper_Pin3 44 //define 3rd pin for e string step motor #define eStepper_Pin4 45 //define 4th pin for e string step motor #define BStepper_Pin1 46 //define 1st pin for B string step motor #define BStepper_Pin2 47 //define 2nd pin for B string step motor #define BStepper_Pin3 48 //define 3rd pin for B string step motor #define BStepper_Pin4 49 //define 4th pin for B string step motor #define GStepper_Pin1 50 //define 1st pin for G string step motor #define GStepper_Pin2 51 //define 2nd pin for G string step motor #define GStepper_Pin3 52 //define 3rd pin for G string step motor #define GStepper_Pin4 53 //define 4th pin for G string step motor #define DStepper_Pin1 22 //define 1st pin for D string step motor #define DStepper_Pin2 23 //define 2nd pin for D string step motor #define DStepper_Pin3 24 //define 3rd pin for D string step motor #define DStepper_Pin4 25 //define 4th pin for D string step motor #define AStepper_Pin1 26 //define 1st pin for A string step motor #define AStepper_Pin2 27 //define 2nd pin for A string step motor #define AStepper_Pin3 28 //define 3rd pin for A string step motor #define AStepper_Pin4 29 //define 4th pin for A string step motor #define EStepper_Pin1 30 //define 1st pin for E string step motor

```
#define EStepper_Pin2 31 //define 2nd pin for E string step motor
#define EStepper_Pin3 32 //define 3rd pin for E string step motor
#define EStepper_Pin4 33 //define 4th pin for E string step motor

boolean clipping = 0, string_1 = 0, string_2 = 0, string_3 = 0, string_4 = 0, string_5 = 0, string_6 = 0; //Name and determine boolean variables
char note = ' '; //Name and determine char variable
byte Manual_Finish = 0, Manual_Select = 0, Finish_Control = 0, Turn_Right = 0, Turn_Left = 0; //Name and determine byte variable
byte newData = 0; //Name and determine byte variable
byte prevData = 0; //Name and determine byte variable
unsigned int time = 0; //Name and determine unsigned integer variable
int timer[10]; //Name and determine integer variable
int slope[10]; //Name and determine integer variable
unsigned int totalTimer; //Name and determine unsigned integer variable
unsigned int period; //Name and determine unsigned integer variable
byte index = 0; //Name and determine byte variable
float frequency; //Name and determine float variable
int maxSlope = 0; //Name and determine integer variable
int newSlope; //Name integer variable
byte noMatch = 0; //Name and determine byte variable
byte slopeTol = 3; //Name and determine byte variable
int timerTol = 10; //Name and determine integer variable
unsigned int ampTimer = 0; //Name and determine unsigned integer variable
byte maxAmp = 0; //Name and determine byte variable
byte checkMaxAmp; //Name byte variable
byte ampThreshold = 30; //Name and determine byte variable

LiquidCrystal lcd(8, 9, 4, 5, 6, 7); // Digital pin that Liquid Crystal screen will use

int x, k = 0, menu = 1, manual_menu = 1, i; //Name and determine integer variables for counters
```

3.1.2 **Ανάλυση της συνάρτησης setup (Κώδικας προγραμματισμού Μέρος 2ο)**

Σε αυτήν την ενότητα, γίνεται ανάλυση του κώδικα προγραμματισμού που περιέχεται στην συνάρτηση setup (πίνακας 3.2). Ξεκινώντας, είναι σημαντικό να αναφερθεί ξανά ότι η συνάρτηση setup εκτελείται μία φορά κατά την ενεργοποίηση του Arduino και σκοπός της είναι να φέρει το σύστημα στην αρχική του κατάσταση. Γενικά, σε αυτό το σημείο μπορεί να γίνει ενεργοποίηση της σειριακής επικοινωνίας, καθορισμός

των ακροδεκτών του Arduino ως είσοδοι ή έξοδοι και να εκτελεστούν κάποιες εντολές μόνο κατά την ενεργοποίηση του Arduino.

Αρχικά, στο συγκεκριμένο κομμάτι του κώδικα προγραμματισμού, γίνεται ενεργοποίηση της σειριακής επικοινωνίας μεταξύ του Arduino και του υπολογιστή μέσω της θύρας USB. Η ενεργοποίηση αυτή επιτυγχάνεται με την εντολή Serial.begin και μέσα σε παρενθέσεις καθορίζεται το baud rate. Ουσιαστικά, το baud rate καθορίζει τον ρυθμό με τον οποίο στέλνονται τα δεδομένα και θα πρέπει να είναι το ίδιο και στο Arduino και στην σειριακή οθόνη (serial monitor) που περιέχει το περιβάλλον ανάπτυξης Arduino IDE. Στην περίπτωση που τα baud rate δεν είναι τα ίδια τότε στην σειριακή οθόνη δεν εμφανίζεται κανένας χαρακτήρας ή εμφανίζεται οτιδήποτε άλλο εκτός από αυτό που πρέπει. Επίσης, με την σειριακή επικοινωνία μπορεί να στηθεί πιο εύκολα ένα πρόγραμμα, μιας και ο προγραμματιστής μπορεί να εκτυπώνει μηνύματα σε διάφορα σημεία του προγράμματος γνωρίζοντας ανά πάσα στιγμή σε πιο σημείο τρέχει ο κώδικας. Με αυτόν τον τρόπο, ο προγραμματιστής μπορεί να εντοπίζει τυχόν λάθη που έχουν δημιουργηθεί. και να δει αν ο κώδικας προγραμματισμού που έχει γράψει ανταποκρίνεται όπως επιθυμεί.

Εν συνεχεία, εκτελείται μία εντολή η οποία καθορίζει τις διαστάσεις της LCD οθόνης. Αυτό επιτυγχάνεται, με την χρήση της εντολής lcd.begin (), η οποία αποτελείται από το όνομα του αντικειμένου lcd και από την συνάρτηση begin() η οποία παρέχεται από την βιβλιοθήκη LiquidCrystal. Η συνάρτηση begin(), χρειάζεται στο εσωτερικό των παρενθέσεων δύο αριθμούς χωρισμένους με κώμα (,). Ο πρώτος αριθμός αποτελεί το πλήθος των στηλών (columns), δηλαδή το πλήθος των χαρακτήρων που χωρούν σε μία γραμμή και ο δεύτερος αριθμός αναφέρεται στο πλήθος των γραμμών (rows) που έχει η LCD οθόνη. Συνεπώς, στην συγκεκριμένη περίπτωση με την εντολή lcd.begin(16,2) καθορίζεται ότι η LCD οθόνη αποτελείται από 16 στήλες, δηλαδή χωράει 16 χαρακτήρες σε μία γραμμή και αποτελείται ακόμα από 2 γραμμές.

Έπειτα, εκτελούνται οι εντολές οι οποίες καθορίζουν τους ακροδέκτες ελέγχου των πλακετών οδήγησης των βηματικών κινητήρων ως έξοδοι. Αυτό επιτυγχάνεται με την χρήση της εντολής pinMode(), η οποία εντός των παρενθέσεων χρειάζεται δύο ακόμα στοιχεία για να ολοκληρωθεί. Πιο συγκεκριμένα, εντός των παρενθέσεων πρέπει πρώτα να γραφτεί ο αριθμός του ακροδέκτη ή αλλιώς το όνομα που έχει καθοριστεί για το συγκεκριμένο ακροδέκτη και έπειτα να οριστεί αν θα λειτουργήσει ως έξοδος (OUTPUT) ή ως είσοδος (INPUT). Στην προκειμένη περίπτωση, με την εντολή pinMode(eStepper_Pin1, Output) ορίζεται ότι ο ψηφιακός ακροδέκτης με όνομα eStepper_Pin1 ή αλλιώς ο ψηφιακός ακροδέκτης No 42 θα ενεργεί ως ψηφιακός ακροδέκτης εξόδου. Επομένως, σύμφωνα με τις υπόλοιπες παρόμοιες εντολές καθορίζονται το ίδιο και οι ακροδέκτες 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52 & 53. Τα ονόματα των ψηφιακών ακροδεκτών έχουν καθοριστεί στο 1^{ο} μέρος του κώδικα προγραμματισμού (Πίνακας 3.1).

Στην συνέχεια, εκτελούνται κάποιες εντολές οι οποίες έχουν να κάνουν με την LCD οθόνη. Σκοπός των εντολών αυτών είναι να εκτελεστούν μόνο μία φορά κατά την ενεργοποίηση του Arduino Mega 2560. Πιο συγκεκριμένα, χρησιμοποιείται η εντολή lcd.clear(), η εντολή lcd.setCursor() και η εντολή lcd.print(“ ”). Στην προκειμένη περίπτωση και οι τρεις εντολές, χρησιμοποιούν το όνομα του αντικειμένου lcd που έχει δημιουργηθεί για την βιβλιοθήκη LiquidCrystal ώστε να επιτευχθεί η πρόσβαση στις συναρτήσεις της βιβλιοθήκης. Πιο συγκεκριμένα, η εντολή lcd.clear(), περιλαμβάνει την συνάρτηση clear() από την βιβλιοθήκη LiquidCrystal και είναι υπεύθυνη για τον

καθαρισμό των χαρακτήρων από την LCD οθόνη. Από την άλλη, η εντολή lcd.setCursor(), περιλαμβάνει την συνάρτηση setCursor() της βιβλιοθήκης LiquidCrystal και σκοπός της είναι να μετακινήσει τον κέρσορα στην επιθυμητή στήλη και γραμμή. Για να επιτευχθεί αυτό θα πρέπει εντός των παρενθέσεων να γραφτούν δύο αριθμοί χωρισμένοι με κώμα (,). Ο πρώτος αριθμός αναφέρεται στην επιθυμητή στήλη που πρέπει να μετακινηθεί ο κέρσορας και ο δεύτερος αριθμός αναφέρεται στην επιθυμητή γραμμή στην οποία θα μετακινηθεί ο κέρσορας της LCD οθόνης. Αφετέρου, η εντολή lcd.print(), χρησιμοποιεί την συνάρτηση print() της βιβλιοθήκης LiquidCrystal και σκοπός της είναι να εκτυπώσει τους επιθυμητούς χαρακτήρες από το σημείο που είναι ο κέρσορας και μετά. Για να επιτευχθεί αυτό θα πρέπει εντός της παρένθεσης και εντός διπλών εισαγωγικών (“ ”) να εισαχθούν οι επιθυμητοί χαρακτήρες. Σε περίπτωση όμως που είναι επιθυμητή η εκτύπωση της τιμής μιας μεταβλητής, τότε εντός τον παρενθέσεων γράφεται το όνομα της μεταβλητής χωρίς την χρήση διπλών εισαγωγικών (“ ”). Επομένως, σύμφωνα με τα όσα αναφέρθηκαν, οι συγκεκριμένες εντολές εμφανίζουν το μήνυμα AUTOMATIC GUITAR TUNER στην LCD οθόνη το οποίο παραμένει για 2.5 sec πριν σβηστεί. Για την παραμονή του μηνύματος στην LCD οθόνη χρησιμοποιείται η εντολή delay() η οποία έχει σκοπό την δημιουργία μιας καθυστέρησης για όσα msec έχουν γραφτεί εντός των εισαγωγικών. Δηλαδή, η εντολή delay(2500) δημιουργεί μια καθυστέρηση 2500 msec ή αλλιώς 2.5 sec. Έπειτα, αφού σβηστεί το πρώτο μήνυμα εμφανίζεται το δεύτερο μήνυμα Choose your Tune, το οποίο έχει σκοπό να προτρέψει τον χρήστη να επιλέξει το κούρδισμα που επιθυμεί. Έπειτα, δημιουργείται ένας βρόγχος επανάληψης (for) ο οποίος εκτελεί το μπλοκ εντολών του έξι φορές με μία καθυστέρηση 0.5sec κάθε φορά εξαιτίας της καθυστέρησης delay(500) που υπάρχει εντός του μπλοκ εντολών. Επίσης, εντός του μπλοκ εντολών του βρόγχου επανάληψης (for) υπάρχει η εντολή lcd.setCursor(), η οποία σε κάθε επανάληψη τοποθετεί τον κέρσορα σε διαφορετική στήλη και η εντολή lcd.print(“ ”) η οποία εκτυπώνει έναν χαρακτήρα στο σημείο που δείχνει ο κέρσορας στην LCD οθόνη. Ουσιαστικά, με αυτόν τον τρόπο επιτυγχάνεται η εκτύπωση έξι κουκίδων στην δεύτερη γραμμή της LCD οθόνης μία προς μία. Αυτό έχει ως αποτέλεσμα να τραβάει την προσοχή του χρήστη και να του δίνει τον απαραίτητο χρόνο να διαβάσει το μήνυμα.

Ολοκληρώνοντας, η τελευταία εντολή που εκτελείται στην συνάρτηση setup() αποσκοπεί στην κλήση της συνάρτησης Menu(). Ουσιαστικά, η συνάρτηση αυτή εμφανίζει στην LCD οθόνη το μενού με τα κουρδίσματα που περιέχει το σύστημα αυτομάτου ελέγχου για το κούρδισμα των χορδών της κιθάρας. Ωστόσο, η συνάρτηση Menu(), αναλύεται στο τρίτο μέρος του κώδικα προγραμματισμού στην επόμενη ενότητα.

Πίνακας 3.2: Ο κώδικας προγραμματισμού της συνάρτησης setup [28].

Κώδικας Προγραμματισμού (Μέρος 2ο)

void setup() { //Set up function
Serial.begin(9600); //Begin Serial communicate at 9600 Baud rate
lcd.begin(16,2); //Determine Liquid Crystal dimensions
pinMode(eStepper_Pin1, OUTPUT); //determine 1st pin for e string step motor as an output
pinMode(eStepper_Pin2, OUTPUT); //determine 2nd pin for e string step motor as an output
pinMode(eStepper_Pin3, OUTPUT); //determine 3rd pin for e string step motor as an

```
output
  pinMode(eStepper_Pin4, OUTPUT); //determine 4th pin for e string step motor as an
output
  pinMode(BStepper_Pin1, OUTPUT); //determine 1st pin for B string step motor as an
output
  pinMode(BStepper_Pin2, OUTPUT); //determine 2nd pin for B string step motor as an
output
  pinMode(BStepper_Pin3, OUTPUT); //determine 3rd pin for B string step motor as an
output
  pinMode(BStepper_Pin4, OUTPUT); //determine 4th pin for B string step motor as an
output
  pinMode(GStepper_Pin1, OUTPUT); //determine 1st pin for G string step motor as an
output
  pinMode(GStepper_Pin2, OUTPUT); //determine 2nd pin for G string step motor as an
output
  pinMode(GStepper_Pin3, OUTPUT); //determine 3rd pin for G string step motor as an
output
  pinMode(GStepper_Pin4, OUTPUT); //determine 4th pin for G string step motor as an
output
  pinMode(DStepper_Pin1, OUTPUT); //determine 1st pin for D string step motor as an
output
  pinMode(DStepper_Pin2, OUTPUT); //determine 2nd pin for D string step motor as an
output
  pinMode(DStepper_Pin3, OUTPUT); //determine 3rd pin for D string step motor as an
output
  pinMode(DStepper_Pin4, OUTPUT); //determine 4th pin for D string step motor as an
output
  pinMode(AStepper_Pin1, OUTPUT); //determine 1st pin for A string step motor as an
output
  pinMode(AStepper_Pin2, OUTPUT); //determine 2nd pin for A string step motor as an
output
  pinMode(AStepper_Pin3, OUTPUT); //determine 3rd pin for A string step motor as an
output
  pinMode(AStepper_Pin4, OUTPUT); //determine 4th pin for A string step motor as an
output
  pinMode(EStepper_Pin1, OUTPUT); //determine 1st pin for E string step motor as an
output
  pinMode(EStepper_Pin2, OUTPUT); //determine 2nd pin for E string step motor as an
output
  pinMode(EStepper_Pin3, OUTPUT); //determine 3rd pin for E string step motor as an
output
  pinMode(EStepper_Pin4, OUTPUT); //determine 4th pin for E string step motor as an
output
```

```
 lcd.setCursor(0,0); //Set cursor at position 0 column and 0 line
 lcd.print("AUTOMATIC GUITAR"); //Print message at LCD screen
 lcd.setCursor(5,1); //Set cursor at position 5 column and 1 line
 lcd.print("TUNER!"); //Print message at LCD
 delay(2500); //Time delay for 2.5 sec
 lcd.clear(); //Clear LCD screen
 lcd.setCursor(0,0); //Set cursor at position 0 column and 0 line
 lcd.print("Choose your Tune"); //Print message at LCD screen
 for (i=0;i<=5;i++) //for 6 times repeat
 {
  lcd.setCursor((i+5),1); //Set Cursor at position i+5 column and 1 line
  lcd.print("."); //Print a Dot at LCD screen
  delay(500); //Delay for 500 msec
 }
 Menu(); //Call Menu(); function
}//void setup ()
```

3.1.3 **Ανάλυση της συνάρτησης loop και της συνάρτησης Menu (Κώδικας προγραμματισμού Μέρος 3ο)**

Μετά την ολοκλήρωση της εκτέλεσης των εντολών της συνάρτησης setup, εκτελείται το μπλοκ εντολών της συνάρτησης loop για όσο το Arduino Mega 2560 είναι ενεργοποιημένο. Πιο συγκεκριμένα, το μπλοκ εντολών της συνάρτησης loop(), εκτελείται ξανά κάθε φορά που ολοκληρώνεται η εκτέλεση του. Συνεπώς, στην συγκεκριμένη ενότητα αναλύεται ο κώδικας προγραμματισμού για την συνάρτηση loop(), αλλά και η συνάρτηση Menu() (Πίνακας 3.3), η οποία κλήθηκε στο τέλος της συνάρτησης setup() και αποτελεί ένα από τα σημαντικότερα μέρη του προγράμματος.

Αρχικά, το μπλοκ εντολών της συνάρτησης loop() αποτελείται από μία μόνο εντολή η οποία καλεί την συνάρτηση ButtonRead(). Η συνάρτηση αυτή καλείται συνέχεια κάθε φορά που ολοκληρώνεται, διότι βρίσκεται εντός της συνάρτησης loop(). Η συνάρτηση ButtonRead() αποτελεί την βασικότερη συνάρτηση του προγράμματος, διότι μέσω αυτής επιλέγεται και πραγματοποιείται το κούρδισμα της κιθάρας. Ουσιαστικά, αναγνωρίζει κάθε φορά πιο κουμπί έχει πιέσει ο χρήστης και ανάλογα αλλάζει τις ενδείξεις στην LCD οθόνη είτε για να αλλάξει το επιθυμητό κούρδισμα είτε για να ξεκινήσει το κούρδισμα που επιλέχτηκε. Συνεπώς, επειδή η συνάρτηση ButtonRead() είναι αξιοσημείωτη και αποτελεί ένα μεγάλο μέρος του προγράμματος, αναλύεται στην επόμενη ενότητα για να μην δημιουργηθεί κάποια σύγχυση.

Στην συνέχεια, ορίζεται η συνάρτηση Menu() μαζί με το μπλοκ εντολών της. Η συνάρτηση αυτή καλείτε για πρώτη φορά στο τέλος της συνάρτησης setup() και σκοπός της είναι να εμφανίσει στον χρήστη το μενού με τα κουρδίσματα που παρέχει το σύστημα αυτομάτου ελέγχου για το κούρδισμα των χορδών της κιθάρας. Στο μπλοκ εντολών της συνάρτησης Menu() για την εμφάνιση του μενού στην LCD οθόνη, χρησιμοποιείται η

τεχνική switch(). Ουσιαστικά, στην τεχνική αυτή εισάγεται εντός των παρενθέσεων του switch() μία μεταβλητή η οποία δέχεται μόνο ακέραιους αριθμούς. Στην συγκεκριμένη περίπτωση η μεταβλητή αυτή ονομάζεται menu και είναι τύπου integer. Επίσης, η τεχνική switch() χρησιμοποιεί περιπτώσεις (case), όπου ανάλογα με την τιμή που έχει η integer μεταβλητή menu εκτελείται και το αντίστοιχο μπλοκ εντολών. Πιο συγκεκριμένα, όταν η μεταβλητή menu έχει την τιμή μηδέν, τότε σύμφωνα με το μπλοκ εντολών της περίπτωσης μηδέν (case 0) αποθηκεύεται στην μεταβλητή menu η τιμή ένα και μετά εκτελείται η εντολή break η οποία έχει ως αποτέλεσμα την έξοδο από το switch(). Στην περίπτωση που η μεταβλητή menu έχει την τιμή ένα, τότε εκτελείται το μπλοκ εντολών της περίπτωσης ένα (case 1). Σε αυτήν την περίπτωση, σβήνονται οι χαρακτήρες από την LCD οθόνη και έπειτα εκτυπώνεται στην πρώτη γραμμή το μήνυμα >Standard Tune και στην δεύτερη γραμμή το μήνυμα Half Step Down. Με το σύμβολο (>) πριν το Standard Tune υποδηλώνεται στον χρήστη πιο κούρδισμα είναι επιλεγμένο. Μετά την εμφάνιση του μηνύματος, εκτελείται η εντολή break ώστε να πραγματοποιηθεί έξοδος από το switch(). Στην περίπτωση που η μεταβλητή menu έχει την τιμή δύο, τότε εκτελείται το μπλοκ εντολών της περίπτωσης δύο (case 2). Ουσιαστικά, σε αυτήν την περίπτωση σβήνονται οι χαρακτήρες από την LCD οθόνη και τυπώνεται στην πρώτη γραμμή το μήνυμα Standard Tune και στην δεύτερη γραμμή το μήνυμα >Half Step Down. Με το σύμβολο (>) πριν το Half Step Down υποδηλώνεται στον χρήστη πιο κούρδισμα είναι επιλεγμένο. Μετά την ολοκλήρωση της εμφάνισης του μηνύματος στην LCD οθόνη, εκτελείται η εντολή break η οποία έχει ως αποτέλεσμα την έξοδο από το switch(). Στην περίπτωση όμως που η μεταβλητή menu έχει την τιμή τρία, τότε εκτελείται το μπλοκ εντολών της περίπτωσης τρία (case 3). Πιο συγκεκριμένα σε αυτήν την περίπτωση, πρώτα σβήνονται οι χαρακτήρες από την LCD οθόνη, έπειτα τυπώνεται στην πρώτη γραμμή το μήνυμα >Drop D Tune και ύστερα στην δεύτερη γραμμή τυπώνεται το μήνυμα Manual Control. Με το σύμβολο (>) πριν το Drop D Tune υποδηλώνεται στον χρήστη πιο κούρδισμα είναι επιλεγμένο. Μετά την εμφάνιση του μηνύματος στην LCD οθόνη, εκτελείται η εντολή break η οποία έχει ως αποτέλεσμα την έξοδο από το switch(). Εν συνεχεία, στην περίπτωση που η μεταβλητή menu έχει την τιμή τέσσερα, τότε εκτελείται το μπλοκ εντολών της περίπτωσης τέσσερα (case 4). Σε αυτήν την περίπτωση, σβήνονται όλοι οι χαρακτήρες από την LCD οθόνη και τυπώνεται στην πρώτη γραμμή το μήνυμα Drop D Tune και έπειτα στην δεύτερη γραμμή το μήνυμα >Manual Control. Με το σύμβολο (>) πριν το Manual Control υποδηλώνεται στον χρήστη ότι είναι επιλεγμένος ο χειροκίνητος έλεγχος του συστήματος. Τέλος, στην περίπτωση όπου η μεταβλητή menu έχει την έχει την τιμή πέντε, τότε σύμφωνα με το μπλοκ εντολών της περίπτωσης πέντε (case 5) αποθηκεύεται στην μεταβλητή menu η τιμή τέσσερα και μετά εκτελείται η εντολή break η οποία έχει ως αποτέλεσμα την έξοδο από το switch().

Ολοκληρώνοντας, είναι ξεκάθαρο πως η συνάρτηση Menu () αποτελεί ουσιαστικά ένα βασικό κομμάτι του κώδικα μιας και εμφανίζει στον χρήστη τα πιθανά κουρδίσματα καθώς και την δυνατότητα του χειροκίνητου ελέγχου. Δηλαδή, η συγκεκριμένη συνάρτηση συμβάλει στην ομαλή αλληλεπίδραση μεταξύ χρήστη και συστήματος.

Πίνακας 3.3: Ο κώδικας προγραμματισμού της συνάρτησης setup [29].

Κώδικας Προγραμματισμού (Μέρος 3ο)
void loop() { //Loop function

```
  ButtonRead(); //Go at ButtonRead(); function
}//loop end


void Menu() { //Menu function
  switch (menu){ //Check menu number with switch
  case 0: //In case that menu = 0
  menu = 1; //Store number 1 at menu variable
  break; //Break the switch loop
  case 1: //In case that menu = 1
  lcd.clear(); //Clear LCD screen
  lcd.print(">Standard Tune"); //Print message at LCD screen
  lcd.setCursor(0,1); //Set cursor at 1st column and 2nd line
  lcd.print(" Half Step Down"); //Print message at LCD screen
  break; //Break the switch loop
  case 2: //In case that menu = 2
  lcd.clear(); //Clear LCD screen
  lcd.print(" Standard Tune"); //Print message at LCD screen
  lcd.setCursor(0,1); //Set cursor at 1st column and 2nd line
  lcd.print(">Half Step Down"); //Print message at LCD screen
  break; //Break switch loop
  case 3: //In case that menu = 3
  lcd.clear(); //Clear LCD screen
  lcd.print(">Drop D Tune"); //Print message at LCD screen
  lcd.setCursor(0,1); //Set cursor at 1st column and 2nd line
  lcd.print(" Manual Control"); //Print message at LCD screen
  break; //Break switch loop
  case 4: //In case that menu = 4
  lcd.clear(); //Clear LCD screen
```

```
  lcd.print(" Drop D Tune"); //Print message at LCD screen
  lcd.setCursor(0,1); //Set cursor at 1st column and 2nd line
  lcd.print(">Manual Control"); //Print message at LCD screen
  break; //Break switch loop
  case 5: //In case that menu = 5
  menu = 4; //Store number 4 at menu Variable
  break; //Break switch loop
  } //switch menu
}
```

3.1.4 **Ανάλυση της συνάρτησης ButtonRead (Κώδικας προγραμματισμού μέρος 4ο, 5ο, 6ο, 7ο & 8ο)**

Σε αυτήν την ενότητα, γίνεται ανάλυση του σημαντικότερου σημείου του κώδικα προγραμματισμού. Ουσιαστικά, η συνάρτηση ButtonRead () είναι αυτή που συμβάλει τόσο στην ομαλή συνεργασία μεταξύ χρήστη και συστήματος όσο και στην επίτευξη του κουρδίσματος των χορδών του οργάνου. Επιπροσθέτως, εξαιτίας του ογκώδες κώδικα προγραμματισμού από τον οποίο αποτελείται η συγκεκριμένη συνάρτηση, η ανάλυσή του θα γίνει σε κομμάτια ώστε να αποφευχθεί οποιαδήποτε σύγχυση.

Αρχικά, το πρώτο κομμάτι της συνάρτησης ButtonRead(), έχει να κάνει με την αναγνώριση των πλήκτρων UP, DOWN, LEFT και RIGHT που περιλαμβάνονται στην πλακέτα της LCD οθόνης (Πίνακας 3.4). Πιο συγκεκριμένα, για την επίτευξη της ανάγνωσης των πλήκτρων που περιλαμβάνει η LCD οθόνη, θα πρέπει να γίνει ενεργοποίηση της αναλογικής θύρας (A0) του Arduino. Θα μπορούσε εύκολα να πραγματοποιηθεί δειγματοληψία μέσω της αναλογικής θύρας (A0) με την χρήση της εντολής analogRead(A0). Αντί αυτού όμως, η έναρξη της δειγματοληψίας από την αναλογική θύρα (A0) πραγματοποιείται με την χρήση πιο πολύπλοκων εντολών. Ουσιαστικά, οι εντολές αυτές καθορίζουν τα bit των καταχωρητών που είναι υπεύθυνοι για την έναρξη της δειγματοληψίας από την αναλογική θύρα (A0). Πρώτα απ' όλα, πριν ξεκινήσει η εναπόθεση των bit, γίνεται εκκαθάριση των καταχωρητών ADCSRA, ADCSRB και ADMUX. Η εκκαθάριση του κάθε καταχωρητή επιτυγχάνεται με τις εντολές ADCSRA = 0, ADCSRB = 0 και ADMUX = 0. Στην συνέχεια, εκτελούνται οι εντολές για την εναπόθεση των bit των καταχωρητών, ξεκινώντας από τον καταχωρητή ADMUX. Πιο συγκεκριμένα, εκτελείται η εντολή ADMUX |=(1<<REFS0)|(0<<REFS1) η οποία εναποθέτει (1) στο bit (REFS0) και (0) στο bit (REFS1) του καταχωρητή ADMUX. Με την εκτέλεση αυτής της εντολής καθορίζεται η τάση αναφοράς του αναλογικό-ψηφιακού μετατροπέα. Έπειτα, εκτελείται η εντολή ADCSRB |=(1<<ACME) η οποία εναποθέτει (1) στο bit ACME του καταχωρητή ADCSRB. Με την εκτέλεση αυτής της εντολής, ενεργοποιείται ο πολυπλέκτης του αναλογικού συγκριτή. Ύστερα, εκτελείται η εντολή ADCSRA |=(1<<ADEN) όπου εναποθέτει (1) στο bit ADEN του καταχωρητή

ADCSRA. Αφού ολοκληρωθεί η εκτέλεση αυτής της εντολής, τότε ενεργοποιείται ο αναλογικό-ψηφιακός μετατροπέας. Στην συνέχεια, εκτελείται η εντολή ADCSRA |=(1<<ADSC) η οποία εναποθέτει (1) στο bit ADSC του καταχωρητή ADCSRA. Με την εκτέλεση αυτής της εντολής, γίνεται εκκίνηση της δειγματοληψίας από τον αναλογικό-ψηφιακό μετατροπέα. Για όσο διαρκεί η δειγματοληψία, το bit αυτό παραμένει (1) και μετά την ολοκλήρωση της δειγματοληψίας, το bit γίνεται (0). Όπως είναι φανερό, μέχρι στιγμής δεν έχει εκτελεστεί κάποια εντολή η οποία να δηλώνει την αναλογική θύρα εισόδου του σήματος. Αφού λοιπόν δεν έχει δηλωθεί κάποια αναλογική θύρα εισόδου, ο αναλογικό-ψηφιακός μετατροπέας λαμβάνει το σήμα αυτόματα από την αναλογική θύρα (A0) του Arduino. Επιπροσθέτως, μετά την ολοκλήρωση της δειγματοληψίας, ο αναλογικό-ψηφιακός μετατροπέας αποθηκεύει το αποτέλεσμα που προκύπτει στον καταχωρητή ADC. Έπειτα, εκτελείται η εντολή x = ADC όπου το αποτέλεσμα από τον καταχωρητή αποθηκεύεται στην μεταβλητή τύπου integer με όνομα (x). Στην συνέχεια, ακολουθεί μία καθυστέρηση 0.2sec και γίνεται εκτύπωση της τιμής της μεταβλητής (x) στην σειριακή οθόνη. Με αυτόν τον τρόπο, ο προγραμματιστής μπορεί να διαβάζει κάθε φορά την τιμή που προκύπτει μετά την ολοκλήρωση της δειγματοληψίας από τον αναλογικό-ψηφιακό μετατροπέα.

Στην συνέχεια, πραγματοποιείτε έλεγχος της τιμής της μεταβλητής (x) ώστε να εντοπιστεί το κουμπί που πάτησε ο χρήστης. Αυτό επιτυγχάνεται μέσω της συνθήκης ελέγχου (if-else if). Πιο συγκεκριμένα, με την εντολή if (x < 60 && x > 0) ελέγχεται αν η τιμή της μεταβλητής (x) είναι μικρότερη του εξήντα και μεγαλύτερη του μηδενός. Αν ικανοποιείται αυτή η συνθήκη, τότε εκτελείται το μπλοκ εντολών της εντολής (if) και αυτό σημαίνει ότι πατήθηκε το κουμπί RIGHT. Κατά την εκτέλεση του μπλοκ εντολών της συνθήκης (if), γίνεται καθαρισμός της LCD οθόνης από του χαρακτήρες μέσω της εντολής lcd.clear() και επανατοποθέτηση του κέρσορα σε συγκεκριμένο σημείο μέσω της εντολής lcd.setCursor(). Έπειτα, γίνεται εκτύπωση χαρακτήρων στην LCD οθόνη μέσω της εντολής lcd.print(“ “), ώστε να προκύψει το μήνυμα (USE UP OR DOWN KEYS!!). Το μήνυμα αυτό θα παραμείνει στην οθόνη για 2.5 sec διότι δημιουργείται μια καθυστέρηση με την χρήση της εντολής delay(2500). Ύστερα, γίνεται κλήση της συνάρτησης Menu(), η οποία σβήνει το προηγούμενο μήνυμα από την LCD και εμφανίζει ξανά το μενού στην LCD οθόνη. Μετά, δημιουργείται μια καθυστέρηση 0.2 sec εξαιτίας της εντολής delay(200). Ουσιαστικά, με αυτό το μπλοκ εντολών επιτυγχάνεται η εμφάνιση μηνύματος σφάλματος στην LCD οθόνη ώστε ο χρήστης να καταλάβει ότι πρέπει να πατήσει διαφορετικό πλήκτρο για να αλλάξει την επιλογή στο μενού.

Αν όμως, δεν ικανοποιείται η συνθήκη (if), τότε εκτελείται η εντολή else if(x < 200), η οποία ελέγχει αν η μεταβλητή (x) είναι μικρότερη από διακόσια. Στην περίπτωση που ικανοποιείται η συνθήκη αυτή, τότε εκτελείται το μπλοκ εντολών της εντολής (else if) και αυτό σημαίνει ότι πατήθηκε το κουμπί UP. Στην συνέχεια, κατά την εκτέλεση του μπλοκ εντολών της συνθήκης (else if) γίνεται μείωση της μεταβλητής (menu) κατά μία μονάδα διότι εκτελείται η εντολή (menu--). Έπειτα, γίνεται κλήση της συνάρτησης Menu() ώστε να πραγματοποιηθεί η αντίστοιχη μεταβολή του βέλους στο μενού ανάλογα με την τιμή που έχει η μεταβλητή (menu). Ύστερα, εκτελείται η εντολή delay(200), η οποία δημιουργεί μία καθυστέρηση των 0.2 sec. Ουσιαστικά, με αυτή την συνθήκη το σύστημα αντιλαμβάνεται πότε ο χρήστης πίεσε το κουμπί UP και σκοπός της είναι να αλλάξει την επιλογή στο μενού.

Στην περίπτωση όμως, που δεν ικανοποιείται η προηγούμενη συνθήκη, τότε εκτελείται η επόμενη εντολή else if (x < 400), η οποία ελέγχει αν η τιμή της μεταβλητής (menu) είναι μικρότερη από τετρακόσια. Στην περίπτωση λοιπόν, που ικανοποιείται αυτή η συνθήκη, τότε εκτελείται το αντίστοιχο μπλοκ της εντολής else if και αυτό σημαίνει ότι πατήθηκε το κουμπί DOWN. Κατά την εκτέλεση λοιπόν, του συγκεκριμένου μπλοκ εντολών (else if), εκτελείται η εντολή (menu++) η οποία αυξάνει την τιμή της μεταβλητής (menu) κατά μία μονάδα. Ύστερα, γίνεται κλήση της συνάρτησης Menu(), ώστε να πραγματοποιηθεί η αντίστοιχη μεταβολή του βέλους στο μενού επιλογών ανάλογα με την τιμή που έχει η μεταβλητή (menu). Έπειτα, εκτελείται η εντολή delay(200), η οποία δημιουργεί μία χρονική καθυστέρηση των 0.2 sec. Ουσιαστικά, με αυτή την συνθήκη το σύστημα αντιλαμβάνεται πότε ο χρήστης έχει πιέσει το κουμπί DOWN και σκοπός της είναι να μεταβάλει την επιλογή στο μενού.

Από την άλλη πλευρά, αν δεν ικανοποιείται η προηγούμενη συνθήκη, τότε εκτελείται η εντολή else if (x < 600 && x > 402), η οποία ελέγχει αν η τιμή της μεταβλητής (x) είναι μικρότερη από εξακόσια και μεγαλύτερη από τετρακόσια δύο. Στην περίπτωση που ικανοποιείται η συγκεκριμένη συνθήκη (else if), τότε εκτελείται το αντίστοιχο μπλοκ εντολών της εντολής else if. Η εκτέλεση του συγκεκριμένου μπλοκ εντολών περιλαμβάνει την εντολή lcd.clear() η οποία σβήνει τους χαρακτήρες που υπάρχουν στην LCD οθόνη και περιλαμβάνει ακόμα την εντολή lcd.setCursor(), η οποία έχει σκοπό την μετακίνηση του κέρσορα σε συγκεκριμένη στήλη και γραμμή της LCD οθόνης. Επίσης, περιλαμβάνει την εντολή lcd.print(“ “) η οποία τυπώνει χαρακτήρες στην LCD οθόνη ώστε να προκύψει το μήνυμα (USE UP OR DOWN KEYS!!). Το μήνυμα αυτό παραμένει στην LCD οθόνη για 2.5sec, διότι εκτελείται η εντολή καθυστέρησης delay(2500). Έπειτα, γίνεται κλήση της συνάρτησης Menu(), ώστε στην LCD οθόνη να εμφανιστεί ξανά το μενού επιλογών. Ύστερα, πραγματοποιείται μια καθυστέρηση των 0.2 sec εξαιτίας της εντολής delay(200). Στόχος του συγκεκριμένου μπλοκ εντολών, είναι η επίτευξη της εμφάνισης ενός μηνύματος σφάλματος στην LCD οθόνη ώστε ο χρήστης να καταλάβει ότι πρέπει να πατήσει διαφορετικό πλήκτρο για να αλλάξει την επιλογή στο μενού.

Στο 4^{ο} μέρος του κώδικα προγραμματισμού (πίνακα 3.4) αναλύεται ουσιαστικά το κομμάτι αναγνώρισης των πλήκτρων UP, DOWN, LEFT και RIGHT. Σκοπός του συγκεκριμένου κομματιού κώδικα είναι η εναλλαγή των επιλογών στο μενού που εμφανίζεται στην LCD οθόνη. Ουσιαστικά, ο χρήστης έχει την δυνατότητα να επιλέξει την επιθυμητή ενέργεια, είτε κουρδίσματος είτε χειροκίνητου ελέγχου του συστήματος, μέσα από το μενού, χρησιμοποιώντας τα πλήκτρα που βρίσκονται ενσωματωμένα στην πλακέτα της LCD οθόνης.

Πίνακας 3.4: Η ανάλυση του κώδικα προγραμματισμού της συνάρτησης ButtonRead() για την ανάγνωση των πλήκτρων [30].

Κώδικας Προγραμματισμού (Μέρος 4^{ο})

void ButtonRead() //ButtonRead function
{
//x=analogRead(A0);
ADCSRA = 0; //Clear ADCSRA register
ADCSRB = 0; //Clear ADCSRB register

```
 ADMUX = 0; //Clear ADMUX register
 ADMUX |= (1<<REFS0)|(0<<REFS1); //Set bit REFS0 and clear bit REFS1 at ADMUX register, set reference voltage
 ADCSRB |= (1<<ACME); //Set bit ACME at ADCSRB register and enable analog comperator multiplexer
 ADCSRA |=(1<<ADEN); //Set bit ADEN at ADCSRA register and enable ADC
 ADCSRA |= (1<<ADSC); //Set bit ADSC at ADCSRA register and start ADC measurments
 x = ADC; //Store value from ADC to x variable
 delay(200); //Delay 200 msec
 Serial.println(x); //Print x value at Serial port

 if (x<60 && x>0) //If RIGHT button has been pressed, then
 {
  lcd.clear(); //Clear LCD screen
  lcd.setCursor(1,0); //Set cursor at first column and first line
  lcd.print("USE UP OR DOWN"); //Print message at LCD screen
  lcd.setCursor(5,1); //Set cursor at first column and first line
  lcd.print("KEYS!!"); //Print message at LCD screen
  delay(2500); //Delay 2.5sec
  Menu(); //Call Menu function
  delay(200); //Delay 200 msec
 }
 else if(x<200) //If UP button has been pressed
 {
  menu--; //Decrease menu by one point
  Menu(); //Call function Menu
  delay(200); //Delay 200 msec
 }
 else if(x<400) //If DOWN button has been pressed
 {
  menu++; //Increase menu by one point
  Menu(); //Call Menu function
  delay(200); //Delay 200 msec
 }
 else if(x<600 && x>402) //If LEFT button has been pressed
 {
  lcd.clear(); //Clear LCD screen
  lcd.setCursor(1,0); //Set cursor at first column and first line
  lcd.print("USE UP OR DOWN"); //Print message at LCD screen
```

```
  lcd.setCursor(5,1); //Set cursor at first column and first line
  lcd.print("KEYS!!"); //Print message at LCD screen
  delay(2500); //Delay 2.5sec
  Menu(); //Call Menu function
  delay(200); //Delay 200 msec
 }
```

Στο προηγούμενο μέρος του κώδικα προγραμματισμού (Πίνακας 3.4), εμφανίζεται ο τρόπος με τον οποίο ο χρήστης μπορεί να μεταβάλει την επιλογή του στο μενού που του εμφανίζεται στην LCD οθόνη. Για την κατοχύρωση της επιλογής όμως, ο χρήστης πρέπει να πιέσει το πλήκτρο SELECT που βρίσκεται ενσωματωμένο στην πλακέτα της LCD οθόνης. Σε αυτό το μέρος του κώδικα προγραμματισμού (Πίνακας 3.5) αναλύεται το πώς το σύστημα αντιλαμβάνεται το κουμπί SELECT και πως πραγματοποιείται το κλασσικό κούρδισμα (Standard Tune) της κιθάρας.

Όπως αναφέρθηκε προηγουμένως, χρησιμοποιήθηκε η τεχνική (if – else if) για την αναγνώριση των πλήκτρων UP, DOWN, LEFT και RIGHT. Για να αναγνωρίσει το σύστημα και το πλήκτρο SELECT, χρησιμοποιήθηκε άλλη μία συνθήκη (else if). Πιο συγκεκριμένα, για την αναγνώριση του πλήκτρου SELECT από το σύστημα, χρησιμοποιήθηκε η εντολή else if(x < 800 && x != 689) η οποία εκτελείται σε περίπτωση που δεν έχει ικανοποιηθεί καμία από τις προηγούμενες συνθήκες που βρίσκονται στο προηγούμενο μέρος του κώδικα προγραμματισμού (Πίνακας 3.4). Κατά την εκτέλεση της συγκεκριμένης εντολής (else if) ελέγχεται αν η μεταβλητή (x) είναι μικρότερη από οκτακόσια και όχι ίση με εξακόσια ογδόντα εννιά. Αν ικανοποιείται η συγκεκριμένη συνθήκη, τότε εκτελείται το μπλοκ εντολών του πλήκτρου SELECT. Η πρώτη εντολή που εκτελείται στο συγκεκριμένο μπλοκ εντολών είναι η (x = 1023) και αφορά την αποθήκευση της τιμής χίλια είκοσι τρία στην μεταβλητή (x).

Έπειτα, εντός του μπλοκ εντολών του πλήκτρου SELECT, χρησιμοποιείται η τεχνική (if – else if) ώστε να μπορέσει το σύστημα να αναγνωρίσει την επιλογή που έχει κάνει ο χρήστης μέσα από το μενού επιλογής. Συνεπώς, με την τεχνική (if – else if) το σύστημα αναγνωρίζει αν ο χρήστης επέλεξε το κλασσικό κούρδισμα (Standard Tune), ή το κούρδισμα μισού ημιτόνου κάτω (Half-Step Down Tune), ή το κούρδισμα σε «πεσμένη» Ρε (Drop D Tune), ή τον χειροκίνητο έλεγχο του συστήματος (Manual Control). Επομένως, η αμέσως επόμενη εντολή που εκτελείται είναι η if(menu = = 1), η οποία ελέγχει αν η μεταβλητή (menu) είναι ίση με την τιμή ένα. Στην περίπτωση που ικανοποιείται η συνθήκη (if), σημαίνει ότι έχει επιλεχτεί το κλασσικό κούρδισμα (Standard Tune) από τον χρήστη και τότε εκτελείται το αντίστοιχο μπλοκ εντολών.

Στην συνέχεια, οι πρώτες εντολές που εκτελούνται στο μπλοκ εντολών της συνθήκης (if) έχουν να κάνουν με το σβήσιμο χαρακτήρων στην LCD οθόνη μέσω της εντολής lcd.clear(), με την μετακίνηση του κέρσορα σε συγκεκριμένη στήλη και γραμμή μέσω της εντολής lcd.setCursor(), αλλά και με την εκτύπωση μηνυμάτων στην LCD οθόνης μέσω της εντολής lcd.print(“ “). Πιο συγκεκριμένα, πρώτα σβήνονται όλοι οι χαρακτήρες και ύστερα εμφανίζεται το μήνυμα (Standrad Tune Selected) για 1 sec εξαιτίας της εντολής καθυστέρησης delay(1000) που εκτελείται αμέσως μετά. Σκοπός του μηνύματος αυτού είναι να ενημερωθεί ο χρήστης από το σύστημα ότι έχει επιλεχθεί το κλασσικό κούρδισμα (Standard Tune) για τις χορδές της κιθάρας. Έπειτα, σβήνονται ξανά

οι χαρακτήρες από την LCD οθόνη και εμφανίζεται το μήνυμα (HIT E STRING) για 1 sec εξαιτίας της εντολής καθυστέρησης delay(1000) που εκτελείται αμέσως μετά. Το μήνυμα αυτό έχει σκοπό να προτρέψει τον χρήστη να διεγείρει την χορδή Μι μπάσο (E). Ύστερα, σβήνονται οι χαρακτήρες από την LCD οθόνη και εμφανίζεται το μήνυμα (E -> 82.41 Hz) στην πρώτη γραμμή, το οποίο δηλώνει την επιθυμητή τιμή της θεμελιώδης συχνότητας ταλάντωσης της χορδής Μι μπάσο (E). Επίσης, στην δεύτερη γραμμή εμφανίζεται το μήνυμα (Hz Note) σε συγκεκριμένες θέσεις, ώστε κατά την διάρκεια του κουρδίσματος να εκτυπώνεται η πραγματική συχνότητα πριν το (Hz) και η νότα στην οποία αντιστοιχεί η συχνότητα ταλάντωσης της χορδής πριν το (Note). Έπειτα, ακολουθούν κάποιες αρχικοποιήσεις μεταβλητών πριν την έναρξη της διαδικασίας κουρδίσματος της χορδής Μι μπάσο (E). Πιο συγκεκριμένα, ο μετρητής (k) αρχικοποιείτε με την τιμή είκοσι ένα, ενώ στις σημαίες (flags) με ονομασίες (string_6, string_5, string_4, string_3, string_2 & string_1) γίνεται εναπόθεση της τιμής μηδέν.

Μετά την ολοκλήρωση των αρχικοποιήσεων, εφαρμόζεται ο βρόγχος επανάληψης (while()) ο οποίος στο μπλοκ εντολών του περιλαμβάνει τις απαραίτητες εντολές για το κούρδισμα της χορδής Μι μπάσο (E). Για την εφαρμογή του βρόγχου επανάληψης (while), θα πρέπει εντός των παρενθέσεων να τοποθετηθεί μία συνθήκη. Όσο η συνθήκη αυτή ικανοποιείται, τόσο θα συνεχίσει να εκτελείται το μπλοκ εντολών του βρόγχου επανάληψης (while). Στην συγκεκριμένη περίπτωση, εντός των παρενθέσεων βρίσκεται η συνθήκη (string_6 != 1) η οποία σημαίνει ότι όσο η σημαία (flag) με όνομα (string_6) δεν είναι ίση με ένα, τόσο θα συνεχίζει να εκτελείται το μπλοκ εντολών του βρόγχου επανάληψης. Πιο συγκεκριμένα, όσο η σημαία (flag) με όνομα (string_6) είναι μηδέν αυτό σημαίνει ότι η χορδή Μι μπάσο (E) είναι ξεκούρδιστη. Μόλις όμως η σημαία (flag) με όνομα (string_6) γίνει ένα, τότε αυτό σημαίνει ότι η χορδή Μι μπάσο (E) έχει κουρδιστεί και το πρόγραμμα μπορεί να συνεχίσει στο κούρδισμα της επόμενης χορδής. Επειδή όμως η συγκεκριμένη σημαία (flag) έχει αρχικοποιηθεί με την τιμή μηδέν, το μπλοκ εντολών του βρόγχου επανάληψης (while) θα εκτελεστεί μία φορά τουλάχιστον.

Εντός λοιπόν του μπλοκ εντολών του βρόγχου επανάληψης (while), εκτελούνται οι εντολές οι οποίες είναι απαραίτητες για το κούρδισμα της χορδής Μι μπάσο (E). Πιο συγκεκριμένα, υπάρχει μία συνθήκη ελέγχου (if), η οποία ελέγχει την τιμή του μετρητή (k) αν είναι μεγαλύτερη από είκοσι. Ο μετρητής (k) έχει αρχικοποιηθεί πριν τον βρόγχο επανάληψης (while) με την τιμή είκοσι ένα, με αποτέλεσμα να εκτελείται το μπλοκ εντολών της συνθήκης ελέγχου (if). Πιο συγκεκριμένα, στο μπλοκ εντολών της συνθήκης (if) εκτελείτε πρώτα η εντολή pinMode(39, OUTPUT), η οποία καθορίζει τον ψηφιακό ακροδέκτη τριάντα εννιά ως έξοδο. Έπειτα, εκτελείται η εντολή digitalWrite(39, LOW), όπου ο ψηφιακός ακροδέκτης τριάντα εννιά μεταβαίνει σε κατάσταση χαμηλού δυναμικού για 10msec, διότι αμέσως μετά εκτελείται η εντολή χρονικής καθυστέρησης delay(10). Μετά εκτελείται η εντολή pinMode(39, INPUT), η οποία καθορίζει τον ψηφιακό ακροδέκτη τριάντα εννιά ως είσοδο. Ύστερα, το μπλοκ εντολών της συνθήκης ελέγχου (if) κλείνει με την εντολή (k = 0), όπου γίνεται μηδενισμός του μετρητή (k). Σκοπός της συγκεκριμένης συνθήκης ελέγχου, είναι η γείωση (GND) του σήματος που προέρχεται από το μικρόφωνο επαφής ανά τακτά χρονικά διαστήματα, ώστε να μην υπάρχουν παράσιτα τα οποία παραμορφώνουν το σήμα και οδηγούν σε λανθασμένους υπολογισμούς της θεμελιώδης συχνότητας ταλάντωσης.

Στην συνέχεια, μετά την συνθήκη ελέγχου (if), γίνεται κλήση της συνάρτησης FREQ_DETECTION(), η οποία ξεκινά την διαδικασία δειγματοληψίας του σήματος από

το μικρόφωνο επαφής και αναλύεται σε επόμενη ενότητα. Η διαδικασία δειγματοληψίας, διαρκεί για 0.5sec εξαιτίας της εντολής καθυστέρησης delay(500) που εκτελείται αμέσως μετά. Ύστερα, γίνεται κλήση της συνάρτησης FREQ_DETECTION1(), η οποία αναλύεται σε επόμενη ενότητα και έχει σκοπό τόσο τον υπολογισμό της θεμελιώδης συχνότητας ταλάντωσης της χορδής όσο και την εύρεση της νότας στην οποία αντιστοιχεί η συγκεκριμένη συχνότητα. Έπειτα, γίνεται κλήση της συνάρτησης FREQ_DETECTION_STOP() η οποία αναλύεται και αυτή σε επόμενη ενότητα και αποσκοπεί στην παύση της δειγματοληψίας του σήματος από το μικρόφωνο επαφής απενεργοποιώντας τον αναλογικό-ψηφιακό μετατροπέα.

Μετά λοιπόν την ολοκλήρωση της δειγματοληψίας και αφού έχει υπολογιστεί η θεμελιώδης συχνότητα ταλάντωσης της χορδής, εμφανίζεται στην σειριακή οθόνη το μήνυμα (E String) και η τιμή της συχνότητας που έχει υπολογιστεί. Αυτό επιτυγχάνεται με την χρήση της εντολής Serial.println(). Στην συνέχεια, χρησιμοποιούνται οι εντολές lcd.setCursor() και lcd.print(" ") η οποίες έχουν σκοπό την μετακίνηση του κέρσορα στην πρώτη στήλη της δεύτερης γραμμής της LCD οθόνης, το σβήσιμο της παλιάς τιμής της συχνότητας και την εμφάνιση της πραγματικής τιμής της συχνότητας που υπολογίστηκε. Με αυτόν τον τρόπο, ο χρήστης μπορεί να δει στην LCD οθόνη την πραγματική τιμή της συχνότητας ταλάντωσης της χορδής Μι μπάσο (E). Αφότου ολοκληρωθεί η εμφάνιση της συχνότητας στην LCD οθόνη, εκτελείται η εντολή (k++), η οποία έχει σκοπό την αύξηση της τιμής του μετρητή (k) κατά μία μονάδα.

Εν συνεχεία, εφαρμόζεται η τεχνική ελέγχου (if – else if – else), με σκοπό τον έλεγχο της συχνότητας ταλάντωσης της χορδής. Πιο συγκεκριμένα, με την εντολή ελέγχου if (frequency < 82.35 && frequency > 65 && frequency > 0), ελέγχεται αν η τιμή της μεταβλητής (frequency) είναι μικρότερη από 82.35Hz και μεγαλύτερη από 65Hz και μεγαλύτερη από 0Hz. Επομένως, αν ικανοποιείται αυτή η συνθήκη σημαίνει ότι η συχνότητα ταλάντωσης της χορδής Μι μπάσο (E) είναι χαμηλότερη από την επιθυμητή συχνότητα και για να διορθωθεί θα πρέπει να εκτελεστεί το μπλοκ εντολών της συνθήκης ελέγχου (if). Εντός λοιπόν του μπλοκ εντολών, γίνεται μηδενισμός της μεταβλητής (frequency) και ύστερα εκτελείται ένας βρόγχος επανάληψης (for). Ο βρόγχος επανάληψης (for) επαναλαμβάνει το μπλοκ εντολών του είκοσι ένα φορές. Εντός του μπλοκ εντολών του βρόγχου επανάληψης (for) γίνεται κλήση της συνάρτησης (Counterclockwise_EStep_Motor ()). Η συνάρτηση αυτή αναλύεται σε επόμενη ενότητα και έχει σκοπό την αντί-ωρολογιακή περιστροφή του άξονα του βηματικού κινητήρα και κατά συνέπεια την αντί-ωρολογιακή περιστροφή του κλειδιού της χορδής Μι μπάσο (E). Μετά την ολοκλήρωση του βρόγχου επανάληψης (for), εκτελείται η εντολή delay(2) η οποία δημιουργεί μια χρονική καθυστέρηση των 2msec. Το μπλοκ εντολών της συνθήκης ελέγχου (if) ολοκληρώνεται με την κλήση της συνάρτησης (EStep_Motor_Stop ()), η οποία αναλύεται σε επόμενο κεφάλαιο και σκοπός της είναι να απενεργοποιήσει τον βηματικό κινητήρα. Στην περίπτωση όμως που δεν ικανοποιείται η συνθήκη ελέγχου (if), τότε εκτελείται η εντολή ελέγχου else if (frequency > 82.47 && frequency < 103 && frequency > 0), όπου ελέγχεται αν η τιμή της μεταβλητής (frequency) είναι μεγαλύτερη από 82.47Hz και μικρότερη από 103Hz και μεγαλύτερη από 0Hz. Συνεπώς, αν ικανοποιείται αυτή η συνθήκη σημαίνει ότι η συχνότητα ταλάντωσης της χορδής Μι μπάσο (E) είναι υψηλότερη από την επιθυμητή συχνότητα και για να διορθωθεί θα πρέπει να εκτελεστεί το μπλοκ εντολών της συνθήκης ελέγχου (else if). Εντός λοιπόν του μπλοκ εντολών, γίνεται μηδενισμός της μεταβλητής (frequency) και ύστερα εκτελείται ένας

βρόγχος επανάληψης (for). Εντός του μπλοκ εντολών του βρόγχου επανάληψης (for) γίνεται κλήση της συνάρτησης (Clockwise_EStep_Motor ()), η οποία αναλύεται σε επόμενη ενότητα. Σκοπός της συνάρτησης (Clockwise_EStep_Motor ()), είναι να περιστρέψει τον άξονα του βηματικού κινητήρα ωρολογιακά και κατά συνέπεια να περιστρέψει ωρολογιακά και το κλειδί της χορδής Μι μπάσο (E). Αφότου ολοκληρωθεί ο βρόγχος επανάληψης (for), εκτελείται η εντολή delay(2) η οποία δημιουργεί μια χρονική καθυστέρηση των 2msec. Η τελευταία εντολή που εκτελείται στο μπλοκ εντολών της συνθήκης ελέγχου (else if) αφορά την κλήση της συνάρτησης (EStep_Motor_Stop ()), η οποία έχει σκοπό να απενεργοποιήσει τον βηματικό κινητήρα. Αν όμως δεν ικανοποιείται η προηγούμενη συνθήκη (else if), τότε εκτελείται η εντολή ελέγχου else if (frequency >=82.35 && frequency <=82.47), η οποία ελέγχει αν η τιμή της μεταβλητής (frequency) είναι μεγαλύτερη ή ίση από 82.35Hz και μικρότερη ή ίση από 82.47Hz. Επομένως, αν ικανοποιείται αυτή η συνθήκη τότε η πραγματική συχνότητα ταλάντωσης της χορδής Μι μπάσο (E) βρίσκεται εντός του εύρους των επιθυμητών συχνοτήτων της χορδής και εκτελείται το αντίστοιχο μπλοκ εντολών. Εντός λοιπόν του συγκεκριμένου μπλοκ εντολών, αποθηκεύεται στην σημαία (flag) με όνομα (string_6) η τιμή ένα. Με αυτόν τον τρόπο υποδηλώνεται στο πρόγραμμα η ολοκλήρωση του κουρδίσματος της χορδής Μι μπάσο (E). Ύστερα, εκτελείται η εντολή (k = 21), όπου πραγματοποιείται αρχικοποίηση του μετρητή (k) με την τιμή είκοσι ένα και αμέσως μετά, πραγματοποιείται μηδενισμός της μεταβλητής (frequency) μέσω της εντολής (frequency = 0.00). Έπειτα, εκτελούνται οι εντολές lcd.clear(), lcd.setCursor() και lcd.print(“ “) με την κατάλληλη σειρά, ώστε να εμφανιστεί το μήνυμα (Hit A string) στην LCD οθόνη. Το μήνυμα αυτό παραμένει στην LCD οθόνη για 1sec εξαιτίας της εντολής (delay1000) και σκοπό έχει να προτρέψει τον χρήστη να διεγείρει την επόμενη χορδή η οποία είναι η Λα (A). Εν συνεχεία, εκτελούνται ξανά οι εντολές lcd.clear(), lcd.setCursor() και lcd.print(“ “) με την κατάλληλη σειρά ώστε, να εμφανιστεί στην πρώτη γραμμή της LCD οθόνης το μήνυμα (A -> 110 Hz), το οποίο υποδηλώνει την επιθυμητή τιμή της συχνότητας της χορδής Λα (A) και στην δεύτερη γραμμή το μήνυμα (Hz Note) σε συγκεκριμένες θέσης. Σκοπός του μηνύματος της δεύτερης γραμμής είναι να εκτυπώνεται η πραγματική συχνότητα πριν το (Hz) και η νότα στην οποία αντιστοιχεί η συχνότητα ταλάντωσης της χορδής πριν το (Note) κατά την διάρκεια κουρδίσματος της χορδής Λα (A). Αν όμως, δεν ικανοποιείται καμία συνθήκη από τις εντολές ελέγχου (if-else if), τότε εκτελείται το μπλοκ εντολών της συνθήκης (else). Πιο συγκεκριμένα, το μπλοκ εντολών της συνθήκης (else) περιλαμβάνει μία μόνο εντολή η οποία μηδενίζει την τιμή της μεταβλητής (frequency). Με αυτό τον τρόπο, αντιμετωπίζονται οι συχνότητες που μπορούν να προκύψουν από την ταυτόχρονη διέγερση δύο χορδών ή από την διέγερση λανθασμένης χορδής. Σε αυτό το σημείο ολοκληρώνεται το μπλοκ εντολών του βρόγχου επανάληψης (while), το οποίο έχει να κάνει με το κούρδισμα της χορδής Μι μπάσο (E). Η διαδικασία που μόλις περιγράφτηκε, εκτελείτε ξανά μέχρι να πραγματοποιηθεί το κούρδισμα της χορδής Μι μπάσο (E) εντός του εύρος των συχνοτήτων ταλάντωσης που έχει οριστεί για την συγκεκριμένη χορδή.

Με την ολοκλήρωση του κουρδίσματος της χορδής Μι μπάσο (E), σειρά έχει το κούρδισμα της χορδής Λα (A). Αυτό επιτυγχάνεται με την χρήση του βρόγχου επανάληψης while(), ο οποίος στο μπλοκ εντολών του περιλαμβάνει τις απαραίτητες εντολές για το κούρδισμα της χορδής Λα (A). Η συνθήκη που βρίσκεται εντός των παρενθέσεων του βρόγχου επανάληψης (while) είναι η (string_5 != 1) η οποία σημαίνει ότι όσο η σημαία (flag) με όνομα (string_5) δεν είναι ίση με ένα, τόσο θα συνεχίζει να εκτελείται το μπλοκ εντολών του βρόγχου επανάληψης. Πιο συγκεκριμένα, όσο η σημαία

(flag) με όνομα (string_5) είναι μηδέν αυτό σημαίνει ότι η χορδή Λα (A) είναι ξεκούρδιστη. Μόλις όμως η σημαία (flag) με όνομα (string_5) γίνει ένα, τότε αυτό σημαίνει ότι η χορδή Λα (A) έχει κουρδιστεί και το πρόγραμμα μπορεί να συνεχίσει στο κούρδισμα της επόμενης χορδής. Επειδή όμως η συγκεκριμένη σημαία (flag) έχει αρχικοποιηθεί με την τιμή μηδέν, το μπλοκ εντολών του βρόγχου επανάληψης (while) θα εκτελεστεί μία φορά τουλάχιστον.

Ουσιαστικά, εντός του μπλοκ εντολών του βρόγχου επανάληψης (while), εκτελούνται οι εντολές οι οποίες είναι απαραίτητες για το κούρδισμα της χορδής Λα (A). Όπως αναφέρθηκε και προηγουμένως, υπάρχει μία συνθήκη ελέγχου (if), η οποία ελέγχει την τιμή του μετρητή (k) αν είναι μεγαλύτερη από είκοσι. Ο μετρητής (k) έχει αρχικοποιηθεί στον προηγούμενο βρόγχο επανάληψης (while) με την τιμή είκοσι ένα, με αποτέλεσμα να εκτελείται το μπλοκ εντολών της συνθήκης ελέγχου (if). Ουσιαστικά, στο μπλοκ εντολών της συνθήκης (if) εκτελείτε πρώτα η εντολή pinMode(39, OUTPUT), η οποία καθορίζει τον ψηφιακό ακροδέκτη τριάντα εννιά ως έξοδο. Έπειτα, εκτελείται η εντολή digitalWrite(39, LOW), όπου ο ψηφιακός ακροδέκτης τριάντα εννιά μεταβαίνει σε κατάσταση χαμηλού δυναμικού για 10msec, διότι αμέσως μετά εκτελείται η εντολή χρονικής καθυστέρησης delay(10). Ύστερα, εκτελείται η εντολή pinMode(39, INPUT), η οποία καθορίζει τον ψηφιακό ακροδέκτη τριάντα εννιά ως είσοδο. Το μπλοκ εντολών της συνθήκης ελέγχου (if) ολοκληρώνεται με την εντολή (k = 0), όπου γίνεται μηδενισμός του μετρητή (k). Σκοπός της συγκεκριμένης συνθήκης ελέγχου είναι η γείωση (GND) του σήματος που προέρχεται από το μικρόφωνο επαφής ανά τακτά χρονικά διαστήματα, ώστε να μην υπάρχουν παράσιτα τα οποία παραμορφώνουν το σήμα και οδηγούν σε λανθασμένους υπολογισμούς της συχνότητας ταλάντωσης της χορδής.

Αφότου ολοκληρωθεί η εντολή της συνθήκης ελέγχου (if), γίνεται κλήση της συνάρτησης FREQ_DETECTION(), η οποία αναλύεται σε επόμενη ενότητα και αποσκοπεί στην εκκίνηση της διαδικασίας της δειγματοληψίας του σήματος από το μικρόφωνο επαφής. Η διαδικασία δειγματοληψίας, διαρκεί για 0.5sec εξαιτίας της εντολής καθυστέρησης delay(500) που εκτελείται αμέσως μετά. Ύστερα, γίνεται κλήση της συνάρτησης FREQ_DETECTION1(), η οποία έχει σκοπό τόσο τον υπολογισμό της θεμελιώδης συχνότητας ταλάντωσης της χορδής όσο και την εύρεση της νότας στην οποία αντιστοιχεί η συγκεκριμένη συχνότητα και η ανάλυσή της γίνεται σε επόμενη ενότητα. Έπειτα, γίνεται κλήση της συνάρτησης FREQ_DETECTION_STOP() η οποία αναλύεται και αυτή σε επόμενη ενότητα και αποσκοπεί στην παύση της δειγματοληψίας του σήματος από το μικρόφωνο επαφής απενεργοποιώντας τον αναλογικό-ψηφιακό μετατροπέα.

Εν συνεχεία, αφού έχει υπολογιστεί η θεμελιώδης συχνότητα ταλάντωσης της χορδής, εμφανίζεται στην σειριακή οθόνη το μήνυμα (A String) και η τιμή της συχνότητας που έχει υπολογιστεί. Αυτό επιτυγχάνεται με την χρήση της εντολής Serial.println(). Στην συνέχεια, χρησιμοποιούνται οι εντολές lcd.setCursor() και lcd.print(" ") η οποίες έχουν σκοπό την μετακίνηση του κέρσορα στην πρώτη στήλη της δεύτερης γραμμής της LCD οθόνης, το σβήσιμο της παλιάς τιμής της συχνότητας και την εμφάνιση της πραγματικής τιμής της συχνότητας που υπολογίστηκε. Με αυτόν τον τρόπο, ο χρήστης μπορεί να δει στην LCD οθόνη την πραγματική τιμή της συχνότητας ταλάντωσης της χορδής Λα (A). Αφότου ολοκληρωθεί η εμφάνιση της συχνότητας στην LCD οθόνη, εκτελείται η εντολή (k++), η οποία έχει σκοπό την αύξηση της τιμής του μετρητή (k) κατά μία μονάδα.

Έπειτα, εφαρμόζεται η τεχνική ελέγχου (if – else if – else), η οποία αποσκοπεί στον έλεγχο της συχνότητας ταλάντωσης της χορδής. Ουσιαστικά, με την εκτέλεση της εντολής ελέγχου if (frequency < 109.80 && frequency > 92 && frequency > 0), ελέγχεται αν η τιμή της μεταβλητής (frequency) είναι μικρότερη από 109.8Hz και μεγαλύτερη από 92Hz και μεγαλύτερη από 0Hz. Συνεπώς, αν ικανοποιείται αυτή η συνθήκη σημαίνει ότι η συχνότητα ταλάντωσης της χορδής Λα (A) είναι χαμηλότερη από την επιθυμητή συχνότητα και για να διορθωθεί θα πρέπει να εκτελεστεί το μπλοκ εντολών της συνθήκης ελέγχου (if). Εντός λοιπόν του μπλοκ εντολών, γίνεται μηδενισμός της μεταβλητής (frequency) και ύστερα εκτελείται ένας βρόγχος επανάληψης (for) ο οποίος επαναλαμβάνει το μπλοκ εντολών του είκοσι ένα φορές. Όσον αφορά το μπλοκ εντολών του βρόγχου επανάληψης (for), γίνεται κλήση της συνάρτησης (Counterclockwise_AStep_Motor ()). Η συνάρτηση αυτή αναλύεται σε επόμενη ενότητα και έχει σκοπό την αντί-ωρολογιακή περιστροφή του άξονα του βηματικού κινητήρα και κατά συνέπεια την αντί-ωρολογιακή περιστροφή του κλειδιού της χορδής Λα (A). Μετά την ολοκλήρωση του βρόγχου επανάληψης (for), εκτελείται η εντολή delay(2) η οποία δημιουργεί μια χρονική καθυστέρηση των 2msec. Το μπλοκ εντολών της συνθήκης ελέγχου (if) ολοκληρώνεται με την κλήση της συνάρτησης (AStep_Motor_Stop ()), η οποία αναλύεται σε επόμενο κεφάλαιο και σκοπός της είναι να απενεργοποιήσει τον βηματικό κινητήρα. Αν όμως δεν ικανοποιείται η συνθήκη ελέγχου (if), τότε εκτελείται η εντολή ελέγχου else if (frequency > 110.21 && frequency < 130 && frequency > 0), όπου ελέγχεται αν η τιμή της μεταβλητής (frequency) είναι μεγαλύτερη από 110.21Hz και μικρότερη από 130Hz και μεγαλύτερη από 0Hz. Συνεπώς, αν ικανοποιείται αυτή η συνθήκη σημαίνει ότι η συχνότητα ταλάντωσης της χορδής Λα (A) είναι υψηλότερη από την επιθυμητή συχνότητα και για να διορθωθεί θα πρέπει να εκτελεστεί το μπλοκ εντολών της συνθήκης ελέγχου (else if). Ουσιαστικά, εντός του μπλοκ εντολών γίνεται μηδενισμός της μεταβλητής (frequency) και ύστερα εκτελείται ένας βρόγχος επανάληψης (for). Εντός του μπλοκ εντολών του βρόγχου επανάληψης (for) γίνεται κλήση της συνάρτησης (Clockwise_AStep_Motor ()), της οποίας η ανάλυση πραγματοποιείται σε επόμενη ενότητα. Η κλήση της συνάρτησης (Clockwise_AStep_Motor ()), αποσκοπεί στην ωρολογιακή περιστροφή του άξονα του βηματικού κινητήρα και κατά συνέπεια στην ωρολογιακή περιστροφή του κλειδιού της χορδής Λα (A). Μετά την ολοκλήρωση του βρόγχου επανάληψης (for), εκτελείται η εντολή delay(2) η οποία δημιουργεί μια χρονική καθυστέρηση των 2msec. Η τελευταία εντολή που εκτελείται στο μπλοκ εντολών της συνθήκης ελέγχου (else if) αφορά την κλήση της συνάρτησης (AStep_Motor_Stop ()), η οποία αποσκοπεί στην απενεργοποίηση του βηματικού κινητήρα. Σε περίπτωση όμως που δεν ικανοποιείται η προηγούμενη συνθήκη (else if), τότε εκτελείται η εντολή ελέγχου else if (frequency >=109.80 && frequency <=110.21), η οποία ελέγχει αν η τιμή της μεταβλητής (frequency) είναι μεγαλύτερη ή ίση από 109.80Hz και μικρότερη ή ίση από 110.21Hz. Συνεπώς, αν ικανοποιείται αυτή η συνθήκη τότε η πραγματική συχνότητα ταλάντωσης της χορδής Λα (A) βρίσκεται εντός του εύρους των επιθυμητών συχνοτήτων της χορδής και εκτελείται το αντίστοιχο μπλοκ εντολών. Πιο συγκεκριμένα, εντός του συγκεκριμένου μπλοκ εντολών, αποθηκεύεται στην σημαία (flag) με όνομα (string_5) η τιμή ένα. Με αυτόν τον τρόπο υποδηλώνεται στο πρόγραμμα η ολοκλήρωση του κουρδίσματος της χορδής Λα (A). Έπειτα, εκτελείται η εντολή (k = 21), όπου πραγματοποιείται αρχικοποίηση του μετρητή (k) με την τιμή είκοσι ένα και αμέσως μετά, εκτελείται η εντολή (frequency = 0.00) η οποία μηδενίζει την τιμή της μεταβλητής (frequency). Ύστερα, πραγματοποιείται εκτέλεση των εντολών lcd.clear(), lcd.setCursor()

και lcd.print(“ “) με την κατάλληλη σειρά, ώστε να εμφανιστεί το μήνυμα (Hit D string) στην LCD οθόνη. Το μήνυμα αυτό παραμένει στην LCD οθόνη για 1sec εξαιτίας της εντολής (delay1000) και έχει σκοπό να προτρέψει τον χρήστη να διεγείρει την επόμενη χορδή η οποία είναι η Ρε (D). Στην συνεχεία, εκτελούνται ξανά οι εντολές lcd.clear(), lcd.setCursor() και lcd.print(“ “) με την κατάλληλη σειρά ώστε, να εμφανιστεί στην πρώτη γραμμή της LCD οθόνης το μήνυμα (D -> 146.83 Hz), το οποίο υποδηλώνει την επιθυμητή τιμή της συχνότητας της χορδής Ρε (D) και στην δεύτερη γραμμή το μήνυμα (Hz Note) σε συγκεκριμένες θέσης. Πιο συγκεκριμένα, το μήνυμα της δεύτερης γραμμής αποσκοπεί στην εκτύπωση της πραγματική συχνότητα πριν το (Hz) και η νότα στην οποία αντιστοιχεί η συχνότητα ταλάντωσης της χορδής πριν το (Note) κατά την διάρκεια κουρδίσματος της χορδής Ρε (D). Στην περίπτωση όμως που δεν ικανοποιείται καμία συνθήκη από τις εντολές ελέγχου (if-else if), τότε εκτελείται το μπλοκ εντολών της συνθήκης (else). Ουσιαστικά, το μπλοκ εντολών της συνθήκης (else) αποτελείται από μία μόνο εντολή η οποία μηδενίζει την τιμή της μεταβλητής (frequency). Με αυτό τον τρόπο, αντιμετωπίζονται οι συχνότητες που μπορούν να προκύψουν από την ταυτόχρονη διέγερση δύο χορδών ή από την διέγερση λανθασμένης χορδής. Σε αυτό το σημείο ολοκληρώνεται το μπλοκ εντολών του βρόγχου επανάληψης (while), το οποίο έχει να κάνει με το κούρδισμα της χορδής Λα (A). Η διαδικασία που μόλις περιγράφτηκε, επαναλαμβάνεται μέχρι να επιτευχθεί το κούρδισμα της χορδής Λα (A) εντός του εύρος των συχνοτήτων ταλάντωσης που έχει οριστεί για την συγκεκριμένη χορδή.

Αφού ολοκληρωθεί το κούρδισμα της χορδής Λα (A), το πρόγραμμα συνεχίζει στην πραγματοποίηση του κουρδίσματος της χορδής Ρε (D). Αυτό επιτυγχάνεται με την χρήση του βρόγχου επανάληψης while(), ο οποίος στο μπλοκ εντολών του περιλαμβάνει τις απαραίτητες εντολές για το κούρδισμα της χορδής Ρε (D). Επιπλέον, η συνθήκη που βρίσκεται εντός των παρενθέσεων του βρόγχου επανάληψης (while) είναι η (string_4 != 1), η οποία δηλώνει ότι όσο η σημαία (flag) με όνομα (string_4) δεν είναι ίση με ένα, τόσο θα συνεχίζει να εκτελείται το μπλοκ εντολών του βρόγχου επανάληψης. Πιο συγκεκριμένα, όσο η σημαία (flag) με όνομα (string_4) είναι μηδέν αυτό σημαίνει ότι η χορδή Ρε (D) είναι ξεκούρδιστη. Μόλις όμως, η σημαία (flag) με όνομα (string_4) γίνει ένα, τότε αυτό σημαίνει ότι η χορδή Ρε (D) έχει κουρδιστεί και το πρόγραμμα μπορεί να συνεχίσει στο κούρδισμα της επόμενης χορδής. Επειδή όμως η συγκεκριμένη σημαία (flag) έχει αρχικοποιηθεί με την τιμή μηδέν, το μπλοκ εντολών του βρόγχου επανάληψης (while) θα εκτελεστεί μία φορά τουλάχιστον.

Εντός λοιπόν του μπλοκ εντολών του βρόγχου επανάληψης (while), εκτελούνται οι εντολές οι οποίες είναι απαραίτητες για το κούρδισμα της χορδής Ρε (D). Πιο συγκεκριμένα, υπάρχει μία συνθήκη ελέγχου (if), η οποία ελέγχει την τιμή του μετρητή (k) αν είναι μεγαλύτερη από είκοσι. Ο μετρητής (k) έχει αρχικοποιηθεί στον προηγούμενο βρόγχο επανάληψης (while) με την τιμή είκοσι ένα, με αποτέλεσμα να εκτελείται το μπλοκ εντολών της συνθήκης ελέγχου (if). Πιο συγκεκριμένα, εντός του μπλοκ εντολών της συνθήκης (if), εκτελείτε πρώτα η εντολή pinMode(39, OUTPUT), η οποία καθορίζει τον ψηφιακό ακροδέκτη τριάντα εννιά ως έξοδο. Έπειτα, εκτελείται η εντολή digitalWrite(39, LOW), όπου ο ψηφιακός ακροδέκτης τριάντα εννιά μεταβαίνει σε κατάσταση χαμηλού δυναμικού για 10msec, διότι αμέσως μετά εκτελείται η εντολή χρονικής καθυστέρησης delay(10). Μετά εκτελείται η εντολή pinMode(39, INPUT), η οποία καθορίζει τον ψηφιακό ακροδέκτη τριάντα εννιά ως είσοδο. Ύστερα, το μπλοκ εντολών της συνθήκης ελέγχου (if) κλείνει με την εντολή (k = 0), όπου γίνεται μηδενισμός

του μετρητή (k). Σκοπός της συγκεκριμένης συνθήκης ελέγχου, είναι η γείωση (GND) του σήματος που προέρχεται από το μικρόφωνο επαφής ανά τακτά χρονικά διαστήματα, ώστε να μην υπάρχουν παράσιτα τα οποία παραμορφώνουν το σήμα και οδηγούν σε λανθασμένους υπολογισμούς της θεμελιώδης συχνότητας ταλάντωσης.

Στην συνέχεια, μετά την συνθήκη ελέγχου (if), γίνεται κλήση της συνάρτησης FREQ_DETECTION(), η οποία ξεκινά την διαδικασία δειγματοληψίας του σήματος από το μικρόφωνο επαφής και αναλύεται σε επόμενη ενότητα. Η διαδικασία δειγματοληψίας, διαρκεί για 0.5sec εξαιτίας της εντολής καθυστέρησης delay(500) που εκτελείται αμέσως μετά. Ύστερα, γίνεται κλήση της συνάρτησης FREQ_DETECTION1(), η οποία αναλύεται σε επόμενη ενότητα και έχει σκοπό τόσο τον υπολογισμό της θεμελιώδης συχνότητας ταλάντωσης της χορδής όσο και την εύρεση της νότας στην οποία αντιστοιχεί η συγκεκριμένη συχνότητα. Έπειτα, γίνεται κλήση της συνάρτησης FREQ_DETECTION_STOP() η οποία αναλύεται και αυτή σε επόμενη ενότητα και αποσκοπεί στην παύση της δειγματοληψίας του σήματος από το μικρόφωνο επαφής απενεργοποιώντας τον αναλογικό-ψηφιακό μετατροπέα.

Μετά λοιπόν την ολοκλήρωση της δειγματοληψίας και αφού έχει υπολογιστεί η θεμελιώδης συχνότητα ταλάντωσης της χορδής, εμφανίζεται στην σειριακή οθόνη το μήνυμα (D String) και η τιμή της συχνότητας που έχει υπολογιστεί. Αυτό επιτυγχάνεται με την χρήση της εντολής Serial.println(). Στην συνέχεια, χρησιμοποιούνται οι εντολές lcd.setCursor() και lcd.print(" ") η οποίες έχουν σκοπό την μετακίνηση του κέρσορα στην πρώτη στήλη της δεύτερης γραμμής της LCD οθόνης, το σβήσιμο της παλιάς τιμής της συχνότητας και την εμφάνιση της πραγματικής τιμής της θεμελιώδης συχνότητας ταλάντωσης της χορδής που υπολογίστηκε. Με αυτόν τον τρόπο, ο χρήστης μπορεί να δει στην LCD οθόνη την πραγματική τιμή της συχνότητας ταλάντωσης της χορδής Ρε (D). Αφότου ολοκληρωθεί η εμφάνιση της συχνότητας στην LCD οθόνη, εκτελείται η εντολή (k++), η οποία έχει σκοπό την αύξηση της τιμής του μετρητή (k) κατά μία μονάδα.

Εν συνεχεία, εφαρμόζεται η τεχνική ελέγχου (if – else if – else), με σκοπό τον έλεγχο της συχνότητας ταλάντωσης της χορδής. Πιο συγκεκριμένα, με την εντολή ελέγχου if (frequency < 146.73 && frequency > 123 && frequency > 0), ελέγχεται αν η τιμή της μεταβλητής (frequency) είναι μικρότερη από 146.73Hz και μεγαλύτερη από 123Hz και μεγαλύτερη από 0Hz. Επομένως, αν ικανοποιείται αυτή η συνθήκη σημαίνει ότι η συχνότητα ταλάντωσης της χορδής Ρε (D) είναι χαμηλότερη από την επιθυμητή συχνότητα και για να διορθωθεί θα πρέπει να εκτελεστεί το μπλοκ εντολών της συνθήκης ελέγχου (if). Εντός λοιπόν του μπλοκ εντολών, γίνεται μηδενισμός της μεταβλητής (frequency) και ύστερα εκτελείται ένας βρόγχος επανάληψης (for). Ο βρόγχος επανάληψης (for) επαναλαμβάνει το μπλοκ εντολών του σαράντα ένα φορές. Εντός του μπλοκ εντολών του βρόγχου επανάληψης (for) γίνεται κλήση της συνάρτησης (Counterclockwise_DStep_Motor ()). Η συνάρτηση αυτή αναλύεται σε επόμενη ενότητα και έχει σκοπό την αντί-ωρολογιακή περιστροφή του άξονα του βηματικού κινητήρα και κατά συνέπεια την αντί-ωρολογιακή περιστροφή του κλειδιού της χορδής Ρε (D). Μετά την ολοκλήρωση του βρόγχου επανάληψης (for), εκτελείται η εντολή delay(2) η οποία δημιουργεί μια χρονική καθυστέρηση των 2msec. Το μπλοκ εντολών της συνθήκης ελέγχου (if) ολοκληρώνεται με την κλήση της συνάρτησης (DStep_Motor_Stop ()), η οποία αναλύεται σε επόμενο κεφάλαιο και σκοπός της είναι να απενεργοποιήσει τον βηματικό κινητήρα. Στην περίπτωση όμως, που δεν ικανοποιείται η συνθήκη ελέγχου (if), τότε εκτελείται η εντολή ελέγχου else if (frequency > 146.93 && frequency < 174 &&

frequency > 0), όπου ελέγχεται αν η τιμή της μεταβλητής (frequency) είναι μεγαλύτερη από 146.93Hz και μικρότερη από 174Hz και μεγαλύτερη από 0Hz. Συνεπώς, αν ικανοποιείται αυτή η συνθήκη σημαίνει ότι η συχνότητα ταλάντωσης της χορδής Ρε (D) είναι υψηλότερη από την επιθυμητή συχνότητα και για να διορθωθεί θα πρέπει να εκτελεστεί το μπλοκ εντολών της συνθήκης ελέγχου (else if). Εντός λοιπόν του μπλοκ εντολών, γίνεται μηδενισμός της μεταβλητής (frequency) και ύστερα εκτελείται ένας βρόγχος επανάληψης (for). Εντός του μπλοκ εντολών του βρόγχου επανάληψης (for) γίνεται κλήση της συνάρτησης (Clockwise_DStep_Motor ()), η οποία αναλύεται σε επόμενη ενότητα. Σκοπός της συνάρτησης (Clockwise_DStep_Motor ()), είναι να περιστρέψει τον άξονα του βηματικού κινητήρα ωρολογιακά και κατά συνέπεια να περιστρέψει ωρολογιακά και το κλειδί της χορδής Ρε (D). Αφότου ολοκληρωθεί ο βρόγχος επανάληψης (for), εκτελείται η εντολή delay(2) η οποία δημιουργεί μια χρονική καθυστέρηση των 2msec. Η τελευταία εντολή που εκτελείται στο μπλοκ εντολών της συνθήκης ελέγχου (else if) αφορά την κλήση της συνάρτησης (DStep_Motor_Stop ()), η οποία έχει σκοπό να απενεργοποιήσει τον βηματικό κινητήρα. Αν όμως δεν ικανοποιείται η προηγούμενη συνθήκη (else if), τότε εκτελείται η εντολή ελέγχου else if (frequency >=146.73 && frequency <=146.93), η οποία ελέγχει αν η τιμή της μεταβλητής (frequency) είναι μεγαλύτερη ή ίση από 146.73Hz και μικρότερη ή ίση από 146.93Hz. Επομένως, αν ικανοποιείται αυτή η συνθήκη τότε η πραγματική συχνότητα ταλάντωσης της χορδής Ρε (D) βρίσκεται εντός του εύρους των επιθυμητών συχνοτήτων της χορδής και εκτελείται το αντίστοιχο μπλοκ εντολών. Εντός λοιπόν του συγκεκριμένου μπλοκ εντολών, αποθηκεύεται στην σημαία (flag) με όνομα (string_4) η τιμή ένα. Με αυτόν τον τρόπο υποδηλώνεται στο πρόγραμμα η ολοκλήρωση του κουρδίσματος της χορδής Ρε (D). Ύστερα, εκτελείται η εντολή (k = 21), όπου πραγματοποιείται αρχικοποίηση του μετρητή (k) με την τιμή είκοσι ένα και αμέσως μετά, πραγματοποιείται μηδενισμός της μεταβλητής (frequency) μέσω της εντολής (frequency = 0.00). Έπειτα, εκτελούνται οι εντολές lcd.clear(), lcd.setCursor() και lcd.print(“ “) με την κατάλληλη σειρά, ώστε να εμφανιστεί το μήνυμα (Hit G string) στην LCD οθόνη. Το μήνυμα αυτό παραμένει στην LCD οθόνη για 1sec εξαιτίας της εντολής (delay1000) και σκοπό έχει να προτρέψει τον χρήστη να διεγείρει την επόμενη χορδή η οποία είναι η Σολ (G). Εν συνεχεία, εκτελούνται ξανά οι εντολές lcd.clear(), lcd.setCursor() και lcd.print(“ “) με την κατάλληλη σειρά ώστε, να εμφανιστεί στην πρώτη γραμμή της LCD οθόνης το μήνυμα (G -> 196 Hz), το οποίο υποδηλώνει την επιθυμητή τιμή της συχνότητας της χορδής Σολ (G) και στην δεύτερη γραμμή το μήνυμα (Hz Note) σε συγκεκριμένες θέσης. Σκοπός του μηνύματος της δεύτερης γραμμής είναι να εκτυπώνεται η πραγματική συχνότητα πριν το (Hz) και η νότα στην οποία αντιστοιχεί η συχνότητα ταλάντωσης της χορδής πριν το (Note) κατά την διάρκεια κουρδίσματος της χορδής Σολ (G). Αν όμως, δεν ικανοποιείται καμία συνθήκη από τις εντολές ελέγχου (if-else if), τότε εκτελείται το μπλοκ εντολών της συνθήκης (else). Πιο συγκεκριμένα, το μπλοκ εντολών της συνθήκης (else) περιλαμβάνει μία μόνο εντολή η οποία μηδενίζει την τιμή της μεταβλητής (frequency). Με αυτό τον τρόπο, αντιμετωπίζονται οι συχνότητες που μπορούν να προκύψουν από την ταυτόχρονη διέγερση δύο χορδών ή από την διέγερση λανθασμένης χορδής. Σε αυτό το σημείο ολοκληρώνεται το μπλοκ εντολών του βρόγχου επανάληψης (while), το οποίο έχει να κάνει με το κούρδισμα της χορδής Ρε (D). Η διαδικασία που μόλις περιγράφτηκε, εκτελείτε ξανά μέχρι να πραγματοποιηθεί το κούρδισμα της χορδής Ρε (D) εντός του εύρος των συχνοτήτων ταλάντωσης που έχει οριστεί για την συγκεκριμένη χορδή.

Με την ολοκλήρωση του κουρδίσματος της χορδής Ρε (D), σειρά έχει το κούρδισμα της χορδής Σολ (G). Αυτό επιτυγχάνεται με την χρήση του βρόγχου επανάληψης while(), ο οποίος στο μπλοκ εντολών του περιλαμβάνει τις απαραίτητες εντολές για το κούρδισμα της χορδής Σολ (G). Η συνθήκη που βρίσκεται εντός των παρενθέσεων του βρόγχου επανάληψης (while) είναι η (string_3 != 1) η οποία σημαίνει ότι όσο η σημαία (flag) με όνομα (string_3) δεν είναι ίση με ένα, τόσο θα συνεχίζει να εκτελείται το μπλοκ εντολών του βρόγχου επανάληψης. Πιο συγκεκριμένα, όσο η σημαία (flag) με όνομα (string_3) είναι μηδέν αυτό σημαίνει ότι η χορδή Σολ (G) είναι ξεκούρδιστη. Μόλις όμως η σημαία (flag) με όνομα (string_3) γίνει ένα, τότε αυτό σημαίνει ότι η χορδή Σολ (G) έχει κουρδιστεί και το πρόγραμμα μπορεί να συνεχίσει στο κούρδισμα της επόμενης χορδής. Επειδή όμως η συγκεκριμένη σημαία (flag) έχει αρχικοποιηθεί με την τιμή μηδέν, το μπλοκ εντολών του βρόγχου επανάληψης (while) θα εκτελεστεί μία φορά τουλάχιστον.

Ουσιαστικά, εντός του μπλοκ εντολών του βρόγχου επανάληψης (while), εκτελούνται οι εντολές οι οποίες είναι απαραίτητες για το κούρδισμα της χορδής Σολ (G). Όπως αναφέρθηκε και προηγουμένως, υπάρχει μία συνθήκη ελέγχου (if), η οποία ελέγχει την τιμή του μετρητή (k) αν είναι μεγαλύτερη από είκοσι. Ο μετρητής (k) έχει αρχικοποιηθεί στον προηγούμενο βρόγχο επανάληψης (while) με την τιμή είκοσι ένα, με αποτέλεσμα να εκτελείται το μπλοκ εντολών της συνθήκης ελέγχου (if). Ουσιαστικά, στο μπλοκ εντολών της συνθήκης (if) εκτελείτε πρώτα η εντολή pinMode(39, OUTPUT), η οποία καθορίζει τον ψηφιακό ακροδέκτη τριάντα εννιά ως έξοδο. Έπειτα, εκτελείται η εντολή digitalWrite(39, LOW), όπου ο ψηφιακός ακροδέκτης τριάντα εννιά μεταβαίνει σε κατάσταση χαμηλού δυναμικού για 10msec, διότι αμέσως μετά εκτελείται η εντολή χρονικής καθυστέρησης delay(10). Ύστερα, εκτελείται η εντολή pinMode(39, INPUT), η οποία καθορίζει τον ψηφιακό ακροδέκτη τριάντα εννιά ως είσοδο. Το μπλοκ εντολών της συνθήκης ελέγχου (if) ολοκληρώνεται με την εντολή (k = 0), όπου γίνεται μηδενισμός του μετρητή (k). Σκοπός της συγκεκριμένης συνθήκης ελέγχου είναι η γείωση (GND) του σήματος που προέρχεται από το μικρόφωνο επαφής ανά τακτά χρονικά διαστήματα, ώστε να μην υπάρχουν παράσιτα τα οποία παραμορφώνουν το σήμα και οδηγούν σε λανθασμένους υπολογισμούς της συχνότητας ταλάντωσης της χορδής..

Αφότου ολοκληρωθεί η εντολή της συνθήκης ελέγχου (if), γίνεται κλήση της συνάρτησης FREQ_DETECTION(), η οποία αναλύεται σε επόμενη ενότητα και αποσκοπεί στην εκκίνηση της διαδικασίας της δειγματοληψίας του σήματος από το μικρόφωνο επαφής. Η διαδικασία δειγματοληψίας, διαρκεί για 0.5sec εξαιτίας της εντολής καθυστέρησης delay(500) που εκτελείται αμέσως μετά. Ύστερα, γίνεται κλήση της συνάρτησης FREQ_DETECTION1(), η οποία έχει σκοπό τόσο τον υπολογισμό της θεμελιώδης συχνότητας ταλάντωσης της χορδής όσο και την εύρεση της νότας στην οποία αντιστοιχεί η συγκεκριμένη συχνότητα και η ανάλυσή της γίνεται σε επόμενη ενότητα. Έπειτα, γίνεται κλήση της συνάρτησης FREQ_DETECTION_STOP() η οποία αναλύεται και αυτή σε επόμενη ενότητα και αποσκοπεί στην παύση της δειγματοληψίας του σήματος από το μικρόφωνο επαφής απενεργοποιώντας τον αναλογικό-ψηφιακό μετατροπέα.

Εν συνεχεία, αφού έχει υπολογιστεί η θεμελιώδης συχνότητα ταλάντωσης της χορδής, εμφανίζεται στην σειριακή οθόνη το μήνυμα (G String) και η τιμή της συχνότητας που έχει υπολογιστεί. Αυτό επιτυγχάνεται με την χρήση της εντολής Serial.println(). Στην συνέχεια, χρησιμοποιούνται οι εντολές lcd.setCursor() και lcd.print(“ “) η οποίες έχουν σκοπό την μετακίνηση του κέρσορα στην πρώτη στήλη της δεύτερης γραμμής της LCD οθόνης, το σβήσιμο της παλιάς τιμής της συχνότητας και την εμφάνιση

της πραγματικής τιμής της συχνότητας που υπολογίστηκε. Με αυτόν τον τρόπο, ο χρήστης μπορεί να δει στην LCD οθόνη την πραγματική τιμή της συχνότητας ταλάντωσης της χορδής Σολ (G). Αφότου ολοκληρωθεί η εμφάνιση της συχνότητας στην LCD οθόνη, εκτελείται η εντολή (k++), η οποία έχει σκοπό την αύξηση της τιμής του μετρητή (k) κατά μία μονάδα.

Έπειτα, εφαρμόζεται η τεχνική ελέγχου (if – else if – else), η οποία αποσκοπεί στον έλεγχο της συχνότητας ταλάντωσης της χορδής. Ουσιαστικά, με την εκτέλεση της εντολής ελέγχου if (frequency < 195.90 && frequency > 164 && frequency > 0), ελέγχεται αν η τιμή της μεταβλητής (frequency) είναι μικρότερη από 195.90Hz και μεγαλύτερη από 164Hz και μεγαλύτερη από 0Hz. Συνεπώς, αν ικανοποιείται αυτή η συνθήκη σημαίνει ότι η συχνότητα ταλάντωσης της χορδής Σολ (G) είναι χαμηλότερη από την επιθυμητή συχνότητα και για να διορθωθεί θα πρέπει να εκτελεστεί το μπλοκ εντολών της συνθήκης ελέγχου (if). Εντός λοιπόν του μπλοκ εντολών, γίνεται μηδενισμός της μεταβλητής (frequency) και ύστερα εκτελείται ένας βρόγχος επανάληψης (for) ο οποίος επαναλαμβάνει το μπλοκ εντολών του δέκα έξι φορές. Όσον αφορά το μπλοκ εντολών του βρόγχου επανάληψης (for), γίνεται κλήση της συνάρτησης (Counterclockwise_GStep_Motor ()). Η συνάρτηση αυτή αναλύεται σε επόμενη ενότητα και έχει σκοπό την αντί-ωρολογιακή περιστροφή του άξονα του βηματικού κινητήρα και κατά συνέπεια την αντί-ωρολογιακή περιστροφή του κλειδιού της χορδής Σολ (G). Μετά την ολοκλήρωση του βρόγχου επανάληψης (for), εκτελείται η εντολή delay(2) η οποία δημιουργεί μια χρονική καθυστέρηση των 2msec. Το μπλοκ εντολών της συνθήκης ελέγχου (if) ολοκληρώνεται με την κλήση της συνάρτησης (GStep_Motor_Stop ()), η οποία αναλύεται σε επόμενο κεφάλαιο και σκοπός της είναι να απενεργοποιήσει τον βηματικό κινητήρα. Αν όμως δεν ικανοποιείται η συνθήκη ελέγχου (if), τότε εκτελείται η εντολή ελέγχου else if (frequency > 196.25 && frequency < 233 && frequency > 0), όπου ελέγχεται αν η τιμή της μεταβλητής (frequency) είναι μεγαλύτερη από 196.25Hz και μικρότερη από 233Hz και μεγαλύτερη από 0Hz. Συνεπώς, αν ικανοποιείται αυτή η συνθήκη σημαίνει ότι η συχνότητα ταλάντωσης της χορδής Σολ (G) είναι υψηλότερη από την επιθυμητή συχνότητα και για να διορθωθεί θα πρέπει να εκτελεστεί το μπλοκ εντολών της συνθήκης ελέγχου (else if). Ουσιαστικά, εντός του μπλοκ εντολών γίνεται μηδενισμός της μεταβλητής (frequency) και ύστερα εκτελείται ένας βρόγχος επανάληψης (for). Εντός του μπλοκ εντολών του βρόγχου επανάληψης (for) γίνεται κλήση της συνάρτησης (Clockwise_GStep_Motor ()), της οποίας η ανάλυση πραγματοποιείται σε επόμενη ενότητα. Η κλήση της συνάρτησης (Clockwise_GStep_Motor ()), αποσκοπεί στην ωρολογιακή περιστροφή του άξονα του βηματικού κινητήρα και κατά συνέπεια στην ωρολογιακή περιστροφή του κλειδιού της χορδής Σολ (G). Μετά την ολοκλήρωση του βρόγχου επανάληψης (for), εκτελείται η εντολή delay(2) η οποία δημιουργεί μια χρονική καθυστέρηση των 2msec. Η τελευταία εντολή που εκτελείται στο μπλοκ εντολών της συνθήκης ελέγχου (else if) αφορά την κλήση της συνάρτησης (GStep_Motor_Stop ()), η οποία αποσκοπεί στην απενεργοποίηση του βηματικού κινητήρα. Σε περίπτωση όμως που δεν ικανοποιείται η προηγούμενη συνθήκη (else if), τότε εκτελείται η εντολή ελέγχου else if (frequency >=195.90 && frequency <=196.25), η οποία ελέγχει αν η τιμή της μεταβλητής (frequency) είναι μεγαλύτερη ή ίση από 195.90Hz και μικρότερη ή ίση από 196.25Hz. Συνεπώς, αν ικανοποιείται αυτή η συνθήκη τότε η πραγματική συχνότητα ταλάντωσης της χορδής Σολ (G) βρίσκεται εντός του εύρους των επιθυμητών συχνοτήτων της χορδής και εκτελείται το αντίστοιχο μπλοκ εντολών. Πιο συγκεκριμένα, εντός του συγκεκριμένου μπλοκ εντολών, αποθηκεύεται στην σημαία (flag) με όνομα (string_3) η

τιμή ένα. Με αυτόν τον τρόπο υποδηλώνεται στο πρόγραμμα η ολοκλήρωση του κουρδίσματος της χορδής Σολ (G). Έπειτα, εκτελείται η εντολή (k = 21), όπου πραγματοποιείται αρχικοποίηση του μετρητή (k) με την τιμή είκοσι ένα και αμέσως μετά, εκτελείται η εντολή (frequency = 0.00) η οποία μηδενίζει την τιμή της μεταβλητής (frequency). Ύστερα, πραγματοποιείται εκτέλεση των εντολών lcd.clear(), lcd.setCursor() και lcd.print(“ “) με την κατάλληλη σειρά, ώστε να εμφανιστεί το μήνυμα (Hit B string) στην LCD οθόνη. Το μήνυμα αυτό παραμένει στην LCD οθόνη για 1sec εξαιτίας της εντολής (delay1000) και έχει σκοπό να προτρέψει τον χρήστη να διεγείρει την επόμενη χορδή η οποία είναι η Σι (B). Στην συνεχεία, εκτελούνται ξανά οι εντολές lcd.clear(), lcd.setCursor() και lcd.print(“ “) με την κατάλληλη σειρά ώστε, να εμφανιστεί στην πρώτη γραμμή της LCD οθόνης το μήνυμα (B -> 246.94 Hz), το οποίο υποδηλώνει την επιθυμητή τιμή της συχνότητας της χορδής Σι (B) και στην δεύτερη γραμμή το μήνυμα (Hz Note) σε συγκεκριμένες θέσης. Πιο συγκεκριμένα, το μήνυμα της δεύτερης γραμμής αποσκοπεί στην εκτύπωση της πραγματική συχνότητα πριν το (Hz) και η νότα στην οποία αντιστοιχεί η συχνότητα ταλάντωσης της χορδής πριν το (Note) κατά την διάρκεια κουρδίσματος της χορδής Σι (B). Στην περίπτωση όμως που δεν ικανοποιείται καμία συνθήκη από τις εντολές ελέγχου (if-else if), τότε εκτελείται το μπλοκ εντολών της συνθήκης (else). Ουσιαστικά, το μπλοκ εντολών της συνθήκης (else) αποτελείται από μία μόνο εντολή η οποία μηδενίζει την τιμή της μεταβλητής (frequency). Με αυτό τον τρόπο, αντιμετωπίζονται οι συχνότητες που μπορούν να προκύψουν από την ταυτόχρονη διέγερση δύο χορδών ή από την διέγερση λανθασμένης χορδής. Σε αυτό το σημείο ολοκληρώνεται το μπλοκ εντολών του βρόγχου επανάληψης (while), το οποίο έχει να κάνει με το κούρδισμα της χορδής Σολ (G). Η διαδικασία που μόλις περιγράφτηκε, επαναλαμβάνεται μέχρι να επιτευχθεί το κούρδισμα της χορδής Σολ (G) εντός του εύρος των συχνοτήτων ταλάντωσης που έχει οριστεί για την συγκεκριμένη χορδή.

Αφού ολοκληρωθεί το κούρδισμα της χορδής Σολ (G), το πρόγραμμα συνεχίζει στην πραγματοποίηση του κουρδίσματος της χορδής Σι (B). Αυτό επιτυγχάνεται με την χρήση του βρόγχου επανάληψης while(), ο οποίος στο μπλοκ εντολών του περιλαμβάνει τις απαραίτητες εντολές για το κούρδισμα της χορδής Σι (B). Επιπλέον, η συνθήκη που βρίσκεται εντός των παρενθέσεων του βρόγχου επανάληψης (while) είναι η (string_2 != 1), η οποία δηλώνει ότι όσο η σημαία (flag) με όνομα (string_2) δεν είναι ίση με ένα, τόσο θα συνεχίζει να εκτελείται το μπλοκ εντολών του βρόγχου επανάληψης. Πιο συγκεκριμένα, όσο η σημαία (flag) με όνομα (string_2) είναι μηδέν αυτό σημαίνει ότι η χορδή Σι (B) είναι ξεκούρδιστη. Μόλις όμως, η σημαία (flag) με όνομα (string_2) γίνει ένα, τότε αυτό σημαίνει ότι η χορδή Σι (B) έχει κουρδιστεί και το πρόγραμμα μπορεί να συνεχίσει στο κούρδισμα της επόμενης χορδής. Επειδή όμως η συγκεκριμένη σημαία (flag) έχει αρχικοποιηθεί με την τιμή μηδέν, το μπλοκ εντολών του βρόγχου επανάληψης (while) θα εκτελεστεί μία φορά τουλάχιστον.

Εντός λοιπόν του μπλοκ εντολών του βρόγχου επανάληψης (while), εκτελούνται οι εντολές οι οποίες είναι απαραίτητες για το κούρδισμα της χορδής Σι (B). Πιο συγκεκριμένα, υπάρχει μία συνθήκη ελέγχου (if), η οποία ελέγχει την τιμή του μετρητή (k) αν είναι μεγαλύτερη από είκοσι. Ο μετρητής (k) έχει αρχικοποιηθεί στον προηγούμενο βρόγχο επανάληψης (while) με την τιμή είκοσι ένα, με αποτέλεσμα να εκτελείται το μπλοκ εντολών της συνθήκης ελέγχου (if). Πιο συγκεκριμένα, εντός του μπλοκ εντολών της συνθήκης (if), εκτελείτε πρώτα η εντολή pinMode(39, OUTPUT), η οποία καθορίζει τον ψηφιακό ακροδέκτη τριάντα εννιά ως έξοδο. Έπειτα, εκτελείται η εντολή

digitalWrite(39, LOW), όπου ο ψηφιακός ακροδέκτης τριάντα εννιά μεταβαίνει σε κατάσταση χαμηλού δυναμικού για 10msec, διότι αμέσως μετά εκτελείται η εντολή χρονικής καθυστέρησης delay(10). Μετά εκτελείται η εντολή pinMode(39, INPUT), η οποία καθορίζει τον ψηφιακό ακροδέκτη τριάντα εννιά ως είσοδο. Ύστερα, το μπλοκ εντολών της συνθήκης ελέγχου (if) κλείνει με την εντολή (k = 0), όπου γίνεται μηδενισμός του μετρητή (k). Σκοπός της συγκεκριμένης συνθήκης ελέγχου, είναι η γείωση (GND) του σήματος που προέρχεται από το μικρόφωνο επαφής ανά τακτά χρονικά διαστήματα, ώστε να μην υπάρχουν παράσιτα τα οποία παραμορφώνουν το σήμα και οδηγούν σε λανθασμένους υπολογισμούς της θεμελιώδης συχνότητας ταλάντωσης.

Στην συνέχεια, μετά την συνθήκη ελέγχου (if), γίνεται κλήση της συνάρτησης FREQ_DETECTION(), η οποία ξεκινά την διαδικασία δειγματοληψίας του σήματος από το μικρόφωνο επαφής και αναλύεται σε επόμενη ενότητα. Η διαδικασία δειγματοληψίας, διαρκεί για 0.5sec εξαιτίας της εντολής καθυστέρησης delay(500) που εκτελείται αμέσως μετά. Ύστερα, γίνεται κλήση της συνάρτησης FREQ_DETECTION1(), η οποία αναλύεται σε επόμενη ενότητα και έχει σκοπό τόσο τον υπολογισμό της θεμελιώδης συχνότητας ταλάντωσης της χορδής όσο και την εύρεση της νότας στην οποία αντιστοιχεί η συγκεκριμένη συχνότητα. Έπειτα, γίνεται κλήση της συνάρτησης FREQ_DETECTION_STOP() η οποία αναλύεται και αυτή σε επόμενη ενότητα και αποσκοπεί στην παύση της δειγματοληψίας του σήματος από το μικρόφωνο επαφής απενεργοποιώντας τον αναλογικό-ψηφιακό μετατροπέα.

Μετά λοιπόν την ολοκλήρωση της δειγματοληψίας και αφού έχει υπολογιστεί η θεμελιώδης συχνότητα ταλάντωσης της χορδής, εμφανίζεται στην σειριακή οθόνη το μήνυμα (B String) και η τιμή της συχνότητας που έχει υπολογιστεί. Αυτό επιτυγχάνεται με την χρήση της εντολής Serial.println(). Στην συνέχεια, χρησιμοποιούνται οι εντολές lcd.setCursor() και lcd.print(“ “) η οποίες έχουν σκοπό την μετακίνηση του κέρσορα στην πρώτη στήλη της δεύτερης γραμμής της LCD οθόνης, το σβήσιμο της παλιάς τιμής της συχνότητας και την εμφάνιση της πραγματικής τιμής της θεμελιώδης συχνότητας ταλάντωσης της χορδής που υπολογίστηκε. Με αυτόν τον τρόπο, ο χρήστης μπορεί να δει στην LCD οθόνη την πραγματική τιμή της συχνότητας ταλάντωσης της χορδής Σι (B). Αφότου ολοκληρωθεί η εμφάνιση της συχνότητας στην LCD οθόνη, εκτελείται η εντολή (k++), η οποία έχει σκοπό την αύξηση της τιμής του μετρητή (k) κατά μία μονάδα.

Εν συνεχεία, εφαρμόζεται η τεχνική ελέγχου (if – else if – else), με σκοπό τον έλεγχο της συχνότητας ταλάντωσης της χορδής. Πιο συγκεκριμένα, με την εντολή ελέγχου if (frequency < 246.54 && frequency > 207 && frequency > 0), ελέγχεται αν η τιμή της μεταβλητής (frequency) είναι μικρότερη από 246.54Hz και μεγαλύτερη από 207Hz και μεγαλύτερη από 0Hz. Επομένως, αν ικανοποιείται αυτή η συνθήκη σημαίνει ότι η συχνότητα ταλάντωσης της χορδής Σι (B) είναι χαμηλότερη από την επιθυμητή συχνότητα και για να διορθωθεί θα πρέπει να εκτελεστεί το μπλοκ εντολών της συνθήκης ελέγχου (if). Εντός λοιπόν του μπλοκ εντολών, γίνεται μηδενισμός της μεταβλητής (frequency) και ύστερα εκτελείται ένας βρόγχος επανάληψης (for). Ο βρόγχος επανάληψης (for) επαναλαμβάνει το μπλοκ εντολών του είκοσι ένα φορές. Εντός του μπλοκ εντολών του βρόγχου επανάληψης (for) γίνεται κλήση της συνάρτησης (Counterclockwise_BStep_Motor ()). Η συνάρτηση αυτή αναλύεται σε επόμενη ενότητα και έχει σκοπό την αντί-ωρολογιακή περιστροφή του άξονα του βηματικού κινητήρα και κατά συνέπεια την αντί-ωρολογιακή περιστροφή του κλειδιού της χορδής Σι (B). Μετά την ολοκλήρωση του βρόγχου επανάληψης (for), εκτελείται η εντολή delay(2) η οποία

δημιουργεί μια χρονική καθυστέρηση των 2msec. Το μπλοκ εντολών της συνθήκης ελέγχου (if) ολοκληρώνεται με την κλήση της συνάρτησης (BStep_Motor_Stop ()), η οποία αναλύεται σε επόμενο κεφάλαιο και σκοπός της είναι να απενεργοποιήσει τον βηματικό κινητήρα. Στην περίπτωση όμως, που δεν ικανοποιείται η συνθήκη ελέγχου (if), τότε εκτελείται η εντολή ελέγχου else if (frequency > 247.04 && frequency < 293 && frequency > 0), όπου ελέγχεται αν η τιμή της μεταβλητής (frequency) είναι μεγαλύτερη από 247.04Hz και μικρότερη από 293Hz και μεγαλύτερη από 0Hz. Συνεπώς, αν ικανοποιείται αυτή η συνθήκη σημαίνει ότι η συχνότητα ταλάντωσης της χορδής Σι (B) είναι υψηλότερη από την επιθυμητή συχνότητα και για να διορθωθεί θα πρέπει να εκτελεστεί το μπλοκ εντολών της συνθήκης ελέγχου (else if). Εντός λοιπόν του μπλοκ εντολών, γίνεται μηδενισμός της μεταβλητής (frequency) και ύστερα εκτελείται ένας βρόγχος επανάληψης (for). Εντός του μπλοκ εντολών του βρόγχου επανάληψης (for) γίνεται κλήση της συνάρτησης (Clockwise_BStep_Motor ()), η οποία αναλύεται σε επόμενη ενότητα. Σκοπός της συνάρτησης (Clockwise_BStep_Motor ()), είναι να περιστρέψει τον άξονα του βηματικού κινητήρα ωρολογιακά και κατά συνέπεια να περιστρέψει ωρολογιακά και το κλειδί της χορδής Σι (B). Αφότου ολοκληρωθεί ο βρόγχος επανάληψης (for), εκτελείται η εντολή delay(2) η οποία δημιουργεί μια χρονική καθυστέρηση των 2msec. Η τελευταία εντολή που εκτελείται στο μπλοκ εντολών της συνθήκης ελέγχου (else if) αφορά την κλήση της συνάρτησης (BStep_Motor_Stop ()), η οποία έχει σκοπό να απενεργοποιήσει τον βηματικό κινητήρα. Αν όμως δεν ικανοποιείται η προηγούμενη συνθήκη (else if), τότε εκτελείται η εντολή ελέγχου else if (frequency >= frequency >=246.55 && frequency <=247.04), η οποία ελέγχει αν η τιμή της μεταβλητής (frequency) είναι μεγαλύτερη ή ίση από 246.55Hz και μικρότερη ή ίση από 247.04Hz. Επομένως, αν ικανοποιείται αυτή η συνθήκη τότε η πραγματική συχνότητα ταλάντωσης της χορδής Σι (B) βρίσκεται εντός του εύρους των επιθυμητών συχνοτήτων της χορδής και εκτελείται το αντίστοιχο μπλοκ εντολών. Εντός λοιπόν του συγκεκριμένου μπλοκ εντολών, αποθηκεύεται στην σημαία (flag) με όνομα (string_2) η τιμή ένα. Με αυτόν τον τρόπο υποδηλώνεται στο πρόγραμμα η ολοκλήρωση του κουρδίσματος της χορδής Σι (B). Ύστερα, εκτελείται η εντολή (k = 21), όπου πραγματοποιείται αρχικοποίηση του μετρητή (k) με την τιμή είκοσι ένα και αμέσως μετά, πραγματοποιείται μηδενισμός της μεταβλητής (frequency) μέσω της εντολής (frequency = 0.00). Έπειτα, εκτελούνται οι εντολές lcd.clear(), lcd.setCursor() και lcd.print(“ “) με την κατάλληλη σειρά, ώστε να εμφανιστεί το μήνυμα (Hit e string) στην LCD οθόνη. Το μήνυμα αυτό παραμένει στην LCD οθόνη για 1sec εξαιτίας της εντολής (delay1000) και σκοπό έχει να προτρέψει τον χρήστη να διεγείρει την επόμενη χορδή η οποία είναι η Μι καντίνι (e). Εν συνεχεία, εκτελούνται ξανά οι εντολές lcd.clear(), lcd.setCursor() και lcd.print(“ “) με την κατάλληλη σειρά ώστε, να εμφανιστεί στην πρώτη γραμμή της LCD οθόνης το μήνυμα (e -> 329.63 Hz), το οποίο υποδηλώνει την επιθυμητή τιμή της συχνότητας της χορδής Μι (e) και στην δεύτερη γραμμή το μήνυμα (Hz Note) σε συγκεκριμένες θέσης. Σκοπός του μηνύματος της δεύτερης γραμμής είναι να εκτυπώνεται η πραγματική συχνότητα πριν το (Hz) και η νότα στην οποία αντιστοιχεί η συχνότητα ταλάντωσης της χορδής πριν το (Note) κατά την διάρκεια κουρδίσματος της χορδής Μι (e). Αν όμως, δεν ικανοποιείται καμία συνθήκη από τις εντολές ελέγχου (if-else if), τότε εκτελείται το μπλοκ εντολών της συνθήκης (else). Πιο συγκεκριμένα, το μπλοκ εντολών της συνθήκης (else) περιλαμβάνει μία μόνο εντολή η οποία μηδενίζει την τιμή της μεταβλητής (frequency). Με αυτό τον τρόπο, αντιμετωπίζονται οι συχνότητες που μπορούν να προκύψουν από την ταυτόχρονη διέγερση δύο χορδών ή από την διέγερση λανθασμένης χορδής. Σε αυτό το σημείο

ολοκληρώνεται το μπλοκ εντολών του βρόγχου επανάληψης (while), το οποίο έχει να κάνει με το κούρδισμα της χορδής Σι (B). Η διαδικασία που μόλις περιγράφτηκε, εκτελείτε ξανά μέχρι να πραγματοποιηθεί το κούρδισμα της χορδής Σι (B) εντός του εύρος των συχνοτήτων ταλάντωσης που έχει οριστεί για την συγκεκριμένη χορδή.

Με την ολοκλήρωση του κουρδίσματος της χορδής Σι (B), σειρά έχει το κούρδισμα της χορδής Μι καντίνι (e). Αυτό επιτυγχάνεται με την χρήση του βρόγχου επανάληψης while(), ο οποίος στο μπλοκ εντολών του περιλαμβάνει τις απαραίτητες εντολές για το κούρδισμα της χορδής Μι καντίνι (e). Η συνθήκη που βρίσκεται εντός των παρενθέσεων του βρόγχου επανάληψης (while) είναι η (string_1 != 1) η οποία σημαίνει ότι όσο η σημαία (flag) με όνομα (string_1) δεν είναι ίση με ένα, τόσο θα συνεχίζει να εκτελείται το μπλοκ εντολών του βρόγχου επανάληψης. Πιο συγκεκριμένα, όσο η σημαία (flag) με όνομα (string_1) είναι μηδέν αυτό σημαίνει ότι η χορδή Μι καντίνι (e) είναι ξεκούρδιστη. Μόλις όμως η σημαία (flag) με όνομα (string_1) γίνει ένα, τότε αυτό σημαίνει ότι η χορδή Μι καντίνι (e) έχει κουρδιστεί. Επειδή όμως η συγκεκριμένη σημαία (flag) έχει αρχικοποιηθεί με την τιμή μηδέν, το μπλοκ εντολών του βρόγχου επανάληψης (while) θα εκτελεστεί μία φορά τουλάχιστον.

Ουσιαστικά, εντός του μπλοκ εντολών του βρόγχου επανάληψης (while), εκτελούνται οι εντολές οι οποίες είναι απαραίτητες για το κούρδισμα της χορδής Μι καντίνι (e). Όπως αναφέρθηκε και προηγουμένως, υπάρχει μία συνθήκη ελέγχου (if), η οποία ελέγχει την τιμή του μετρητή (k) αν είναι μεγαλύτερη από είκοσι. Ο μετρητής (k) έχει αρχικοποιηθεί στον προηγούμενο βρόγχο επανάληψης (while) με την τιμή είκοσι ένα, με αποτέλεσμα να εκτελείται το μπλοκ εντολών της συνθήκης ελέγχου (if). Ουσιαστικά, στο μπλοκ εντολών της συνθήκης (if) εκτελείτε πρώτα η εντολή pinMode(39, OUTPUT), η οποία καθορίζει τον ψηφιακό ακροδέκτη τριάντα εννιά ως έξοδο. Έπειτα, εκτελείται η εντολή digitalWrite(39, LOW), όπου ο ψηφιακός ακροδέκτης τριάντα εννιά μεταβαίνει σε κατάσταση χαμηλού δυναμικού για 10msec, διότι αμέσως μετά εκτελείται η εντολή χρονικής καθυστέρησης delay(10). Ύστερα, εκτελείται η εντολή pinMode(39, INPUT), η οποία καθορίζει τον ψηφιακό ακροδέκτη τριάντα εννιά ως είσοδο. Το μπλοκ εντολών της συνθήκης ελέγχου (if) ολοκληρώνεται με την εντολή (k = 0), όπου γίνεται μηδενισμός του μετρητή (k). Σκοπός της συγκεκριμένης συνθήκης ελέγχου είναι η γείωση (GND) του σήματος που προέρχεται από το μικρόφωνο επαφής ανά τακτά χρονικά διαστήματα, ώστε να μην υπάρχουν παράσιτα τα οποία παραμορφώνουν το σήμα και οδηγούν σε λανθασμένους υπολογισμούς της συχνότητας ταλάντωσης της χορδής..

Αφότου ολοκληρωθεί η εντολή της συνθήκης ελέγχου (if), γίνεται κλήση της συνάρτησης FREQ_DETECTION(), η οποία αναλύεται σε επόμενη ενότητα και αποσκοπεί στην εκκίνηση της διαδικασίας της δειγματοληψίας του σήματος από το μικρόφωνο επαφής. Η διαδικασία δειγματοληψίας, διαρκεί για 0.5sec εξαιτίας της εντολής καθυστέρησης delay(500) που εκτελείται αμέσως μετά. Ύστερα, γίνεται κλήση της συνάρτησης FREQ_DETECTION1(), η οποία έχει σκοπό τόσο τον υπολογισμό της θεμελιώδης συχνότητας ταλάντωσης της χορδής όσο και την εύρεση της νότας στην οποία αντιστοιχεί η συγκεκριμένη συχνότητα και η ανάλυσή της γίνεται σε επόμενη ενότητα. Έπειτα, γίνεται κλήση της συνάρτησης FREQ_DETECTION_STOP() η οποία αναλύεται και αυτή σε επόμενη ενότητα και αποσκοπεί στην παύση της δειγματοληψίας του σήματος από το μικρόφωνο επαφής απενεργοποιώντας τον αναλογικό-ψηφιακό μετατροπέα.

Εν συνεχεία, αφού έχει υπολογιστεί η θεμελιώδης συχνότητα ταλάντωσης της χορδής, εμφανίζεται στην σειριακή οθόνη το μήνυμα (e String) και η τιμή της συχνότητας

που έχει υπολογιστεί. Αυτό επιτυγχάνεται με την χρήση της εντολής Serial.println(). Στην συνέχεια, χρησιμοποιούνται οι εντολές lcd.setCursor() και lcd.print(“ “) η οποίες έχουν σκοπό την μετακίνηση του κέρσορα στην πρώτη στήλη της δεύτερης γραμμής της LCD οθόνης, το σβήσιμο της παλιάς τιμής της συχνότητας και την εμφάνιση της πραγματικής τιμής της συχνότητας που υπολογίστηκε. Με αυτόν τον τρόπο, ο χρήστης μπορεί να δει στην LCD οθόνη την πραγματική τιμή της συχνότητας ταλάντωσης της χορδής Μι καντίνι (e). Αφότου ολοκληρωθεί η εμφάνιση της συχνότητας στην LCD οθόνη, εκτελείται η εντολή (k++), η οποία έχει σκοπό την αύξηση της τιμής του μετρητή (k) κατά μία μονάδα.

Έπειτα, εφαρμόζεται η τεχνική ελέγχου (if – else if – else), η οποία αποσκοπεί στον έλεγχο της συχνότητας ταλάντωσης της χορδής. Ουσιαστικά, με την εκτέλεση της εντολής ελέγχου if (frequency < 328.73 && frequency > 277 && frequency > 0), ελέγχεται αν η τιμή της μεταβλητής (frequency) είναι μικρότερη από 328.73Hz και μεγαλύτερη από 277Hz και μεγαλύτερη από 0Hz. Συνεπώς, αν ικανοποιείται αυτή η συνθήκη σημαίνει ότι η συχνότητα ταλάντωσης της χορδής Μι καντίνι (e) είναι χαμηλότερη από την επιθυμητή συχνότητα και για να διορθωθεί θα πρέπει να εκτελεστεί το μπλοκ εντολών της συνθήκης ελέγχου (if). Εντός λοιπόν του μπλοκ εντολών, γίνεται μηδενισμός της μεταβλητής (frequency) και ύστερα εκτελείται ένας βρόγχος επανάληψης (for) ο οποίος επαναλαμβάνει το μπλοκ εντολών του είκοσι ένα φορές. Όσον αφορά το μπλοκ εντολών του βρόγχου επανάληψης (for), γίνεται κλήση της συνάρτησης (Counterclockwise_eStep_Motor ()). Η συνάρτηση αυτή αναλύεται σε επόμενη ενότητα και έχει σκοπό την αντί-ωρολογιακή περιστροφή του άξονα του βηματικού κινητήρα και κατά συνέπεια την αντί-ωρολογιακή περιστροφή του κλειδιού της χορδής Μι καντίνι (e). Μετά την ολοκλήρωση του βρόγχου επανάληψης (for), εκτελείται η εντολή delay(2) η οποία δημιουργεί μια χρονική καθυστέρηση των 2msec. Το μπλοκ εντολών της συνθήκης ελέγχου (if) ολοκληρώνεται με την κλήση της συνάρτησης (eStep_Motor_Stop ()), η οποία αναλύεται σε επόμενο κεφάλαιο και σκοπός της είναι να απενεργοποιήσει τον βηματικό κινητήρα. Αν όμως δεν ικανοποιείται η συνθήκη ελέγχου (if), τότε εκτελείται η εντολή ελέγχου else if (frequency > 331.58 && frequency < 392 && frequency > 0), όπου ελέγχεται αν η τιμή της μεταβλητής (frequency) είναι μεγαλύτερη από 331.58Hz και μικρότερη από 392Hz και μεγαλύτερη από 0Hz. Συνεπώς, αν ικανοποιείται αυτή η συνθήκη σημαίνει ότι η συχνότητα ταλάντωσης της χορδής Μι καντίνι (e) είναι υψηλότερη από την επιθυμητή συχνότητα και για να διορθωθεί θα πρέπει να εκτελεστεί το μπλοκ εντολών της συνθήκης ελέγχου (else if). Ουσιαστικά, εντός του μπλοκ εντολών γίνεται μηδενισμός της μεταβλητής (frequency) και ύστερα εκτελείται ένας βρόγχος επανάληψης (for). Εντός του μπλοκ εντολών του βρόγχου επανάληψης (for) γίνεται κλήση της συνάρτησης (Clockwise_eStep_Motor ()), της οποίας η ανάλυση πραγματοποιείται σε επόμενη ενότητα. Η κλήση της συνάρτησης (Clockwise_eStep_Motor ()), αποσκοπεί στην ωρολογιακή περιστροφή του άξονα του βηματικού κινητήρα και κατά συνέπεια στην ωρολογιακή περιστροφή του κλειδιού της χορδής Μι καντίνι (e). Μετά την ολοκλήρωση του βρόγχου επανάληψης (for), εκτελείται η εντολή delay(2) η οποία δημιουργεί μια χρονική καθυστέρηση των 2msec. Η τελευταία εντολή που εκτελείται στο μπλοκ εντολών της συνθήκης ελέγχου (else if) αφορά την κλήση της συνάρτησης (eStep_Motor_Stop ()), η οποία αποσκοπεί στην απενεργοποίηση του βηματικού κινητήρα. Σε περίπτωση όμως που δεν ικανοποιείται η προηγούμενη συνθήκη (else if), τότε εκτελείται η εντολή ελέγχου else if (frequency >=328.73 && frequency <=331.58), η οποία ελέγχει αν η τιμή της μεταβλητής (frequency) είναι μεγαλύτερη ή ίση από 328.73Hz και μικρότερη ή ίση από 331.58Hz. Συνεπώς, αν ικανοποιείται αυτή η συνθήκη τότε η πραγματική συχνότητα

ταλάντωσης της χορδής Μι καντίνι (e) βρίσκεται εντός του εύρους των επιθυμητών συχνοτήτων της χορδής και εκτελείται το αντίστοιχο μπλοκ εντολών. Πιο συγκεκριμένα, εντός του συγκεκριμένου μπλοκ εντολών, αποθηκεύεται στην σημαία (flag) με όνομα (string_1) η τιμή ένα. Με αυτόν τον τρόπο υποδηλώνεται στο πρόγραμμα η ολοκλήρωση του κουρδίσματος της χορδής Μι καντίνι (e). Έπειτα, εκτελείται η εντολή (k = 21), όπου πραγματοποιείται αρχικοποίηση του μετρητή (k) με την τιμή είκοσι ένα και αμέσως μετά, εκτελείται η εντολή (frequency = 0.00) η οποία μηδενίζει την τιμή της μεταβλητής (frequency). Στην περίπτωση όμως που δεν ικανοποιείται καμία συνθήκη από τις εντολές ελέγχου (if-else if), τότε εκτελείται το μπλοκ εντολών της συνθήκης (else). Ουσιαστικά, το μπλοκ εντολών της συνθήκης (else) αποτελείται από μία μόνο εντολή η οποία μηδενίζει την τιμή της μεταβλητής (frequency). Με αυτό τον τρόπο, αντιμετωπίζονται οι συχνότητες που μπορούν να προκύψουν από την ταυτόχρονη διέγερση δύο χορδών ή από την διέγερση λανθασμένης χορδής. Σε αυτό το σημείο ολοκληρώνεται το μπλοκ εντολών του βρόγχου επανάληψης (while), το οποίο έχει να κάνει με το κούρδισμα της χορδής Μι καντίνι (e). Η διαδικασία που μόλις περιγράφτηκε, επαναλαμβάνεται μέχρι να επιτευχθεί το κούρδισμα της χορδής Μι καντίνι (e) εντός του εύρος των συχνοτήτων ταλάντωσης που έχει οριστεί για την συγκεκριμένη χορδή.

Αφού λοιπόν κουρδιστεί και η τελευταία χορδή της κιθάρας, εκτελούνται οι τελευταίες εντολές του μπλοκ εντολών της συνθήκης ελέγχου (if). Πιο συγκεκριμένα, εκτελούνται οι εντολές lcd.clear(), lcd.setCursor() και lcd.print(“ “) με κατάλληλη σειρά ώστε να εμφανιστεί στην LCD οθόνη το μήνυμα (TUNE FINISHED!). Το μήνυμα αυτό παραμένει στην οθόνη για 5sec εξαιτίας της εντολής καθυστέρησης που εκτελείται delay(5000). Έπειτα, γίνεται κλήση της συνάρτησης Menu (), ώστε να εμφανιστεί στον χρήστη μέσω της LCD οθόνης το μενού επιλογών.

Πίνακας 3.5: Ο κώδικας προγραμματισμού για την αναγνώριση του πλήκτρου SELECT από το σύστημα και του κλασσικού κουρδίσματος (Standard Tune) της κιθάρας [31].

Κώδικας Προγραμματισμού (Μέρος 5ο)

else if(x<800 && x!=689) //If SELECT button has been pressed
{
x = 1023 ; //Store value at x variable
if (menu==1) //If menu value is 1, then
{
lcd.clear(); //Clear LCD screen
lcd.setCursor(0,0); //Set cursor at first column and first line
lcd.print(" Standard Tune "); //Print message at LCD screen
lcd.setCursor(4,1); //Set cursor at fifth column and first line
lcd.print("Selected"); //Print message at LCD screen
delay(1000); //Delay 1 sec
lcd.clear(); //Clear LCD screen
lcd.setCursor(2,0); //Set Cursor at third column and first line
lcd.print("Hit E string"); //Print message at LCD screen
delay(1000); //Delay 1 sec

```
    lcd.clear(); //Clear LCD screen
    lcd.setCursor(1,0); //Set Cursor at third column and first line
    lcd.print("E -> 82.41 Hz"); //Print message at LCD screen
    lcd.setCursor(7,1); //Set Cursor at first column and second line
    lcd.print("Hz   Note"); //print message at LCD screen
    k = 21; //Store value at k variable
    string_6 = 0;//Reset the value at string_6 variable
    string_5 = 0;//Reset the value at string_5 variable
    string_4 = 0;//Reset the value at string_4 variable
    string_3 = 0;//Reset the value at string_3 variable
    string_2 = 0;//Reset the value at string_2 variable
    string_1 = 0;//Reset the value at string_1 variable
    while(string_6 != 1) //While string_6 flag isn't set repeat
    {
     if (k>20) //If k variable is bigger than 20, then
     {
      pinMode(39,OUTPUT); //Determine digital pin 39 as Output
      digitalWrite(39,LOW); //Set low voltage (GND) at pin 39
      delay(10); //Delay 10 msec
      pinMode(39,INPUT); //Determine digital pin 39 as Input
      k = 0; //Reset k varible
     } //if (k>20)
     FREQ_DETECTION(); //Call FREQ_DETECTION function (Change analog port for
the income signal)
     delay(500); //Delay 0.5 sec
     FREQ_DETECTION1(); //Call FREQ_DETECTION1 function (Calculate frequency
and stop interrupts by ADC)
     FREQ_DETECTION_STOP(); //Call FREQ_DETECTION_STOP function (Stop
frequency detection)
     Serial.println("E String"); //Print message at Serial Port
     Serial.println(frequency); //Print frequency value at Serial Port
     lcd.setCursor(0,1); //Set cursor at first column and second line
     lcd.print("       "); //Print message at lcd screen
     lcd.setCursor(0,1); //Set cursor at first column and second line
     lcd.print(frequency); //Print frequency value at LCD screen
     k++; //Increase variable k value by one
     if (frequency < 82.35 && frequency > 65 && frequency > 0) //If frequency value is
smaller than 82.35Hz and bigger than 65Hz and bigger than 0Hz, then
     {
      frequency = 0.00; //Reset frequency value
```

```
        for (i=0; i<=20; i++) //For 21 times repeat
        {
          Counterclockwise_EStep_Motor (); //Call function for E bass string step motor
        }
        delay(2); //Delay 2 msec
        EStep_Motor_Stop (); //Disable E bass string step motor
      } //if (frequency...
      else if (frequency > 82.47 && frequency < 103 && frequency > 0) //Else if
frequency value is bigger than 82.47Hz and smaller than 103Hz and bigger than 0Hz, then
      {
        frequency = 0.00; //Reset frequency value
        for(i=0; i<=20; i++) //For 21 times repeat
        {
          Clockwise_EStep_Motor (); //Call function for E bass string step motor
        }
        delay(2); //Delay 2 msec
        EStep_Motor_Stop (); //Disable E bass step motor
      } //else if (frequency...
      else if (frequency >=82.35 && frequency <=82.47) //Else if frequency value is bigger
or equal with 82.35Hz an smaller or equal with 82.47Hz, then E bass string has been tuned
      {
        string_6 = 1; //Set string_6 flag to finish tuning with E bass string
        k = 21; //Store value at k variable
        frequency = 0.00; //Reset frequency value
        lcd.clear(); //Clear LCD screen
        lcd.setCursor(2,0); //Set Cursor at third column and first line
        lcd.print("Hit A string"); //Print message at LCD screen
        delay(1000); //Delay 1 sec
        lcd.clear(); //Clear LCD screen
        lcd.setCursor(1,0); //Set Cursor at third column and first line
        lcd.print("A -> 110 Hz"); //Print message at LCD screen
        lcd.setCursor(7,1); //Set Cursor at first column and second line
        lcd.print("Hz   Note"); //print message at LCD screen
      } //else if (frequency...
      else //Else
      {
        frequency = 0.00; //Reset frequency value
      } //else
    } //while(string_6...
```

```
    while(string_5 != 1) //While string_5 flag isn't set repeat
    {
     if (k>20) //If k variable is bigger than 20, then
     {
      pinMode(39,OUTPUT); //Determine digital pin 39 as Output
      digitalWrite(39,LOW); //Set low voltage (GND) at pin 39
      delay(10); //Delay 10 msec
      pinMode(39,INPUT); //Determine digital pin 39 as Input
      k = 0;//Reset k variable
     }//if (k>20)
     FREQ_DETECTION(); //Call FREQ_DETECTION function (Change analog port for the income signal)
     delay(500); //Delay 0.5 sec
     FREQ_DETECTION1(); //Call FREQ_DETECTION1 function (Calculate frequency and stop interrupts by ADC)
     FREQ_DETECTION_STOP(); //Call FREQ_DETECTION_STOP function (Stop frequency detection)
     Serial.println("A String"); //Print message at Serial Port
     Serial.println(frequency); //Print frequency value at Serial Port
     lcd.setCursor(0,1); //Set cursor at first column and second line
     lcd.print("       "); //Print message at lcd screen
     lcd.setCursor(0,1); //Set cursor at first column and second line
     lcd.print(frequency); //Print frequency value at LCD screen
     k++; //Increase variable k value by one
     if (frequency < 109.80 && frequency > 92 && frequency > 0) //If frequency value is smaller than 109.80Hz and bigger than 92Hz and bigger than 0Hz, then
     {
      frequency = 0.00; //Reset frequency value
      for (i=0; i<=20; i++) //For 21 times repeat
      {
       Counterclockwise_AStep_Motor (); //Call function for A string step motor
      }
      delay(2); //Delay 2 msec
      AStep_Motor_Stop (); //Disable A string step motor
     } //if (frequency...
     else if (frequency > 110.21 && frequency < 130 && frequency > 0)  //Else if frequency value is bigger than 110.21Hz and smaller than 130Hz and bigger than 0Hz, then
     {
      frequency = 0.00; //Reset frequency value
```

```
        for(i=0; i<=20; i++) //For 21 times repeat
        {
          Clockwise_AStep_Motor (); //Call function for A string step motor
        }
        delay(2); //Delay 2 msec
        AStep_Motor_Stop (); //Disable A string step motor
      } //else if (frequency...
      else if (frequency >=109.80 && frequency <=110.21) //Else if frequency value is
bigger or equal with 109.80Hz an smaller or equal with 110.21Hz, then A string has been
tuned
      {
        string_5 = 1;//Set string_5 flag to finish tuning with A string
        k = 21; //Store value at k variable
        frequency = 0.00; //Reset frequency value
        lcd.clear(); //Clear LCD screen
        lcd.setCursor(2,0); //Set Cursor at third column and first line
        lcd.print("Hit D string"); //Print message at LCD screen
        delay(1000);
        lcd.clear(); //Clear LCD screen
        lcd.setCursor(1,0); //Set Cursor at third column and first line
        lcd.print("D -> 146.83 Hz"); //Print message at LCD screen
        lcd.setCursor(7,1); //Set Cursor at first column and second line
        lcd.print("Hz   Note"); //print message at LCD screen
      } //else if (frequency...
      else //Else
      {
        frequency = 0.00; //Reset frequency value
      }
    } //while(string_5...
    while(string_4 != 1) //While string_4 flag isn't set repeat
    {
      if (k>20) //If k variable is bigger than 20, then
      {
        pinMode(39,OUTPUT); //Determine digital pin 39 as Output
        digitalWrite(39,LOW); //Set low voltage (GND) at pin 39
        delay(10); //Delay 10 msec
        pinMode(39,INPUT); //Determine digital pin 39 as Input
        k = 0; //Reset k variable
      } //if (k>20)
```

```
    FREQ_DETECTION(); //Call FREQ_DETECTION function (Change analog port for the income signal)
    delay(500); //Delay 0.5 sec
    FREQ_DETECTION1(); //Call FREQ_DETECTION1 function (Calculate frequency and stop interrupts by ADC)
    FREQ_DETECTION_STOP(); //Call FREQ_DETECTION_STOP function (Stop frequency detection)
    Serial.println("D String"); //Print message at Serial Port
    Serial.println(frequency); //Print frequency value at Serial Port
    lcd.setCursor(0,1); //Set cursor at first column and second line
    lcd.print("       "); //Print message at lcd screen
    lcd.setCursor(0,1); //Set cursor at first column and second line
    lcd.print(frequency); //Print frequency value at LCD screen
    k++; //Increase variable k value by one
    if (frequency < 146.73 && frequency > 123 && frequency > 0) //If frequency value is smaller than 146.73Hz and bigger than 123Hz and bigger than 0Hz, then
    {
     frequency = 0.00; //Reset frequency value
     for (i=0; i<=40; i++) //For 41 times repeat
     {
      Counterclockwise_DStep_Motor (); //Call function for D string step motor
     }
     delay(2); //Delay 2 msec
     DStep_Motor_Stop (); //Disable D string step motor
    } //if (frequency...
    else if (frequency > 146.93 && frequency < 174 && frequency > 0)  //Else if frequency value is bigger than 146.93Hz and smaller than 174Hz and bigger than 0Hz, then
    {
     frequency = 0.00; //Reset frequency value
     for(i=0; i<=30; i++) //For 31 times repeat
     {
      Clockwise_DStep_Motor ();//Call function for D string step motor
     }
     delay(2); //Delay 2 msec
     DStep_Motor_Stop (); //Disable D string step motor
    } //else if (frequency...
    else if (frequency >=146.73 && frequency <=146.93) //Else if frequency value is bigger or equal with 146.73Hz an smaller or equal with 146.93Hz, then D string has been tuned
```

```
      {
        string_4 = 1; //Set string_4 flag to finish tuning with D string
        k = 21; //Store value at k variable
        frequency = 0.00; //Reset frequency value
        lcd.clear(); //Clear LCD screen
        lcd.setCursor(2,0); //Set Cursor at third column and first line
        lcd.print("Hit G string"); //Print message at LCD screen
        delay(1000);
        lcd.clear(); //Clear LCD screen
        lcd.setCursor(1,0); //Set Cursor at third column and first line
        lcd.print("G -> 196 Hz"); //Print message at LCD screen
        lcd.setCursor(7,1); //Set Cursor at first column and second line
        lcd.print("Hz   Note"); //print message at LCD screen
      } //else if (frequency...
      else //Else
      {
        frequency = 0.00; //Reset frequency value
      }
    } //while(string_4...*/
    while(string_3 != 1) //While string_3 flag isn't set repeat
    {
      if (k>20) //If k variable is bigger than 20, then
      {
        pinMode(39,OUTPUT); //Determine digital pin 39 as Output
        digitalWrite(39,LOW); //Set low voltage (GND) at pin 39
        delay(10); //Delay 10 msec
        pinMode(39,INPUT); //Determine digital pin 39 as Input
        k = 0; //Reset k variable
      } //if (k>20)
      FREQ_DETECTION(); //Call FREQ_DETECTION function (Change analog port for
the income signal)
      delay(500); //Delay 0.5 sec
      FREQ_DETECTION1(); //Call FREQ_DETECTION1 function (Calculate frequency
and stop interrupts by ADC)
      FREQ_DETECTION_STOP(); //Call FREQ_DETECTION_STOP function (Stop
frequency detection)
      Serial.println("G String"); //Print message at Serial Port
      Serial.println(frequency); //Print frequency value at Serial Port
      lcd.setCursor(0,1); //Set cursor at first column and second line
      lcd.print("       "); //Print message at lcd screen
```

```
      lcd.setCursor(0,1); //Set cursor at first column and second line
      lcd.print(frequency); //Print frequency value at LCD screen
      k++; //Increase variable k value by one
      if (frequency < 195.90 && frequency > 164 && frequency > 0) //If frequency value
is smaller than 195.90Hz and bigger than 164Hz and bigger than 0Hz, then
      {
       frequency = 0.00; //Reset frequency value
       for (i=0; i<=15; i++) //For 16 times repeat
       {
        Clockwise_GStep_Motor ();//Call function for G string step motor
       }
       delay(2); //Delay 2 msec
       GStep_Motor_Stop (); //Disable G string step motor
      } //if (frequency...
      else if (frequency > 196.25 && frequency < 233 && frequency > 0)  //Else if
frequency value is bigger than 196.25Hz and smaller than 233Hz and bigger than 0Hz,
then
      {
       frequency = 0.00; //Reset frequency value
       for(i=0; i<=15; i++) //For 16 times repeat
       {
        Counterclockwise_GStep_Motor (); //Call function for G string step motor
       }
       delay(2); //Delay 2 msec
       GStep_Motor_Stop (); //Disable G string step motor
      } //else if (frequency...
      else if (frequency >=195.90 && frequency <=196.25) //Else if frequency value is
bigger or equal with 195.90Hz an smaller or equal with 196.25Hz, then G string has been
tuned
      {
       string_3 = 1; //Set string_3 flag to finish tuning with G string
       k = 20; //Store value at k variable
       frequency = 0.00; //Reset frequency value
       lcd.clear(); //Clear LCD screen
       lcd.setCursor(2,0); //Set Cursor at third column and first line
       lcd.print("Hit B string"); //Print message at LCD screen
       delay(1000);
       lcd.clear(); //Clear LCD screen
       lcd.setCursor(1,0); //Set Cursor at third column and first line
       lcd.print("B -> 246.94 Hz"); //Print message at LCD screen
```

```
        lcd.setCursor(7,1); //Set Cursor at first column and second line
        lcd.print("Hz   Note"); //print message at LCD screen
       } //else if (frequency...
       else //Else
       {
        frequency = 0.00; //Reset frequency value
       }
     } //while(string_3...*/
     while(string_2 != 1) //While string_2 flag isn't set repeat
     {
      if (k>20) //If k variable is bigger than 20, then
      {
       pinMode(39,OUTPUT); //Determine digital pin 39 as Output
       digitalWrite(39,LOW); //Set low voltage (GND) at pin 39
       delay(10); //Delay 10 msec
       pinMode(39,INPUT); //Determine digital pin 39 as Input
       k = 0; //Reset k variable
      } //if (k>20)
      FREQ_DETECTION(); //Call FREQ_DETECTION function (Change analog port for the income signal)
      delay(500); //Delay 0.5 sec
      FREQ_DETECTION1(); //Call FREQ_DETECTION1 function (Calculate frequency and stop interrupts by ADC)
      FREQ_DETECTION_STOP(); //Call FREQ_DETECTION_STOP function (Stop frequency detection)
      Serial.println("B String"); //Print message at Serial Port
      Serial.println(frequency); //Print frequency value at Serial Port
      lcd.setCursor(0,1); //Set cursor at first column and second line
      lcd.print("       "); //Print message at lcd screen
      lcd.setCursor(0,1); //Set cursor at first column and second line
      lcd.print(frequency); //Print frequency value at LCD screen
      k++; //Increase variable k value by one
      if (frequency < 246.54 && frequency > 207 && frequency > 0)  //If frequency value is smaller than 246.54Hz and bigger than 207Hz and bigger than 0Hz, then
      {
       frequency = 0.00; //Reset frequency value
       for (i=0; i<=20; i++) //For 21 times repeat
       {
        Clockwise_BStep_Motor (); //Call function for B string step motor
       }
```

```
        delay(2); //Delay 2 msec
        BStep_Motor_Stop (); //Disable B string step motor
       } //if (frequency...
      else if (frequency > 247.04 && frequency < 293 && frequency > 0)  //Else if frequency value is bigger than 247.04Hz and smaller than 293Hz and bigger than 0Hz, then
       {
        frequency = 0.00; //Reset frequency value
        for(i=0; i<=20; i++) //For 21 times repeat
        {
         Counterclockwise_BStep_Motor (); //Call function for B string step motor
        }
        delay(2); //Delay 2 msec
        BStep_Motor_Stop (); //Disable B string step motor
       } //else if (frequency...
      else if (frequency >=246.55 && frequency <=247.04) //Else if frequency value is bigger or equal with 246.55Hz an smaller or equal with 247.04Hz, then B string has been tuned
       {
        string_2 = 1; //Set string_2 flag to finish tuning with B string
        k = 21; //Store value at k variable
        frequency = 0.00; //Reset frequency value
        lcd.clear(); //Clear LCD screen
        lcd.setCursor(2,0); //Set Cursor at third column and first line
        lcd.print("Hit e string"); //Print message at LCD screen
        delay(1000);
        lcd.clear(); //Clear LCD screen
        lcd.setCursor(1,0); //Set Cursor at third column and first line
        lcd.print("e -> 329.63 Hz"); //Print message at LCD screen
        lcd.setCursor(7,1); //Set Cursor at first column and second line
        lcd.print("Hz   Note"); //print message at LCD screen
       } //else if (frequency...
      else //Else
       {
        frequency = 0.00; //Reset frequency value
       }
     } //while(string_2...*/
     while(string_1 != 1) //While string_1 flag isn't set repeat
     {
      if (k>20) //If k variable is bigger than 20, then
```

```
    {
      pinMode(39,OUTPUT); //Determine digital pin 39 as Output
      digitalWrite(39,LOW); //Set low voltage (GND) at pin 39
      delay(10); //Delay 10 msec
      pinMode(39,INPUT); //Determine digital pin 39 as Input
      k = 0; //Reset k variable
    } //if (k>20)
    FREQ_DETECTION(); //Call FREQ_DETECTION function (Change analog port for the income signal)
    delay(500); //Delay 0.5 sec
    FREQ_DETECTION1(); //Call FREQ_DETECTION1 function (Calculate frequency and stop interrupts by ADC)
    FREQ_DETECTION_STOP(); //Call FREQ_DETECTION_STOP function (Stop frequency detection)
    Serial.println("e String"); //Print message at Serial Port
    Serial.println(frequency); //Print frequency value at Serial Port
    lcd.setCursor(0,1); //Set cursor at first column and second line
    lcd.print("       "); //Print message at lcd screen
    lcd.setCursor(0,1); //Set cursor at first column and second line
    lcd.print(frequency); //Print frequency value at LCD screen
    k++; //Increase variable k value by one
    if (frequency < 328.73 && frequency > 277 && frequency > 0) //If frequency value is smaller than 328.73Hz and bigger than 277Hz and bigger than 0Hz, then
    {
      frequency = 0.00; //Reset frequency value
      for (i=0; i<=20; i++) //For 21 times repeat
      {
        Clockwise_eStep_Motor (); //Call function for e string step motor
      }
      delay(2); //Delay 2 msec
      eStep_Motor_Stop (); //Disable e string step motor
    } //if (frequency...
    else if (frequency > 331.58 && frequency < 392 && frequency > 0)  //Else if frequency value is bigger than 331.58Hz and smaller than 392Hz and bigger than 0Hz, then
    {
      frequency = 0.00; //Reset frequency value
      for(i=0; i<=20; i++) //For 21 times repeat
      {
        Counterclockwise_eStep_Motor (); //Call function for e string step motor
```

```
        }
        delay(2); //Delay 2 msec
        eStep_Motor_Stop (); //Disable e string step motor
      } //else if (frequency...
      else if (frequency >=328.73 && frequency <=331.58) //Else if frequency value is
bigger or equal with 328.73Hz an smaller or equal with 331.58Hz, then e string has been
tuned
      {
        string_1 = 1; //Set string_1 flag to finish tuning with e string
        k = 21; //Store value at k variable
        frequency = 0.00; //Reset frequency value
      } //else if (frequency...
      else //Else
      {
        frequency = 0.00; //Reset frequency value
      }
    } //while(string_1...*/
    lcd.clear(); //Clear LCD screen
    lcd.setCursor(1,0); //Set cursor at second column and first line
    lcd.print("Tune finished!"); //Print message at LCD
    delay(5000); //Delay 5 sec
    Menu(); //Call Menu () function
  } //if (menu = =1)
```

Όπως αναφέρθηκε και προηγουμένως, εντός του μπλοκ εντολών του πλήκτρου SELECT, χρησιμοποιείται η τεχνική (if – else if) ώστε να μπορέσει το σύστημα να αναγνωρίσει την επιλογή που έχει κάνει ο χρήστης μέσα από το μενού επιλογής. Στον προηγούμενο πίνακα (Πίνακα 3.5) παρουσιάζεται ο κώδικας, ο οποίος αφορά την συνθήκη ελέγχου if(menu = = 1), η οποία ελέγχει αν η μεταβλητή (menu) είναι ίση με την τιμή ένα και αντίστοιχα εκτελεί το μπλοκ εντολών για την επίτευξη του κλασσικού κουρδίσματος στην κιθάρα (Standard Tune). Στην περίπτωση όμως, που δεν ικανοποιείται η συνθήκη ελέγχου (if), δηλαδή δεν έχει επιλεχτεί το κλασσικό κούρδισμα (Standard Tune), τότε εκτελείται η αμέσως επόμενη εντολή ελέγχου else if(menu = = 2). Η συγκεκριμένη συνθήκη ελέγχου (else if) ελέγχει αν η τιμή της μεταβλητής (menu) είναι ίση με δύο. Συνεπώς, σε περίπτωση που ικανοποιείται η συγκεκριμένη συνθήκη, τότε εκτελείτε το μπλοκ εντολών της συγκεκριμένης συνθήκης ελέγχου (else if), γεγονός που σημαίνει ότι έχει επιλεχτεί το κούρδισμα όλων των χορδών της κιθάρας ένα ημίτονο κάτω (Half-Step Down Tune) (Πίνακας 3.6)

Στην συνέχεια, οι πρώτες εντολές που εκτελούνται στο μπλοκ εντολών της συνθήκης (else if) έχουν να κάνουν με το σβήσιμο χαρακτήρων στην LCD οθόνη μέσω της εντολής lcd.clear(), με την μετακίνηση του κέρσορα σε συγκεκριμένη στήλη και γραμμή μέσω της εντολής lcd.setCursor(), αλλά και με την εκτύπωση μηνυμάτων στην

LCD οθόνης μέσω της εντολής lcd.print(“ “). Πιο συγκεκριμένα, πρώτα σβήνονται όλοι οι χαρακτήρες και ύστερα εμφανίζεται το μήνυμα (Half Step Down Selected) για 1 sec εξαιτίας της εντολής καθυστέρησης delay(1000) που εκτελείται αμέσως μετά. Σκοπός του μηνύματος αυτού είναι να ενημερωθεί ο χρήστης από το σύστημα ότι έχει επιλεχθεί το κούρδισμα όλων των χορδών της κιθάρας ενός ημιτόνου κάτω (Half-Step Down Tune). Έπειτα, σβήνονται ξανά οι χαρακτήρες από την LCD οθόνη και εμφανίζεται το μήνυμα (HIT Eb STRING) για 1 sec εξαιτίας της εντολής καθυστέρησης delay(1000) που εκτελείται αμέσως μετά. Το μήνυμα αυτό έχει σκοπό να προτρέψει τον χρήστη να διεγείρει την χορδή Μιb μπάσο (Eb). Ύστερα, σβήνονται οι χαρακτήρες από την LCD οθόνη και εμφανίζεται το μήνυμα (Eb -> 77.78 Hz) στην πρώτη γραμμή, το οποίο δηλώνει την επιθυμητή τιμή της θεμελιώδης συχνότητας ταλάντωσης της χορδής Μιb μπάσο (Eb). Επίσης, στην δεύτερη γραμμή εμφανίζεται το μήνυμα (Hz Note) σε συγκεκριμένες θέσεις, ώστε κατά την διάρκεια του κουρδίσματος να εκτυπώνεται η πραγματική συχνότητα πριν το (Hz) και η νότα στην οποία αντιστοιχεί η συχνότητα ταλάντωσης της χορδής πριν το (Note). Έπειτα, ακολουθούν κάποιες αρχικοποιήσεις μεταβλητών πριν την έναρξη της διαδικασίας κουρδίσματος της χορδής Μιb μπάσο (Eb). Πιο συγκεκριμένα, ο μετρητής (k) αρχικοποιείτε με την τιμή είκοσι ένα, ενώ στις σημαίες (flags) με ονομασίες (string_6, string_5, string_4, string_3, string_2 & string_1) γίνεται εναπόθεση της τιμής μηδέν.

Μετά την ολοκλήρωση των αρχικοποιήσεων, εφαρμόζεται ο βρόγχος επανάληψης (while()) ο οποίος στο μπλοκ εντολών του περιλαμβάνει τις απαραίτητες εντολές για το κούρδισμα της χορδής Μιb μπάσο (Eb). Για την εφαρμογή του βρόγχου επανάληψης (while), θα πρέπει εντός των παρενθέσεων να τοποθετηθεί μία συνθήκη. Όσο η συνθήκη αυτή ικανοποιείται, τόσο θα συνεχίσει να εκτελείται το μπλοκ εντολών του βρόγχου επανάληψης (while). Στην συγκεκριμένη περίπτωση, εντός των παρενθέσεων βρίσκεται η συνθήκη (string_6 != 1) η οποία σημαίνει ότι όσο η σημαία (flag) με όνομα (string_6) δεν είναι ίση με ένα, τόσο θα συνεχίζει να εκτελείται το μπλοκ εντολών του βρόγχου επανάληψης. Πιο συγκεκριμένα, όσο η σημαία (flag) με όνομα (string_6) είναι μηδέν αυτό σημαίνει ότι η χορδή Μιb μπάσο (Eb) είναι ξεκούρδιστη. Μόλις όμως η σημαία (flag) με όνομα (string_6) γίνει ένα, τότε αυτό σημαίνει ότι η χορδή Μιb μπάσο (Eb) έχει κουρδιστεί και το πρόγραμμα μπορεί να συνεχίσει στο κούρδισμα της επόμενης χορδής. Επειδή όμως, η συγκεκριμένη σημαία (flag) έχει αρχικοποιηθεί με την τιμή μηδέν, το μπλοκ εντολών του βρόγχου επανάληψης (while) θα εκτελεστεί μία φορά τουλάχιστον.

Εντός λοιπόν του μπλοκ εντολών του βρόγχου επανάληψης (while), εκτελούνται οι εντολές οι οποίες είναι απαραίτητες για το κούρδισμα της χορδής Μιb μπάσο (Eb). Πιο συγκεκριμένα, υπάρχει μία συνθήκη ελέγχου (if), η οποία ελέγχει την τιμή του μετρητή (k) αν είναι μεγαλύτερη από είκοσι. Ο μετρητής (k) έχει αρχικοποιηθεί πριν τον βρόγχο επανάληψης (while) με την τιμή είκοσι ένα, με αποτέλεσμα να εκτελείται το μπλοκ εντολών της συνθήκης ελέγχου (if). Πιο συγκεκριμένα, στο μπλοκ εντολών της συνθήκης (if) εκτελείτε πρώτα η εντολή pinMode(39, OUTPUT), η οποία καθορίζει τον ψηφιακό ακροδέκτη τριάντα εννιά ως έξοδο. Έπειτα, εκτελείται η εντολή digitalWrite(39, LOW), όπου ο ψηφιακός ακροδέκτης τριάντα εννιά μεταβαίνει σε κατάσταση χαμηλού δυναμικού για 10msec, διότι αμέσως μετά εκτελείται η εντολή χρονικής καθυστέρησης delay(10). Μετά εκτελείται η εντολή pinMode(39, INPUT), η οποία καθορίζει τον ψηφιακό ακροδέκτη τριάντα εννιά ως είσοδο. Ύστερα, το μπλοκ εντολών της συνθήκης ελέγχου (if) κλείνει με την εντολή (k = 0), όπου γίνεται μηδενισμός του μετρητή (k). Σκοπός της

συγκεκριμένης συνθήκης ελέγχου, είναι η γείωση (GND) του σήματος που προέρχεται από το μικρόφωνο επαφής ανά τακτά χρονικά διαστήματα, ώστε να μην υπάρχουν παράσιτα τα οποία παραμορφώνουν το σήμα και οδηγούν σε λανθασμένους υπολογισμούς της θεμελιώδης συχνότητας ταλάντωσης.

Στην συνέχεια, μετά την συνθήκη ελέγχου (if), γίνεται κλήση της συνάρτησης FREQ_DETECTION(), η οποία ξεκινά την διαδικασία δειγματοληψίας του σήματος από το μικρόφωνο επαφής και αναλύεται σε επόμενη ενότητα. Η διαδικασία δειγματοληψίας, διαρκεί για 0.5sec εξαιτίας της εντολής καθυστέρησης delay(500) που εκτελείται αμέσως μετά. Ύστερα, γίνεται κλήση της συνάρτησης FREQ_DETECTION1(), η οποία αναλύεται σε επόμενη ενότητα και έχει σκοπό τόσο τον υπολογισμό της θεμελιώδης συχνότητας ταλάντωσης της χορδής όσο και την εύρεση της νότας στην οποία αντιστοιχεί η συγκεκριμένη συχνότητα. Έπειτα, γίνεται κλήση της συνάρτησης FREQ_DETECTION_STOP() η οποία αναλύεται και αυτή σε επόμενη ενότητα και αποσκοπεί στην παύση της δειγματοληψίας του σήματος από το μικρόφωνο επαφής απενεργοποιώντας τον αναλογικό-ψηφιακό μετατροπέα.

Μετά λοιπόν την ολοκλήρωση της δειγματοληψίας και αφού έχει υπολογιστεί η θεμελιώδης συχνότητα ταλάντωσης της χορδής, εμφανίζεται στην σειριακή οθόνη το μήνυμα (Eb String) και η τιμή της συχνότητας που έχει υπολογιστεί. Αυτό επιτυγχάνεται με την χρήση της εντολής Serial.println(). Στην συνέχεια, χρησιμοποιούνται οι εντολές lcd.setCursor() και lcd.print(" ") η οποίες έχουν σκοπό την μετακίνηση του κέρσορα στην πρώτη στήλη της δεύτερης γραμμής της LCD οθόνης, το σβήσιμο της παλιάς τιμής της συχνότητας και την εμφάνιση της πραγματικής τιμής της συχνότητας που υπολογίστηκε. Με αυτόν τον τρόπο, ο χρήστης μπορεί να δει στην LCD οθόνη την πραγματική τιμή της συχνότητας ταλάντωσης της χορδής Μιb μπάσο (Eb). Αφότου ολοκληρωθεί η εμφάνιση της συχνότητας στην LCD οθόνη, εκτελείται η εντολή (k++), η οποία έχει σκοπό την αύξηση της τιμής του μετρητή (k) κατά μία μονάδα.

Εν συνεχεία, εφαρμόζεται η τεχνική ελέγχου (if – else if – else), με σκοπό τον έλεγχο της συχνότητας ταλάντωσης της χορδής. Πιο συγκεκριμένα, με την εντολή ελέγχου if (frequency < 77.68 && frequency > 65 && frequency > 0), ελέγχεται αν η τιμή της μεταβλητής (frequency) είναι μικρότερη από 77.68Hz και μεγαλύτερη από 65Hz και μεγαλύτερη από 0Hz. Επομένως, αν ικανοποιείται αυτή η συνθήκη σημαίνει ότι η συχνότητα ταλάντωσης της χορδής Μιb μπάσο (Eb) είναι χαμηλότερη από την επιθυμητή συχνότητα και για να διορθωθεί θα πρέπει να εκτελεστεί το μπλοκ εντολών της συνθήκης ελέγχου (if). Εντός λοιπόν του μπλοκ εντολών, γίνεται μηδενισμός της μεταβλητής (frequency) και ύστερα εκτελείται ένας βρόγχος επανάληψης (for). Ο βρόγχος επανάληψης (for) επαναλαμβάνει το μπλοκ εντολών του έντεκα φορές. Εντός του μπλοκ εντολών του βρόγχου επανάληψης (for) γίνεται κλήση της συνάρτησης (Counterclockwise_EStep_Motor ()). Η συνάρτηση αυτή αναλύεται σε επόμενη ενότητα και έχει σκοπό την αντί-ωρολογιακή περιστροφή του άξονα του βηματικού κινητήρα και κατά συνέπεια την αντί-ωρολογιακή περιστροφή του κλειδιού της χορδής Μιb μπάσο (Eb). Μετά την ολοκλήρωση του βρόγχου επανάληψης (for), εκτελείται η εντολή delay(2) η οποία δημιουργεί μια χρονική καθυστέρηση των 2msec. Το μπλοκ εντολών της συνθήκης ελέγχου (if) ολοκληρώνεται με την κλήση της συνάρτησης (EStep_Motor_Stop ()), η οποία αναλύεται σε επόμενο κεφάλαιο και σκοπός της είναι να απενεργοποιήσει τον βηματικό κινητήρα. Στην περίπτωση όμως που δεν ικανοποιείται η συνθήκη ελέγχου (if), τότε εκτελείται η εντολή ελέγχου else if (frequency > 77.88 && frequency < 103 &&

frequency > 0), όπου ελέγχεται αν η τιμή της μεταβλητής (frequency) είναι μεγαλύτερη από 77.88Hz και μικρότερη από 103Hz και μεγαλύτερη από 0Hz. Συνεπώς, αν ικανοποιείται αυτή η συνθήκη σημαίνει ότι η συχνότητα ταλάντωσης της χορδής Μιb μπάσο (Eb) είναι υψηλότερη από την επιθυμητή συχνότητα και για να διορθωθεί θα πρέπει να εκτελεστεί το μπλοκ εντολών της συνθήκης ελέγχου (else if). Εντός λοιπόν του μπλοκ εντολών, γίνεται μηδενισμός της μεταβλητής (frequency) και ύστερα εκτελείται ένας βρόγχος επανάληψης (for). Εντός του μπλοκ εντολών του βρόγχου επανάληψης (for) γίνεται κλήση της συνάρτησης (Clockwise_EStep_Motor ()), η οποία αναλύεται σε επόμενη ενότητα. Σκοπός της συνάρτησης (Clockwise_EStep_Motor ()), είναι να περιστρέψει τον άξονα του βηματικού κινητήρα ωρολογιακά και κατά συνέπεια να περιστρέψει ωρολογιακά και το κλειδί της χορδής Μιb μπάσο (Eb). Αφότου ολοκληρωθεί ο βρόγχος επανάληψης (for), εκτελείται η εντολή delay(2) η οποία δημιουργεί μια χρονική καθυστέρηση των 2msec. Η τελευταία εντολή που εκτελείται στο μπλοκ εντολών της συνθήκης ελέγχου (else if) αφορά την κλήση της συνάρτησης (EStep_Motor_Stop ()), η οποία έχει σκοπό να απενεργοποιήσει τον βηματικό κινητήρα. Αν όμως δεν ικανοποιείται η προηγούμενη συνθήκη (else if), τότε εκτελείται η εντολή ελέγχου else if (frequency >=77.68 && frequency <=77.88), η οποία ελέγχει αν η τιμή της μεταβλητής (frequency) είναι μεγαλύτερη ή ίση από 77.68Hz και μικρότερη ή ίση από 77.88Hz. Επομένως, αν ικανοποιείται αυτή η συνθήκη τότε η πραγματική συχνότητα ταλάντωσης της χορδής Μιb μπάσο (Eb) βρίσκεται εντός του εύρους των επιθυμητών συχνοτήτων της χορδής και εκτελείται το αντίστοιχο μπλοκ εντολών. Εντός λοιπόν του συγκεκριμένου μπλοκ εντολών, αποθηκεύεται στην σημαία (flag) με όνομα (string_6) η τιμή ένα. Με αυτόν τον τρόπο υποδηλώνεται στο πρόγραμμα η ολοκλήρωση του κουρδίσματος της χορδής Μιb μπάσο (Eb). Ύστερα, εκτελείται η εντολή (k = 21), όπου πραγματοποιείται αρχικοποίηση του μετρητή (k) με την τιμή είκοσι ένα και αμέσως μετά, πραγματοποιείται μηδενισμός της μεταβλητής (frequency) μέσω της εντολής (frequency = 0.00). Έπειτα, εκτελούνται οι εντολές lcd.clear(), lcd.setCursor() και lcd.print(“ “) με την κατάλληλη σειρά, ώστε να εμφανιστεί το μήνυμα (Hit A string) στην LCD οθόνη. Το μήνυμα αυτό παραμένει στην LCD οθόνη για 1sec εξαιτίας της εντολής (delay1000) και σκοπό έχει να προτρέψει τον χρήστη να διεγείρει την επόμενη χορδή η οποία είναι η Λαb (Ab). Εν συνεχεία, εκτελούνται ξανά οι εντολές lcd.clear(), lcd.setCursor() και lcd.print(“ “) με την κατάλληλη σειρά ώστε, να εμφανιστεί στην πρώτη γραμμή της LCD οθόνης το μήνυμα (Ab -> 103.83 Hz), το οποίο υποδηλώνει την επιθυμητή τιμή της συχνότητας της χορδής Λαb (Ab) και στην δεύτερη γραμμή το μήνυμα (Hz Note) σε συγκεκριμένες θέσης. Σκοπός του μηνύματος της δεύτερης γραμμής είναι να εκτυπώνεται η πραγματική συχνότητα πριν το (Hz) και η νότα στην οποία αντιστοιχεί η συχνότητα ταλάντωσης της χορδής πριν το (Note) κατά την διάρκεια κουρδίσματος της χορδής Λαb (Ab). Αν όμως, δεν ικανοποιείται καμία συνθήκη από τις εντολές ελέγχου (if-else if), τότε εκτελείται το μπλοκ εντολών της συνθήκης (else). Πιο συγκεκριμένα, το μπλοκ εντολών της συνθήκης (else) περιλαμβάνει μία μόνο εντολή η οποία μηδενίζει την τιμή της μεταβλητής (frequency). Με αυτό τον τρόπο, αντιμετωπίζονται οι συχνότητες που μπορούν να προκύψουν από την ταυτόχρονη διέγερση δύο χορδών ή από την διέγερση λανθασμένης χορδής. Σε αυτό το σημείο ολοκληρώνεται το μπλοκ εντολών του βρόγχου επανάληψης (while), το οποίο έχει να κάνει με το κούρδισμα της χορδής Μιb μπάσο (Eb). Η διαδικασία που μόλις περιγράφτηκε, εκτελείτε ξανά μέχρι να πραγματοποιηθεί το κούρδισμα της χορδής Μιb μπάσο (Eb) εντός του εύρος των συχνοτήτων ταλάντωσης που έχει οριστεί για την συγκεκριμένη χορδή.

Με την ολοκλήρωση του κουρδίσματος της χορδής Μιb μπάσο (Eb), σειρά έχει το κούρδισμα της χορδής Λαb (Ab). Αυτό επιτυγχάνεται με την χρήση του βρόγχου επανάληψης while(), ο οποίος στο μπλοκ εντολών του περιλαμβάνει τις απαραίτητες εντολές για το κούρδισμα της χορδής Λαb (Ab). Η συνθήκη που βρίσκεται εντός των παρενθέσεων του βρόγχου επανάληψης (while) είναι η (string_5 != 1) η οποία σημαίνει ότι όσο η σημαία (flag) με όνομα (string_5) δεν είναι ίση με ένα, τόσο θα συνεχίζει να εκτελείται το μπλοκ εντολών του βρόγχου επανάληψης. Πιο συγκεκριμένα, όσο η σημαία (flag) με όνομα (string_5) είναι μηδέν αυτό σημαίνει ότι η χορδή Λαb (Ab) είναι ξεκούρδιστη. Μόλις όμως η σημαία (flag) με όνομα (string_5) γίνει ένα, τότε αυτό σημαίνει ότι η χορδή Λαb (Ab) έχει κουρδιστεί και το πρόγραμμα μπορεί να συνεχίσει στο κούρδισμα της επόμενης χορδής. Επειδή όμως η συγκεκριμένη σημαία (flag) έχει αρχικοποιηθεί με την τιμή μηδέν, το μπλοκ εντολών του βρόγχου επανάληψης (while) θα εκτελεστεί μία φορά τουλάχιστον.

Ουσιαστικά, εντός του μπλοκ εντολών του βρόγχου επανάληψης (while), εκτελούνται οι εντολές οι οποίες είναι απαραίτητες για το κούρδισμα της χορδής Λαb (Ab). Όπως αναφέρθηκε και προηγουμένως, υπάρχει μία συνθήκη ελέγχου (if), η οποία ελέγχει την τιμή του μετρητή (k) αν είναι μεγαλύτερη από είκοσι. Ο μετρητής (k) έχει αρχικοποιηθεί στον προηγούμενο βρόγχο επανάληψης (while) με την τιμή είκοσι ένα, με αποτέλεσμα να εκτελείται το μπλοκ εντολών της συνθήκης ελέγχου (if). Ουσιαστικά, στο μπλοκ εντολών της συνθήκης (if) εκτελείτε πρώτα η εντολή pinMode(39, OUTPUT), η οποία καθορίζει τον ψηφιακό ακροδέκτη τριάντα εννιά ως έξοδο. Έπειτα, εκτελείται η εντολή digitalWrite(39, LOW), όπου ο ψηφιακός ακροδέκτης τριάντα εννιά μεταβαίνει σε κατάσταση χαμηλού δυναμικού για 10msec, διότι αμέσως μετά εκτελείται η εντολή χρονικής καθυστέρησης delay(10). Ύστερα, εκτελείται η εντολή pinMode(39, INPUT), η οποία καθορίζει τον ψηφιακό ακροδέκτη τριάντα εννιά ως είσοδο. Το μπλοκ εντολών της συνθήκης ελέγχου (if) ολοκληρώνεται με την εντολή (k = 0), όπου γίνεται μηδενισμός του μετρητή (k). Σκοπός της συγκεκριμένης συνθήκης ελέγχου είναι η γείωση (GND) του σήματος που προέρχεται από το μικρόφωνο επαφής ανά τακτά χρονικά διαστήματα, ώστε να μην υπάρχουν παράσιτα τα οποία παραμορφώνουν το σήμα και οδηγούν σε λανθασμένους υπολογισμούς της συχνότητας ταλάντωσης της χορδής.

Αφότου ολοκληρωθεί η εντολή της συνθήκης ελέγχου (if), γίνεται κλήση της συνάρτησης FREQ_DETECTION(), η οποία αναλύεται σε επόμενη ενότητα και αποσκοπεί στην εκκίνηση της διαδικασίας της δειγματοληψίας του σήματος από το μικρόφωνο επαφής. Η διαδικασία δειγματοληψίας, διαρκεί για 0.5sec εξαιτίας της εντολής καθυστέρησης delay(500) που εκτελείται αμέσως μετά. Ύστερα, γίνεται κλήση της συνάρτησης FREQ_DETECTION1(), η οποία έχει σκοπό τόσο τον υπολογισμό της θεμελιώδης συχνότητας ταλάντωσης της χορδής όσο και την εύρεση της νότας στην οποία αντιστοιχεί η συγκεκριμένη συχνότητα και η ανάλυσή της γίνεται σε επόμενη ενότητα. Έπειτα, γίνεται κλήση της συνάρτησης FREQ_DETECTION_STOP() η οποία αναλύεται και αυτή σε επόμενη ενότητα και αποσκοπεί στην παύση της δειγματοληψίας του σήματος από το μικρόφωνο επαφής απενεργοποιώντας τον αναλογικό-ψηφιακό μετατροπέα.

Εν συνεχεία, αφού έχει υπολογιστεί η θεμελιώδης συχνότητα ταλάντωσης της χορδής, εμφανίζεται στην σειριακή οθόνη το μήνυμα (Ab String) και η τιμή της συχνότητας που έχει υπολογιστεί. Αυτό επιτυγχάνεται με την χρήση της εντολής Serial.println(). Στην συνέχεια, χρησιμοποιούνται οι εντολές lcd.setCursor() και lcd.print(“ “) η οποίες έχουν σκοπό την μετακίνηση του κέρσορα στην πρώτη στήλη της δεύτερης

γραμμής της LCD οθόνης, το σβήσιμο της παλιάς τιμής της συχνότητας και την εμφάνιση της πραγματικής τιμής της συχνότητας που υπολογίστηκε. Με αυτόν τον τρόπο, ο χρήστης μπορεί να δει στην LCD οθόνη την πραγματική τιμή της συχνότητας ταλάντωσης της χορδής Λαb (Ab). Αφότου ολοκληρωθεί η εμφάνιση της συχνότητας στην LCD οθόνη, εκτελείται η εντολή (k++), η οποία έχει σκοπό την αύξηση της τιμής του μετρητή (k) κατά μία μονάδα.

Έπειτα, εφαρμόζεται η τεχνική ελέγχου (if – else if – else), η οποία αποσκοπεί στον έλεγχο της συχνότητας ταλάντωσης της χορδής. Ουσιαστικά, με την εκτέλεση της εντολής ελέγχου if (frequency < 103.67 && frequency > 92 && frequency > 0), ελέγχεται αν η τιμή της μεταβλητής (frequency) είναι μικρότερη από 103.67Hz και μεγαλύτερη από 92Hz και μεγαλύτερη από 0Hz. Συνεπώς, αν ικανοποιείται αυτή η συνθήκη σημαίνει ότι η συχνότητα ταλάντωσης της χορδής Λαb (Ab) είναι χαμηλότερη από την επιθυμητή συχνότητα και για να διορθωθεί θα πρέπει να εκτελεστεί το μπλοκ εντολών της συνθήκης ελέγχου (if). Εντός λοιπόν του μπλοκ εντολών, γίνεται μηδενισμός της μεταβλητής (frequency) και ύστερα εκτελείται ένας βρόγχος επανάληψης (for) ο οποίος επαναλαμβάνει το μπλοκ εντολών του είκοσι ένα φορές. Όσον αφορά το μπλοκ εντολών του βρόγχου επανάληψης (for), γίνεται κλήση της συνάρτησης (Counterclockwise_AStep_Motor ()). Η συνάρτηση αυτή αναλύεται σε επόμενη ενότητα και έχει σκοπό την αντί-ωρολογιακή περιστροφή του άξονα του βηματικού κινητήρα και κατά συνέπεια την αντί-ωρολογιακή περιστροφή του κλειδιού της χορδής Λαb (Ab). Μετά την ολοκλήρωση του βρόγχου επανάληψης (for), εκτελείται η εντολή delay(2) η οποία δημιουργεί μια χρονική καθυστέρηση των 2msec. Το μπλοκ εντολών της συνθήκης ελέγχου (if) ολοκληρώνεται με την κλήση της συνάρτησης (AStep_Motor_Stop ()), η οποία αναλύεται σε επόμενο κεφάλαιο και σκοπός της είναι να απενεργοποιήσει τον βηματικό κινητήρα. Αν όμως δεν ικανοποιείται η συνθήκη ελέγχου (if), τότε εκτελείται η εντολή ελέγχου else if (frequency > 103.96 && frequency < 130 && frequency > 0), όπου ελέγχεται αν η τιμή της μεταβλητής (frequency) είναι μεγαλύτερη από 103.96Hz και μικρότερη από 130Hz και μεγαλύτερη από 0Hz. Συνεπώς, αν ικανοποιείται αυτή η συνθήκη σημαίνει ότι η συχνότητα ταλάντωσης της χορδής Λαb (Ab) είναι υψηλότερη από την επιθυμητή συχνότητα και για να διορθωθεί θα πρέπει να εκτελεστεί το μπλοκ εντολών της συνθήκης ελέγχου (else if). Ουσιαστικά, εντός του μπλοκ εντολών γίνεται μηδενισμός της μεταβλητής (frequency) και ύστερα εκτελείται ένας βρόγχος επανάληψης (for). Εντός του μπλοκ εντολών του βρόγχου επανάληψης (for) γίνεται κλήση της συνάρτησης (Clockwise_AStep_Motor ()), της οποίας η ανάλυση πραγματοποιείται σε επόμενη ενότητα. Η κλήση της συνάρτησης (Clockwise_AStep_Motor ()), αποσκοπεί στην ωρολογιακή περιστροφή του άξονα του βηματικού κινητήρα και κατά συνέπεια στην ωρολογιακή περιστροφή του κλειδιού της χορδής Λαb (Ab). Μετά την ολοκλήρωση του βρόγχου επανάληψης (for), εκτελείται η εντολή delay(2) η οποία δημιουργεί μια χρονική καθυστέρηση των 2msec. Η τελευταία εντολή που εκτελείται στο μπλοκ εντολών της συνθήκης ελέγχου (else if) αφορά την κλήση της συνάρτησης (AStep_Motor_Stop ()), η οποία αποσκοπεί στην απενεργοποίηση του βηματικού κινητήρα. Σε περίπτωση όμως που δεν ικανοποιείται η προηγούμενη συνθήκη (else if), τότε εκτελείται η εντολή ελέγχου else if (frequency >=103.66 && frequency <=103.96), η οποία ελέγχει αν η τιμή της μεταβλητής (frequency) είναι μεγαλύτερη ή ίση από 103.66Hz και μικρότερη ή ίση από 103.96Hz. Συνεπώς, αν ικανοποιείται αυτή η συνθήκη τότε η πραγματική συχνότητα ταλάντωσης της χορδής Λαb (Ab) βρίσκεται εντός του εύρους των επιθυμητών συχνοτήτων της χορδής και εκτελείται το αντίστοιχο μπλοκ εντολών. Πιο συγκεκριμένα,

εντός του συγκεκριμένου μπλοκ εντολών, αποθηκεύεται στην σημαία (flag) με όνομα (string_5) η τιμή ένα. Με αυτόν τον τρόπο υποδηλώνεται στο πρόγραμμα η ολοκλήρωση του κουρδίσματος της χορδής Λαb (Ab). Έπειτα, εκτελείται η εντολή (k = 21), όπου πραγματοποιείται αρχικοποίηση του μετρητή (k) με την τιμή είκοσι ένα και αμέσως μετά, εκτελείται η εντολή (frequency = 0.00) η οποία μηδενίζει την τιμή της μεταβλητής (frequency). Ύστερα, πραγματοποιείται εκτέλεση των εντολών lcd.clear(), lcd.setCursor() και lcd.print(“ “) με την κατάλληλη σειρά, ώστε να εμφανιστεί το μήνυμα (Hit Db string) στην LCD οθόνη. Το μήνυμα αυτό παραμένει στην LCD οθόνη για 1sec εξαιτίας της εντολής (delay1000) και έχει σκοπό να προτρέψει τον χρήστη να διεγείρει την επόμενη χορδή η οποία είναι η Ρeb (Db). Στην συνεχεία, εκτελούνται ξανά οι εντολές lcd.clear(), lcd.setCursor() και lcd.print(“ “) με την κατάλληλη σειρά ώστε, να εμφανιστεί στην πρώτη γραμμή της LCD οθόνης το μήνυμα (Db -> 138.59 Hz), το οποίο υποδηλώνει την επιθυμητή τιμή της συχνότητας της χορδής Ρeb (Db) και στην δεύτερη γραμμή το μήνυμα (Hz Note) σε συγκεκριμένες θέσης. Πιο συγκεκριμένα, το μήνυμα της δεύτερης γραμμής αποσκοπεί στην εκτύπωση της πραγματική συχνότητα πριν το (Hz) και η νότα στην οποία αντιστοιχεί η συχνότητα ταλάντωσης της χορδής πριν το (Note) κατά την διάρκεια κουρδίσματος της χορδής Ρeb (Db). Στην περίπτωση όμως που δεν ικανοποιείται καμία συνθήκη από τις εντολές ελέγχου (if-else if), τότε εκτελείται το μπλοκ εντολών της συνθήκης (else). Ουσιαστικά, το μπλοκ εντολών της συνθήκης (else) αποτελείται από μία μόνο εντολή η οποία μηδενίζει την τιμή της μεταβλητής (frequency). Με αυτό τον τρόπο, αντιμετωπίζονται οι συχνότητες που μπορούν να προκύψουν από την ταυτόχρονη διέγερση δύο χορδών ή από την διέγερση λανθασμένης χορδής. Σε αυτό το σημείο ολοκληρώνεται το μπλοκ εντολών του βρόγχου επανάληψης (while), το οποίο έχει να κάνει με το κούρδισμα της χορδής Λαb (Ab). Η διαδικασία που μόλις περιγράφτηκε, επαναλαμβάνεται μέχρι να επιτευχθεί το κούρδισμα της χορδής Λαb (Ab) εντός του εύρος των συχνοτήτων ταλάντωσης που έχει οριστεί για την συγκεκριμένη χορδή.

Αφού ολοκληρωθεί το κούρδισμα της χορδής Λαb (Ab), το πρόγραμμα συνεχίζει στην πραγματοποίηση του κουρδίσματος της χορδής Ρeb (Db). Αυτό επιτυγχάνεται με την χρήση του βρόγχου επανάληψης while(), ο οποίος στο μπλοκ εντολών του περιλαμβάνει τις απαραίτητες εντολές για το κούρδισμα της χορδής Ρeb (Db). Επιπλέον, η συνθήκη που βρίσκεται εντός των παρενθέσεων του βρόγχου επανάληψης (while) είναι η (string_4 != 1), η οποία δηλώνει ότι όσο η σημαία (flag) με όνομα (string_4) δεν είναι ίση με ένα, τόσο θα συνεχίζει να εκτελείται το μπλοκ εντολών του βρόγχου επανάληψης. Πιο συγκεκριμένα, όσο η σημαία (flag) με όνομα (string_4) είναι μηδέν αυτό σημαίνει ότι η χορδή Ρeb (Db) είναι ξεκούρδιστη. Μόλις όμως, η σημαία (flag) με όνομα (string_4) γίνει ένα, τότε αυτό σημαίνει ότι η χορδή Ρeb (Db) έχει κουρδιστεί και το πρόγραμμα μπορεί να συνεχίσει στο κούρδισμα της επόμενης χορδής. Επειδή όμως η συγκεκριμένη σημαία (flag) έχει αρχικοποιηθεί με την τιμή μηδέν, το μπλοκ εντολών του βρόγχου επανάληψης (while) θα εκτελεστεί μία φορά τουλάχιστον.

Εντός λοιπόν του μπλοκ εντολών του βρόγχου επανάληψης (while), εκτελούνται οι εντολές οι οποίες είναι απαραίτητες για το κούρδισμα της χορδής Ρeb (Db). Πιο συγκεκριμένα, υπάρχει μία συνθήκη ελέγχου (if), η οποία ελέγχει την τιμή του μετρητή (k) αν είναι μεγαλύτερη από είκοσι. Ο μετρητής (k) έχει αρχικοποιηθεί στον προηγούμενο βρόγχο επανάληψης (while) με την τιμή είκοσι ένα, με αποτέλεσμα να εκτελείται το μπλοκ εντολών της συνθήκης ελέγχου (if). Πιο συγκεκριμένα, εντός του μπλοκ εντολών της συνθήκης (if), εκτελείτε πρώτα η εντολή pinMode(39, OUTPUT), η οποία καθορίζει

τον ψηφιακό ακροδέκτη τριάντα εννιά ως έξοδο. Έπειτα, εκτελείται η εντολή digitalWrite(39, LOW), όπου ο ψηφιακός ακροδέκτης τριάντα εννιά μεταβαίνει σε κατάσταση χαμηλού δυναμικού για 10msec, διότι αμέσως μετά εκτελείται η εντολή χρονικής καθυστέρησης delay(10). Μετά εκτελείται η εντολή pinMode(39, INPUT), η οποία καθορίζει τον ψηφιακό ακροδέκτη τριάντα εννιά ως είσοδο. Ύστερα, το μπλοκ εντολών της συνθήκης ελέγχου (if) κλείνει με την εντολή (k = 0), όπου γίνεται μηδενισμός του μετρητή (k). Σκοπός της συγκεκριμένης συνθήκης ελέγχου, είναι η γείωση (GND) του σήματος που προέρχεται από το μικρόφωνο επαφής ανά τακτά χρονικά διαστήματα, ώστε να μην υπάρχουν παράσιτα τα οποία παραμορφώνουν το σήμα και οδηγούν σε λανθασμένους υπολογισμούς της θεμελιώδης συχνότητας ταλάντωσης.

Στην συνέχεια, μετά την συνθήκη ελέγχου (if), γίνεται κλήση της συνάρτησης FREQ_DETECTION(), η οποία ξεκινά την διαδικασία δειγματοληψίας του σήματος από το μικρόφωνο επαφής και αναλύεται σε επόμενη ενότητα. Η διαδικασία δειγματοληψίας, διαρκεί για 0.5sec εξαιτίας της εντολής καθυστέρησης delay(500) που εκτελείται αμέσως μετά. Ύστερα, γίνεται κλήση της συνάρτησης FREQ_DETECTION1(), η οποία αναλύεται σε επόμενη ενότητα και έχει σκοπό τόσο τον υπολογισμό της θεμελιώδης συχνότητας ταλάντωσης της χορδής όσο και την εύρεση της νότας στην οποία αντιστοιχεί η συγκεκριμένη συχνότητα. Έπειτα, γίνεται κλήση της συνάρτησης FREQ_DETECTION_STOP() η οποία αναλύεται και αυτή σε επόμενη ενότητα και αποσκοπεί στην παύση της δειγματοληψίας του σήματος από το μικρόφωνο επαφής απενεργοποιώντας τον αναλογικό-ψηφιακό μετατροπέα.

Μετά λοιπόν την ολοκλήρωση της δειγματοληψίας και αφού έχει υπολογιστεί η θεμελιώδης συχνότητα ταλάντωσης της χορδής, εμφανίζεται στην σειριακή οθόνη το μήνυμα (Db String) και η τιμή της συχνότητας που έχει υπολογιστεί. Αυτό επιτυγχάνεται με την χρήση της εντολής Serial.println(). Στην συνέχεια, χρησιμοποιούνται οι εντολές lcd.setCursor() και lcd.print(“ “) η οποίες έχουν σκοπό την μετακίνηση του κέρσορα στην πρώτη στήλη της δεύτερης γραμμής της LCD οθόνης, το σβήσιμο της παλιάς τιμής της συχνότητας και την εμφάνιση της πραγματικής τιμής της θεμελιώδης συχνότητας ταλάντωσης της χορδής που υπολογίστηκε. Με αυτόν τον τρόπο, ο χρήστης μπορεί να δει στην LCD οθόνη την πραγματική τιμή της συχνότητας ταλάντωσης της χορδής Pεb (Db). Αφότου ολοκληρωθεί η εμφάνιση της συχνότητας στην LCD οθόνη, εκτελείται η εντολή (k++), η οποία έχει σκοπό την αύξηση της τιμής του μετρητή (k) κατά μία μονάδα.

Εν συνεχεία, εφαρμόζεται η τεχνική ελέγχου (if – else if – else), με σκοπό τον έλεγχο της συχνότητας ταλάντωσης της χορδής. Πιο συγκεκριμένα, με την εντολή ελέγχου if (frequency < 138.35 && frequency > 123 && frequency > 0), ελέγχεται αν η τιμή της μεταβλητής (frequency) είναι μικρότερη από 138.35Hz και μεγαλύτερη από 123Hz και μεγαλύτερη από 0Hz. Επομένως, αν ικανοποιείται αυτή η συνθήκη σημαίνει ότι η συχνότητα ταλάντωσης της χορδής Pεb (Db) είναι χαμηλότερη από την επιθυμητή συχνότητα και για να διορθωθεί θα πρέπει να εκτελεστεί το μπλοκ εντολών της συνθήκης ελέγχου (if). Εντός λοιπόν του μπλοκ εντολών, γίνεται μηδενισμός της μεταβλητής (frequency) και ύστερα εκτελείται ένας βρόγχος επανάληψης (for). Ο βρόγχος επανάληψης (for) επαναλαμβάνει το μπλοκ εντολών του τριάντα μία φορές. Εντός του μπλοκ εντολών του βρόγχου επανάληψης (for) γίνεται κλήση της συνάρτησης (Counterclockwise_DStep_Motor ()). Η συνάρτηση αυτή αναλύεται σε επόμενη ενότητα και έχει σκοπό την αντί-ωρολογιακή περιστροφή του άξονα του βηματικού κινητήρα και κατά συνέπεια την αντί-ωρολογιακή περιστροφή του κλειδιού της χορδής Pεb (Db). Μετά

την ολοκλήρωση του βρόγχου επανάληψης (for), εκτελείται η εντολή delay(2) η οποία δημιουργεί μια χρονική καθυστέρηση των 2msec. Το μπλοκ εντολών της συνθήκης ελέγχου (if) ολοκληρώνεται με την κλήση της συνάρτησης (DStep_Motor_Stop ()), η οποία αναλύεται σε επόμενο κεφάλαιο και σκοπός της είναι να απενεργοποιήσει τον βηματικό κινητήρα. Στην περίπτωση όμως, που δεν ικανοποιείται η συνθήκη ελέγχου (if), τότε εκτελείται η εντολή ελέγχου else if (frequency > 138.85 && frequency < 174 && frequency > 0), όπου ελέγχεται αν η τιμή της μεταβλητής (frequency) είναι μεγαλύτερη από 138.85Hz και μικρότερη από 174Hz και μεγαλύτερη από 0Hz. Συνεπώς, αν ικανοποιείται αυτή η συνθήκη σημαίνει ότι η συχνότητα ταλάντωσης της χορδής Ρεb (Db) είναι υψηλότερη από την επιθυμητή συχνότητα και για να διορθωθεί θα πρέπει να εκτελεστεί το μπλοκ εντολών της συνθήκης ελέγχου (else if). Εντός λοιπόν του μπλοκ εντολών, γίνεται μηδενισμός της μεταβλητής (frequency) και ύστερα εκτελείται ένας βρόγχος επανάληψης (for). Εντός του μπλοκ εντολών του βρόγχου επανάληψης (for) γίνεται κλήση της συνάρτησης (Clockwise_DStep_Motor ()), η οποία αναλύεται σε επόμενη ενότητα. Σκοπός της συνάρτησης (Clockwise_DStep_Motor ()), είναι να περιστρέψει τον άξονα του βηματικού κινητήρα ωρολογιακά και κατά συνέπεια να περιστρέψει ωρολογιακά και το κλειδί της χορδής Ρεb (Db). Αφότου ολοκληρωθεί ο βρόγχος επανάληψης (for), εκτελείται η εντολή delay(2) η οποία δημιουργεί μια χρονική καθυστέρηση των 2msec. Η τελευταία εντολή που εκτελείται στο μπλοκ εντολών της συνθήκης ελέγχου (else if) αφορά την κλήση της συνάρτησης (DStep_Motor_Stop ()), η οποία έχει σκοπό να απενεργοποιήσει τον βηματικό κινητήρα. Αν όμως δεν ικανοποιείται η προηγούμενη συνθήκη (else if), τότε εκτελείται η εντολή ελέγχου else if (frequency >=138.35 && frequency <=138.85), η οποία ελέγχει αν η τιμή της μεταβλητής (frequency) είναι μεγαλύτερη ή ίση από 138.35Hz και μικρότερη ή ίση από 138.85Hz. Επομένως, αν ικανοποιείται αυτή η συνθήκη τότε η πραγματική συχνότητα ταλάντωσης της χορδής Ρεb (Db) βρίσκεται εντός του εύρους των επιθυμητών συχνοτήτων της χορδής και εκτελείται το αντίστοιχο μπλοκ εντολών. Εντός λοιπόν του συγκεκριμένου μπλοκ εντολών, αποθηκεύεται στην σημαία (flag) με όνομα (string_4) η τιμή ένα. Με αυτόν τον τρόπο υποδηλώνεται στο πρόγραμμα η ολοκλήρωση του κουρδίσματος της χορδής Ρεb (Db). Ύστερα, εκτελείται η εντολή (k = 21), όπου πραγματοποιείται αρχικοποίηση του μετρητή (k) με την τιμή είκοσι ένα και αμέσως μετά, πραγματοποιείται μηδενισμός της μεταβλητής (frequency) μέσω της εντολής (frequency = 0.00). Έπειτα, εκτελούνται οι εντολές lcd.clear(), lcd.setCursor() και lcd.print(“ “) με την κατάλληλη σειρά, ώστε να εμφανιστεί το μήνυμα (Hit Gb string) στην LCD οθόνη. Το μήνυμα αυτό παραμένει στην LCD οθόνη για 1sec εξαιτίας της εντολής (delay1000) και σκοπό έχει να προτρέψει τον χρήστη να διεγείρει την επόμενη χορδή η οποία είναι η Σολb (Gb). Εν συνεχεία, εκτελούνται ξανά οι εντολές lcd.clear(), lcd.setCursor() και lcd.print(“ “) με την κατάλληλη σειρά ώστε, να εμφανιστεί στην πρώτη γραμμή της LCD οθόνης το μήνυμα (Gb -> 185 Hz), το οποίο υποδηλώνει την επιθυμητή τιμή της συχνότητας της χορδής Σολb (Gb) και στην δεύτερη γραμμή το μήνυμα (Hz Note) σε συγκεκριμένες θέσης. Σκοπός του μηνύματος της δεύτερης γραμμής είναι να εκτυπώνεται η πραγματική συχνότητα πριν το (Hz) και η νότα στην οποία αντιστοιχεί η συχνότητα ταλάντωσης της χορδής πριν το (Note) κατά την διάρκεια κουρδίσματος της χορδής Σολb (Gb). Αν όμως, δεν ικανοποιείται καμία συνθήκη από τις εντολές ελέγχου (if-else if), τότε εκτελείται το μπλοκ εντολών της συνθήκης (else). Πιο συγκεκριμένα, το μπλοκ εντολών της συνθήκης (else) περιλαμβάνει μία μόνο εντολή η οποία μηδενίζει την τιμή της μεταβλητής (frequency). Με αυτό τον τρόπο, αντιμετωπίζονται οι συχνότητες που μπορούν να

προκύψουν από την ταυτόχρονη διέγερση δύο χορδών ή από την διέγερση λανθασμένης χορδής. Σε αυτό το σημείο ολοκληρώνεται το μπλοκ εντολών του βρόγχου επανάληψης (while), το οποίο έχει να κάνει με το κούρδισμα της χορδής Ρεb (Db). Η διαδικασία που μόλις περιγράφτηκε, εκτελείτε ξανά μέχρι να πραγματοποιηθεί το κούρδισμα της χορδής Ρεb (Db) εντός του εύρος των συχνοτήτων ταλάντωσης που έχει οριστεί για την συγκεκριμένη χορδή.

Με την ολοκλήρωση του κουρδίσματος της χορδής Ρεb (Db), σειρά έχει το κούρδισμα της χορδής Σολb (Gb). Αυτό επιτυγχάνεται με την χρήση του βρόγχου επανάληψης while(), ο οποίος στο μπλοκ εντολών του περιλαμβάνει τις απαραίτητες εντολές για το κούρδισμα της χορδής Σολb (Gb). Η συνθήκη που βρίσκεται εντός των παρενθέσεων του βρόγχου επανάληψης (while) είναι η (string_3 != 1) η οποία σημαίνει ότι όσο η σημαία (flag) με όνομα (string_3) δεν είναι ίση με ένα, τόσο θα συνεχίζει να εκτελείται το μπλοκ εντολών του βρόγχου επανάληψης. Πιο συγκεκριμένα, όσο η σημαία (flag) με όνομα (string_3) είναι μηδέν αυτό σημαίνει ότι η χορδή Σολb (Gb) είναι ξεκούρδιστη. Μόλις όμως η σημαία (flag) με όνομα (string_3) γίνει ένα, τότε αυτό σημαίνει ότι η χορδή Σολb (Gb) έχει κουρδιστεί και το πρόγραμμα μπορεί να συνεχίσει στο κούρδισμα της επόμενης χορδής. Επειδή όμως η συγκεκριμένη σημαία (flag) έχει αρχικοποιηθεί με την τιμή μηδέν, το μπλοκ εντολών του βρόγχου επανάληψης (while) θα εκτελεστεί μία φορά τουλάχιστον.

Ουσιαστικά, εντός του μπλοκ εντολών του βρόγχου επανάληψης (while), εκτελούνται οι εντολές οι οποίες είναι απαραίτητες για το κούρδισμα της χορδής Σολb (Gb). Όπως αναφέρθηκε και προηγουμένως, υπάρχει μία συνθήκη ελέγχου (if), η οποία ελέγχει την τιμή του μετρητή (k) αν είναι μεγαλύτερη από είκοσι. Ο μετρητής (k) έχει αρχικοποιηθεί στον προηγούμενο βρόγχο επανάληψης (while) με την τιμή είκοσι ένα, με αποτέλεσμα να εκτελείται το μπλοκ εντολών της συνθήκης ελέγχου (if). Ουσιαστικά, στο μπλοκ εντολών της συνθήκης (if) εκτελείτε πρώτα η εντολή pinMode(39, OUTPUT), η οποία καθορίζει τον ψηφιακό ακροδέκτη τριάντα εννιά ως έξοδο. Έπειτα, εκτελείται η εντολή digitalWrite(39, LOW), όπου ο ψηφιακός ακροδέκτης τριάντα εννιά μεταβαίνει σε κατάσταση χαμηλού δυναμικού για 10msec, διότι αμέσως μετά εκτελείται η εντολή χρονικής καθυστέρησης delay(10). Ύστερα, εκτελείται η εντολή pinMode(39, INPUT), η οποία καθορίζει τον ψηφιακό ακροδέκτη τριάντα εννιά ως είσοδο. Το μπλοκ εντολών της συνθήκης ελέγχου (if) ολοκληρώνεται με την εντολή (k = 0), όπου γίνεται μηδενισμός του μετρητή (k). Σκοπός της συγκεκριμένης συνθήκης ελέγχου είναι η γείωση (GND) του σήματος που προέρχεται από το μικρόφωνο επαφής ανά τακτά χρονικά διαστήματα, ώστε να μην υπάρχουν παράσιτα τα οποία παραμορφώνουν το σήμα και οδηγούν σε λανθασμένους υπολογισμούς της συχνότητας ταλάντωσης της χορδής..

Αφότου ολοκληρωθεί η εντολή της συνθήκης ελέγχου (if), γίνεται κλήση της συνάρτησης FREQ_DETECTION(), η οποία αναλύεται σε επόμενη ενότητα και αποσκοπεί στην εκκίνηση της διαδικασίας της δειγματοληψίας του σήματος από το μικρόφωνο επαφής. Η διαδικασία δειγματοληψίας, διαρκεί για 0.5sec εξαιτίας της εντολής καθυστέρησης delay(500) που εκτελείται αμέσως μετά. Ύστερα, γίνεται κλήση της συνάρτησης FREQ_DETECTION1(), η οποία έχει σκοπό τόσο τον υπολογισμό της θεμελιώδης συχνότητας ταλάντωσης της χορδής όσο και την εύρεση της νότας στην οποία αντιστοιχεί η συγκεκριμένη συχνότητα και η ανάλυσή της γίνεται σε επόμενη ενότητα. Έπειτα, γίνεται κλήση της συνάρτησης FREQ_DETECTION_STOP() η οποία αναλύεται

και αυτή σε επόμενη ενότητα και αποσκοπεί στην παύση της δειγματοληψίας του σήματος από το μικρόφωνο επαφής απενεργοποιώντας τον αναλογικό-ψηφιακό μετατροπέα.

Εν συνεχεία, αφού έχει υπολογιστεί η θεμελιώδης συχνότητα ταλάντωσης της χορδής, εμφανίζεται στην σειριακή οθόνη το μήνυμα (Gb String) και η τιμή της συχνότητας που έχει υπολογιστεί. Αυτό επιτυγχάνεται με την χρήση της εντολής Serial.println(). Στην συνέχεια, χρησιμοποιούνται οι εντολές lcd.setCursor() και lcd.print(" ") η οποίες έχουν σκοπό την μετακίνηση του κέρσορα στην πρώτη στήλη της δεύτερης γραμμής της LCD οθόνης, το σβήσιμο της παλιάς τιμής της συχνότητας και την εμφάνιση της πραγματικής τιμής της συχνότητας που υπολογίστηκε. Με αυτόν τον τρόπο, ο χρήστης μπορεί να δει στην LCD οθόνη την πραγματική τιμή της συχνότητας ταλάντωσης της χορδής Σολb (Gb). Αφότου ολοκληρωθεί η εμφάνιση της συχνότητας στην LCD οθόνη, εκτελείται η εντολή (k++), η οποία έχει σκοπό την αύξηση της τιμής του μετρητή (k) κατά μία μονάδα.

Έπειτα, εφαρμόζεται η τεχνική ελέγχου (if – else if – else), η οποία αποσκοπεί στον έλεγχο της συχνότητας ταλάντωσης της χορδής. Ουσιαστικά, με την εκτέλεση της εντολής ελέγχου if (frequency < 184.90 && frequency > 164 && frequency > 0), ελέγχεται αν η τιμή της μεταβλητής (frequency) είναι μικρότερη από 184.90Hz και μεγαλύτερη από 164Hz και μεγαλύτερη από 0Hz. Συνεπώς, αν ικανοποιείται αυτή η συνθήκη σημαίνει ότι η συχνότητα ταλάντωσης της χορδής Σολb (Gb) είναι χαμηλότερη από την επιθυμητή συχνότητα και για να διορθωθεί θα πρέπει να εκτελεστεί το μπλοκ εντολών της συνθήκης ελέγχου (if). Εντός λοιπόν του μπλοκ εντολών, γίνεται μηδενισμός της μεταβλητής (frequency) και ύστερα εκτελείται ένας βρόγχος επανάληψης (for) ο οποίος επαναλαμβάνει το μπλοκ εντολών του δέκα έξι φορές. Όσον αφορά το μπλοκ εντολών του βρόγχου επανάληψης (for), γίνεται κλήση της συνάρτησης (Counterclockwise_GStep_Motor ()). Η συνάρτηση αυτή αναλύεται σε επόμενη ενότητα και έχει σκοπό την αντί-ωρολογιακή περιστροφή του άξονα του βηματικού κινητήρα και κατά συνέπεια την αντί-ωρολογιακή περιστροφή του κλειδιού της χορδής Σολb (Gb). Μετά την ολοκλήρωση του βρόγχου επανάληψης (for), εκτελείται η εντολή delay(2) η οποία δημιουργεί μια χρονική καθυστέρηση των 2msec. Το μπλοκ εντολών της συνθήκης ελέγχου (if) ολοκληρώνεται με την κλήση της συνάρτησης (GStep_Motor_Stop ()), η οποία αναλύεται σε επόμενο κεφάλαιο και σκοπός της είναι να απενεργοποιήσει τον βηματικό κινητήρα. Αν όμως δεν ικανοποιείται η συνθήκη ελέγχου (if), τότε εκτελείται η εντολή ελέγχου else if (frequency > 185.10 && frequency < 233 && frequency > 0), όπου ελέγχεται αν η τιμή της μεταβλητής (frequency) είναι μεγαλύτερη από 185.10Hz και μικρότερη από 233Hz και μεγαλύτερη από 0Hz. Συνεπώς, αν ικανοποιείται αυτή η συνθήκη σημαίνει ότι η συχνότητα ταλάντωσης της χορδής Σολb (Gb) είναι υψηλότερη από την επιθυμητή συχνότητα και για να διορθωθεί θα πρέπει να εκτελεστεί το μπλοκ εντολών της συνθήκης ελέγχου (else if). Ουσιαστικά, εντός του μπλοκ εντολών γίνεται μηδενισμός της μεταβλητής (frequency) και ύστερα εκτελείται ένας βρόγχος επανάληψης (for). Εντός του μπλοκ εντολών του βρόγχου επανάληψης (for) γίνεται κλήση της συνάρτησης (Clockwise_GStep_Motor ()), της οποίας η ανάλυση πραγματοποιείται σε επόμενη ενότητα. Η κλήση της συνάρτησης (Clockwise_GStep_Motor ()), αποσκοπεί στην ωρολογιακή περιστροφή του άξονα του βηματικού κινητήρα και κατά συνέπεια στην ωρολογιακή περιστροφή του κλειδιού της χορδής Σολb (Gb). Μετά την ολοκλήρωση του βρόγχου επανάληψης (for), εκτελείται η εντολή delay(2) η οποία δημιουργεί μια χρονική καθυστέρηση των 2msec. Η τελευταία εντολή που εκτελείται στο μπλοκ εντολών της

συνθήκης ελέγχου (else if) αφορά την κλήση της συνάρτησης (GStep_Motor_Stop ()), η οποία αποσκοπεί στην απενεργοποίηση του βηματικού κινητήρα. Σε περίπτωση όμως που δεν ικανοποιείται η προηγούμενη συνθήκη (else if), τότε εκτελείται η εντολή ελέγχου else if (frequency >=184.90 && frequency <=185.10), η οποία ελέγχει αν η τιμή της μεταβλητής (frequency) είναι μεγαλύτερη ή ίση από 184.90Hz και μικρότερη ή ίση από 185.10Hz. Συνεπώς, αν ικανοποιείται αυτή η συνθήκη τότε η πραγματική συχνότητα ταλάντωσης της χορδής Σολb (Gb) βρίσκεται εντός του εύρους των επιθυμητών συχνοτήτων της χορδής και εκτελείται το αντίστοιχο μπλοκ εντολών. Πιο συγκεκριμένα, εντός του συγκεκριμένου μπλοκ εντολών, αποθηκεύεται στην σημαία (flag) με όνομα (string_3) η τιμή ένα. Με αυτόν τον τρόπο υποδηλώνεται στο πρόγραμμα η ολοκλήρωση του κουρδίσματος της χορδής Σολb (Gb). Έπειτα, εκτελείται η εντολή (k = 21), όπου πραγματοποιείται αρχικοποίηση του μετρητή (k) με την τιμή είκοσι ένα και αμέσως μετά, εκτελείται η εντολή (frequency = 0.00) η οποία μηδενίζει την τιμή της μεταβλητής (frequency). Ύστερα, πραγματοποιείται εκτέλεση των εντολών lcd.clear(), lcd.setCursor() και lcd.print(“ “) με την κατάλληλη σειρά, ώστε να εμφανιστεί το μήνυμα (Hit Bb string) στην LCD οθόνη. Το μήνυμα αυτό παραμένει στην LCD οθόνη για 1sec εξαιτίας της εντολής (delay1000) και έχει σκοπό να προτρέψει τον χρήστη να διεγείρει την επόμενη χορδή η οποία είναι η Σιb (Bb). Στην συνεχεία, εκτελούνται ξανά οι εντολές lcd.clear(), lcd.setCursor() και lcd.print(“ “) με την κατάλληλη σειρά ώστε, να εμφανιστεί στην πρώτη γραμμή της LCD οθόνης το μήνυμα (Bb -> 233.08 Hz), το οποίο υποδηλώνει την επιθυμητή τιμή της συχνότητας της χορδής Σιb (Bb) και στην δεύτερη γραμμή το μήνυμα (Hz Note) σε συγκεκριμένες θέσης. Πιο συγκεκριμένα, το μήνυμα της δεύτερης γραμμής αποσκοπεί στην εκτύπωση της πραγματική συχνότητα πριν το (Hz) και η νότα στην οποία αντιστοιχεί η συχνότητα ταλάντωσης της χορδής πριν το (Note) κατά την διάρκεια κουρδίσματος της χορδής Σιb (Bb). Στην περίπτωση όμως που δεν ικανοποιείται καμία συνθήκη από τις εντολές ελέγχου (if-else if), τότε εκτελείται το μπλοκ εντολών της συνθήκης (else). Ουσιαστικά, το μπλοκ εντολών της συνθήκης (else) αποτελείται από μία μόνο εντολή η οποία μηδενίζει την τιμή της μεταβλητής (frequency). Με αυτό τον τρόπο, αντιμετωπίζονται οι συχνότητες που μπορούν να προκύψουν από την ταυτόχρονη διέγερση δύο χορδών ή από την διέγερση λανθασμένης χορδής. Σε αυτό το σημείο ολοκληρώνεται το μπλοκ εντολών του βρόγχου επανάληψης (while), το οποίο έχει να κάνει με το κούρδισμα της χορδής Σολb (Gb). Η διαδικασία που μόλις περιγράφτηκε, επαναλαμβάνεται μέχρι να επιτευχθεί το κούρδισμα της χορδής Σολb (Gb) εντός του εύρος των συχνοτήτων ταλάντωσης που έχει οριστεί για την συγκεκριμένη χορδή.

Αφού ολοκληρωθεί το κούρδισμα της χορδής Σολb (Gb), το πρόγραμμα συνεχίζει στην πραγματοποίηση του κουρδίσματος της χορδής Σιb (Bb). Αυτό επιτυγχάνεται με την χρήση του βρόγχου επανάληψης while(), ο οποίος στο μπλοκ εντολών του περιλαμβάνει τις απαραίτητες εντολές για το κούρδισμα της χορδής Σιb (Bb). Επιπλέον, η συνθήκη που βρίσκεται εντός των παρενθέσεων του βρόγχου επανάληψης (while) είναι η (string_2 != 1), η οποία δηλώνει ότι όσο η σημαία (flag) με όνομα (string_2) δεν είναι ίση με ένα, τόσο θα συνεχίζει να εκτελείται το μπλοκ εντολών του βρόγχου επανάληψης. Πιο συγκεκριμένα, όσο η σημαία (flag) με όνομα (string_2) είναι μηδέν αυτό σημαίνει ότι η χορδή Σιb (Bb) είναι ξεκούρδιστη. Μόλις όμως, η σημαία (flag) με όνομα (string_2) γίνει ένα, τότε αυτό σημαίνει ότι η χορδή Σιb (Bb) έχει κουρδιστεί και το πρόγραμμα μπορεί να συνεχίσει στο κούρδισμα της επόμενης χορδής. Επειδή όμως η συγκεκριμένη σημαία (flag) έχει αρχικοποιηθεί με την τιμή μηδέν, το μπλοκ εντολών του βρόγχου επανάληψης (while) θα εκτελεστεί μία φορά τουλάχιστον.

Εντός λοιπόν του μπλοκ εντολών του βρόγχου επανάληψης (while), εκτελούνται οι εντολές οι οποίες είναι απαραίτητες για το κούρδισμα της χορδής Σιb (Bb). Πιο συγκεκριμένα, υπάρχει μία συνθήκη ελέγχου (if), η οποία ελέγχει την τιμή του μετρητή (k) αν είναι μεγαλύτερη από είκοσι. Ο μετρητής (k) έχει αρχικοποιηθεί στον προηγούμενο βρόγχο επανάληψης (while) με την τιμή είκοσι ένα, με αποτέλεσμα να εκτελείται το μπλοκ εντολών της συνθήκης ελέγχου (if). Πιο συγκεκριμένα, εντός του μπλοκ εντολών της συνθήκης (if), εκτελείτε πρώτα η εντολή pinMode(39, OUTPUT), η οποία καθορίζει τον ψηφιακό ακροδέκτη τριάντα εννιά ως έξοδο. Έπειτα, εκτελείται η εντολή digitalWrite(39, LOW), όπου ο ψηφιακός ακροδέκτης τριάντα εννιά μεταβαίνει σε κατάσταση χαμηλού δυναμικού για 10msec, διότι αμέσως μετά εκτελείται η εντολή χρονικής καθυστέρησης delay(10). Μετά εκτελείται η εντολή pinMode(39, INPUT), η οποία καθορίζει τον ψηφιακό ακροδέκτη τριάντα εννιά ως είσοδο. Ύστερα, το μπλοκ εντολών της συνθήκης ελέγχου (if) κλείνει με την εντολή (k = 0), όπου γίνεται μηδενισμός του μετρητή (k). Σκοπός της συγκεκριμένης συνθήκης ελέγχου, είναι η γείωση (GND) του σήματος που προέρχεται από το μικρόφωνο επαφής ανά τακτά χρονικά διαστήματα, ώστε να μην υπάρχουν παράσιτα τα οποία παραμορφώνουν το σήμα και οδηγούν σε λανθασμένους υπολογισμούς της θεμελιώδης συχνότητας ταλάντωσης.

Στην συνέχεια, μετά την συνθήκη ελέγχου (if), γίνεται κλήση της συνάρτησης FREQ_DETECTION(), η οποία ξεκινά την διαδικασία δειγματοληψίας του σήματος από το μικρόφωνο επαφής και αναλύεται σε επόμενη ενότητα. Η διαδικασία δειγματοληψίας, διαρκεί για 0.5sec εξαιτίας της εντολής καθυστέρησης delay(500) που εκτελείται αμέσως μετά. Ύστερα, γίνεται κλήση της συνάρτησης FREQ_DETECTION1(), η οποία αναλύεται σε επόμενη ενότητα και έχει σκοπό τόσο τον υπολογισμό της θεμελιώδης συχνότητας ταλάντωσης της χορδής όσο και την εύρεση της νότας στην οποία αντιστοιχεί η συγκεκριμένη συχνότητα. Έπειτα, γίνεται κλήση της συνάρτησης FREQ_DETECTION_STOP() η οποία αναλύεται και αυτή σε επόμενη ενότητα και αποσκοπεί στην παύση της δειγματοληψίας του σήματος από το μικρόφωνο επαφής απενεργοποιώντας τον αναλογικό-ψηφιακό μετατροπέα.

Μετά λοιπόν την ολοκλήρωση της δειγματοληψίας και αφού έχει υπολογιστεί η θεμελιώδης συχνότητα ταλάντωσης της χορδής, εμφανίζεται στην σειριακή οθόνη το μήνυμα (Bb String) και η τιμή της συχνότητας που έχει υπολογιστεί. Αυτό επιτυγχάνεται με την χρήση της εντολής Serial.println(). Στην συνέχεια, χρησιμοποιούνται οι εντολές lcd.setCursor() και lcd.print(“ “) η οποίες έχουν σκοπό την μετακίνηση του κέρσορα στην πρώτη στήλη της δεύτερης γραμμής της LCD οθόνης, το σβήσιμο της παλιάς τιμής της συχνότητας και την εμφάνιση της πραγματικής τιμής της θεμελιώδης συχνότητας ταλάντωσης της χορδής που υπολογίστηκε. Με αυτόν τον τρόπο, ο χρήστης μπορεί να δει στην LCD οθόνη την πραγματική τιμή της συχνότητας ταλάντωσης της χορδής Σιb (Bb). Αφότου ολοκληρωθεί η εμφάνιση της συχνότητας στην LCD οθόνη, εκτελείται η εντολή (k++), η οποία έχει σκοπό την αύξηση της τιμής του μετρητή (k) κατά μία μονάδα.

Εν συνεχεία, εφαρμόζεται η τεχνική ελέγχου (if – else if – else), με σκοπό τον έλεγχο της συχνότητας ταλάντωσης της χορδής. Πιο συγκεκριμένα, με την εντολή ελέγχου if (frequency < 232.98 && frequency > 207 && frequency > 0), ελέγχεται αν η τιμή της μεταβλητής (frequency) είναι μικρότερη από 232.98Hz και μεγαλύτερη από 207Hz και μεγαλύτερη από 0Hz. Επομένως, αν ικανοποιείται αυτή η συνθήκη σημαίνει ότι η συχνότητα ταλάντωσης της χορδής Σιb (Bb) είναι χαμηλότερη από την επιθυμητή συχνότητα και για να διορθωθεί θα πρέπει να εκτελεστεί το μπλοκ εντολών της συνθήκης

ελέγχου (if). Εντός λοιπόν του μπλοκ εντολών, γίνεται μηδενισμός της μεταβλητής (frequency) και ύστερα εκτελείται ένας βρόγχος επανάληψης (for). Ο βρόγχος επανάληψης (for) επαναλαμβάνει το μπλοκ εντολών του είκοσι ένα φορές. Εντός του μπλοκ εντολών του βρόγχου επανάληψης (for) γίνεται κλήση της συνάρτησης (Counterclockwise_BStep_Motor ()). Η συνάρτηση αυτή αναλύεται σε επόμενη ενότητα και έχει σκοπό την αντί-ωρολογιακή περιστροφή του άξονα του βηματικού κινητήρα και κατά συνέπεια την αντί-ωρολογιακή περιστροφή του κλειδιού της χορδής Σιb (B). Μετά την ολοκλήρωση του βρόγχου επανάληψης (for), εκτελείται η εντολή delay(2) η οποία δημιουργεί μια χρονική καθυστέρηση των 2msec. Το μπλοκ εντολών της συνθήκης ελέγχου (if) ολοκληρώνεται με την κλήση της συνάρτησης (BStep_Motor_Stop ()), η οποία αναλύεται σε επόμενο κεφάλαιο και σκοπός της είναι να απενεργοποιήσει τον βηματικό κινητήρα. Στην περίπτωση όμως, που δεν ικανοποιείται η συνθήκη ελέγχου (if), τότε εκτελείται η εντολή ελέγχου else if (frequency > 233.18 && frequency < 293 && frequency > 0), όπου ελέγχεται αν η τιμή της μεταβλητής (frequency) είναι μεγαλύτερη από 233.18Hz και μικρότερη από 293Hz και μεγαλύτερη από 0Hz. Συνεπώς, αν ικανοποιείται αυτή η συνθήκη σημαίνει ότι η συχνότητα ταλάντωσης της χορδής Σιb (Bb) είναι υψηλότερη από την επιθυμητή συχνότητα και για να διορθωθεί θα πρέπει να εκτελεστεί το μπλοκ εντολών της συνθήκης ελέγχου (else if). Εντός λοιπόν του μπλοκ εντολών, γίνεται μηδενισμός της μεταβλητής (frequency) και ύστερα εκτελείται ένας βρόγχος επανάληψης (for). Εντός του μπλοκ εντολών του βρόγχου επανάληψης (for) γίνεται κλήση της συνάρτησης (Clockwise_BStep_Motor ()), η οποία αναλύεται σε επόμενη ενότητα. Σκοπός της συνάρτησης (Clockwise_BStep_Motor ()), είναι να περιστρέψει τον άξονα του βηματικού κινητήρα ωρολογιακά και κατά συνέπεια να περιστρέψει ωρολογιακά και το κλειδί της χορδής Σιb (Bb). Αφότου ολοκληρωθεί ο βρόγχος επανάληψης (for), εκτελείται η εντολή delay(2) η οποία δημιουργεί μια χρονική καθυστέρηση των 2msec. Η τελευταία εντολή που εκτελείται στο μπλοκ εντολών της συνθήκης ελέγχου (else if) αφορά την κλήση της συνάρτησης (BStep_Motor_Stop ()), η οποία έχει σκοπό να απενεργοποιήσει τον βηματικό κινητήρα. Αν όμως δεν ικανοποιείται η προηγούμενη συνθήκη (else if), τότε εκτελείται η εντολή ελέγχου else if (frequency >=232.98 && frequency <=233.18), η οποία ελέγχει αν η τιμή της μεταβλητής (frequency) είναι μεγαλύτερη ή ίση από 232.98Hz και μικρότερη ή ίση από 233.18Hz. Επομένως, αν ικανοποιείται αυτή η συνθήκη τότε η πραγματική συχνότητα ταλάντωσης της χορδής Σιb (Bb) βρίσκεται εντός του εύρους των επιθυμητών συχνοτήτων της χορδής και εκτελείται το αντίστοιχο μπλοκ εντολών. Εντός λοιπόν του συγκεκριμένου μπλοκ εντολών, αποθηκεύεται στην σημαία (flag) με όνομα (string_2) η τιμή ένα. Με αυτόν τον τρόπο υποδηλώνεται στο πρόγραμμα η ολοκλήρωση του κουρδίσματος της χορδής Σιb (Bb). Ύστερα, εκτελείται η εντολή (k = 21), όπου πραγματοποιείται αρχικοποίηση του μετρητή (k) με την τιμή είκοσι ένα και αμέσως μετά, πραγματοποιείται μηδενισμός της μεταβλητής (frequency) μέσω της εντολής (frequency = 0.00). Έπειτα, εκτελούνται οι εντολές lcd.clear(), lcd.setCursor() και lcd.print(“ “) με την κατάλληλη σειρά, ώστε να εμφανιστεί το μήνυμα (Hit eb string) στην LCD οθόνη. Το μήνυμα αυτό παραμένει στην LCD οθόνη για 1sec εξαιτίας της εντολής (delay1000) και σκοπό έχει να προτρέψει τον χρήστη να διεγείρει την επόμενη χορδή η οποία είναι η Μιb καντίνι (eb). Εν συνεχεία, εκτελούνται ξανά οι εντολές lcd.clear(), lcd.setCursor() και lcd.print(“ “) με την κατάλληλη σειρά ώστε, να εμφανιστεί στην πρώτη γραμμή της LCD οθόνης το μήνυμα (eb -> 311.13 Hz), το οποίο υποδηλώνει την επιθυμητή τιμή της συχνότητας της χορδής Μιb καντίνι (eb) και στην δεύτερη γραμμή το μήνυμα (Hz Note) σε συγκεκριμένες

θέσης. Σκοπός του μηνύματος της δεύτερης γραμμής είναι να εκτυπώνεται η πραγματική συχνότητα πριν το (Hz) και η νότα στην οποία αντιστοιχεί η συχνότητα ταλάντωσης της χορδής πριν το (Note) κατά την διάρκεια κουρδίσματος της χορδής Μιb καντίνι (eb). Αν όμως, δεν ικανοποιείται καμία συνθήκη από τις εντολές ελέγχου (if-else if), τότε εκτελείται το μπλοκ εντολών της συνθήκης (else). Πιο συγκεκριμένα, το μπλοκ εντολών της συνθήκης (else) περιλαμβάνει μία μόνο εντολή η οποία μηδενίζει την τιμή της μεταβλητής (frequency). Με αυτό τον τρόπο, αντιμετωπίζονται οι συχνότητες που μπορούν να προκύψουν από την ταυτόχρονη διέγερση δύο χορδών ή από την διέγερση λανθασμένης χορδής. Σε αυτό το σημείο ολοκληρώνεται το μπλοκ εντολών του βρόγχου επανάληψης (while), το οποίο έχει να κάνει με το κούρδισμα της χορδής Σιb (Bb). Η διαδικασία που μόλις περιγράφτηκε, εκτελείτε ξανά μέχρι να πραγματοποιηθεί το κούρδισμα της χορδής Σιb (Bb) εντός του εύρος των συχνοτήτων ταλάντωσης που έχει οριστεί για την συγκεκριμένη χορδή.

Με την ολοκλήρωση του κουρδίσματος της χορδής Σιb (Bb), σειρά έχει το κούρδισμα της χορδής Μιb καντίνι (eb). Αυτό επιτυγχάνεται με την χρήση του βρόγχου επανάληψης while(), ο οποίος στο μπλοκ εντολών του περιλαμβάνει τις απαραίτητες εντολές για το κούρδισμα της χορδής Μιb καντίνι (eb). Η συνθήκη που βρίσκεται εντός των παρενθέσεων του βρόγχου επανάληψης (while) είναι η (string_1 != 1) η οποία σημαίνει ότι όσο η σημαία (flag) με όνομα (string_1) δεν είναι ίση με ένα, τόσο θα συνεχίζει να εκτελείται το μπλοκ εντολών του βρόγχου επανάληψης. Πιο συγκεκριμένα, όσο η σημαία (flag) με όνομα (string_1) είναι μηδέν αυτό σημαίνει ότι η χορδή Μιb καντίνι (eb) είναι ξεκούρδιστη. Μόλις όμως η σημαία (flag) με όνομα (string_1) γίνει ένα, τότε αυτό σημαίνει ότι η χορδή Μιb καντίνι (eb) έχει κουρδιστεί και το πρόγραμμα μπορεί να συνεχίσει στις επόμενες εντολές. Επειδή όμως η συγκεκριμένη σημαία (flag) έχει αρχικοποιηθεί με την τιμή μηδέν, το μπλοκ εντολών του βρόγχου επανάληψης (while) θα εκτελεστεί μία φορά τουλάχιστον.

Ουσιαστικά, εντός του μπλοκ εντολών του βρόγχου επανάληψης (while), εκτελούνται οι εντολές οι οποίες είναι απαραίτητες για το κούρδισμα της χορδής Μιb καντίνι (eb). Όπως αναφέρθηκε και προηγουμένως, υπάρχει μία συνθήκη ελέγχου (if), η οποία ελέγχει την τιμή του μετρητή (k) αν είναι μεγαλύτερη από είκοσι. Ο μετρητής (k) έχει αρχικοποιηθεί στον προηγούμενο βρόγχο επανάληψης (while) με την τιμή είκοσι ένα, με αποτέλεσμα να εκτελείται το μπλοκ εντολών της συνθήκης ελέγχου (if). Ουσιαστικά, στο μπλοκ εντολών της συνθήκης (if) εκτελείτε πρώτα η εντολή pinMode(39, OUTPUT), η οποία καθορίζει τον ψηφιακό ακροδέκτη τριάντα εννιά ως έξοδο. Έπειτα, εκτελείται η εντολή digitalWrite(39, LOW), όπου ο ψηφιακός ακροδέκτης τριάντα εννιά μεταβαίνει σε κατάσταση χαμηλού δυναμικού για 10msec, διότι αμέσως μετά εκτελείται η εντολή χρονικής καθυστέρησης delay(10). Ύστερα, εκτελείται η εντολή pinMode(39, INPUT), η οποία καθορίζει τον ψηφιακό ακροδέκτη τριάντα εννιά ως είσοδο. Το μπλοκ εντολών της συνθήκης ελέγχου (if) ολοκληρώνεται με την εντολή (k = 0), όπου γίνεται μηδενισμός του μετρητή (k). Σκοπός της συγκεκριμένης συνθήκης ελέγχου είναι η γείωση (GND) του σήματος που προέρχεται από το μικρόφωνο επαφής ανά τακτά χρονικά διαστήματα, ώστε να μην υπάρχουν παράσιτα τα οποία παραμορφώνουν το σήμα και οδηγούν σε λανθασμένους υπολογισμούς της συχνότητας ταλάντωσης της χορδής..

Αφότου ολοκληρωθεί η εντολή της συνθήκης ελέγχου (if), γίνεται κλήση της συνάρτησης FREQ_DETECTION(), η οποία αναλύεται σε επόμενη ενότητα και αποσκοπεί στην εκκίνηση της διαδικασίας της δειγματοληψίας του σήματος από το

μικρόφωνο επαφής. Η διαδικασία δειγματοληψίας, διαρκεί για 0.5sec εξαιτίας της εντολής καθυστέρησης delay(500) που εκτελείται αμέσως μετά. Ύστερα, γίνεται κλήση της συνάρτησης FREQ_DETECTION1(), η οποία έχει σκοπό τόσο τον υπολογισμό της θεμελιώδης συχνότητας ταλάντωσης της χορδής όσο και την εύρεση της νότας στην οποία αντιστοιχεί η συγκεκριμένη συχνότητα και η ανάλυσή της γίνεται σε επόμενη ενότητα. Έπειτα, γίνεται κλήση της συνάρτησης FREQ_DETECTION_STOP() η οποία αναλύεται και αυτή σε επόμενη ενότητα και αποσκοπεί στην παύση της δειγματοληψίας του σήματος από το μικρόφωνο επαφής απενεργοποιώντας τον αναλογικό-ψηφιακό μετατροπέα.

Εν συνεχεία, αφού έχει υπολογιστεί η θεμελιώδης συχνότητα ταλάντωσης της χορδής, εμφανίζεται στην σειριακή οθόνη το μήνυμα (eb String) και η τιμή της συχνότητας που έχει υπολογιστεί. Αυτό επιτυγχάνεται με την χρήση της εντολής Serial.println(). Στην συνέχεια, χρησιμοποιούνται οι εντολές lcd.setCursor() και lcd.print(" ") η οποίες έχουν σκοπό την μετακίνηση του κέρσορα στην πρώτη στήλη της δεύτερης γραμμής της LCD οθόνης, το σβήσιμο της παλιάς τιμής της συχνότητας και την εμφάνιση της πραγματικής τιμής της συχνότητας που υπολογίστηκε. Με αυτόν τον τρόπο, ο χρήστης μπορεί να δει στην LCD οθόνη την πραγματική τιμή της συχνότητας ταλάντωσης της χορδής Μιb καντίνι (eb). Αφότου ολοκληρωθεί η εμφάνιση της συχνότητας στην LCD οθόνη, εκτελείται η εντολή (k++), η οποία έχει σκοπό την αύξηση της τιμής του μετρητή (k) κατά μία μονάδα.

Έπειτα, εφαρμόζεται η τεχνική ελέγχου (if – else if – else), η οποία αποσκοπεί στον έλεγχο της συχνότητας ταλάντωσης της χορδής. Ουσιαστικά, με την εκτέλεση της εντολής ελέγχου if (frequency < 310.18 && frequency > 277 && frequency > 0), ελέγχεται αν η τιμή της μεταβλητής (frequency) είναι μικρότερη από 310.18Hz και μεγαλύτερη από 277Hz και μεγαλύτερη από 0Hz. Συνεπώς, αν ικανοποιείται αυτή η συνθήκη σημαίνει ότι η συχνότητα ταλάντωσης της χορδής Μιb καντίνι (eb) είναι χαμηλότερη από την επιθυμητή συχνότητα και για να διορθωθεί θα πρέπει να εκτελεστεί το μπλοκ εντολών της συνθήκης ελέγχου (if). Εντός λοιπόν του μπλοκ εντολών, γίνεται μηδενισμός της μεταβλητής (frequency) και ύστερα εκτελείται ένας βρόγχος επανάληψης (for) ο οποίος επαναλαμβάνει το μπλοκ εντολών του είκοσι ένα φορές. Όσον αφορά το μπλοκ εντολών του βρόγχου επανάληψης (for), γίνεται κλήση της συνάρτησης (Counterclockwise_eStep_Motor ()). Η συνάρτηση αυτή αναλύεται σε επόμενη ενότητα και έχει σκοπό την αντί-ωρολογιακή περιστροφή του άξονα του βηματικού κινητήρα και κατά συνέπεια την αντί-ωρολογιακή περιστροφή του κλειδιού της χορδής Μιb καντίνι (eb). Μετά την ολοκλήρωση του βρόγχου επανάληψης (for), εκτελείται η εντολή delay(2) η οποία δημιουργεί μια χρονική καθυστέρηση των 2msec. Το μπλοκ εντολών της συνθήκης ελέγχου (if) ολοκληρώνεται με την κλήση της συνάρτησης (eStep_Motor_Stop ()), η οποία αναλύεται σε επόμενο κεφάλαιο και σκοπός της είναι να απενεργοποιήσει τον βηματικό κινητήρα. Αν όμως δεν ικανοποιείται η συνθήκη ελέγχου (if), τότε εκτελείται η εντολή ελέγχου else if (frequency > 312.70 && frequency < 392 && frequency > 0), όπου ελέγχεται αν η τιμή της μεταβλητής (frequency) είναι μεγαλύτερη από 312.70Hz και μικρότερη από 392Hz και μεγαλύτερη από 0Hz. Συνεπώς, αν ικανοποιείται αυτή η συνθήκη σημαίνει ότι η συχνότητα ταλάντωσης της χορδής Μιb καντίνι (eb) είναι υψηλότερη από την επιθυμητή συχνότητα και για να διορθωθεί θα πρέπει να εκτελεστεί το μπλοκ εντολών της συνθήκης ελέγχου (else if). Ουσιαστικά, εντός του μπλοκ εντολών γίνεται μηδενισμός της μεταβλητής (frequency) και ύστερα εκτελείται ένας βρόγχος επανάληψης (for). Εντός του μπλοκ εντολών του βρόγχου επανάληψης (for) γίνεται κλήση

της συνάρτησης (Clockwise_eStep_Motor ()), της οποίας η ανάλυση πραγματοποιείται σε επόμενη ενότητα. Η κλήση της συνάρτησης (Clockwise_eStep_Motor ()), αποσκοπεί στην ωρολογιακή περιστροφή του άξονα του βηματικού κινητήρα και κατά συνέπεια στην ωρολογιακή περιστροφή του κλειδιού της χορδής Mιb καντίνι (eb). Μετά την ολοκλήρωση του βρόγχου επανάληψης (for), εκτελείται η εντολή delay(2) η οποία δημιουργεί μια χρονική καθυστέρηση των 2msec. Η τελευταία εντολή που εκτελείται στο μπλοκ εντολών της συνθήκης ελέγχου (else if) αφορά την κλήση της συνάρτησης (eStep_Motor_Stop ()), η οποία αποσκοπεί στην απενεργοποίηση του βηματικού κινητήρα. Σε περίπτωση όμως που δεν ικανοποιείται η προηγούμενη συνθήκη (else if), τότε εκτελείται η εντολή ελέγχου else if (frequency >=310.18 && frequency <=312.70), η οποία ελέγχει αν η τιμή της μεταβλητής (frequency) είναι μεγαλύτερη ή ίση από 310.18Hz και μικρότερη ή ίση από 312.70Hz. Συνεπώς, αν ικανοποιείται αυτή η συνθήκη τότε η πραγματική συχνότητα ταλάντωσης της χορδής Mιb καντίνι (eb) βρίσκεται εντός του εύρους των επιθυμητών συχνοτήτων της χορδής και εκτελείται το αντίστοιχο μπλοκ εντολών. Πιο συγκεκριμένα, εντός του συγκεκριμένου μπλοκ εντολών, αποθηκεύεται στην σημαία (flag) με όνομα (string_1) η τιμή ένα. Με αυτόν τον τρόπο υποδηλώνεται στο πρόγραμμα η ολοκλήρωση του κουρδίσματος της χορδής Mιb καντίνι (eb). Έπειτα, εκτελείται η εντολή (k = 21), όπου πραγματοποιείται αρχικοποίηση του μετρητή (k) με την τιμή είκοσι ένα και αμέσως μετά, εκτελείται η εντολή (frequency = 0.00) η οποία μηδενίζει την τιμή της μεταβλητής (frequency). Στην περίπτωση όμως που δεν ικανοποιείται καμία συνθήκη από τις εντολές ελέγχου (if-else if), τότε εκτελείται το μπλοκ εντολών της συνθήκης (else). Ουσιαστικά, το μπλοκ εντολών της συνθήκης (else) αποτελείται από μία μόνο εντολή η οποία μηδενίζει την τιμή της μεταβλητής (frequency). Με αυτό τον τρόπο, αντιμετωπίζονται οι συχνότητες που μπορούν να προκύψουν από την ταυτόχρονη διέγερση δύο χορδών ή από την διέγερση λανθασμένης χορδής. Σε αυτό το σημείο ολοκληρώνεται το μπλοκ εντολών του βρόγχου επανάληψης (while), το οποίο έχει να κάνει με το κούρδισμα της χορδής Mιb καντίνι (eb). Η διαδικασία που μόλις περιγράφτηκε, επαναλαμβάνεται μέχρι να επιτευχθεί το κούρδισμα της χορδής Mιb καντίνι (eb) εντός του εύρος των συχνοτήτων ταλάντωσης που έχει οριστεί για την συγκεκριμένη χορδή.

Αφού λοιπόν κουρδιστεί και η τελευταία χορδή της κιθάρας, εκτελούνται οι τελευταίες εντολές του μπλοκ εντολών της συνθήκης ελέγχου (if). Πιο συγκεκριμένα, εκτελούνται οι εντολές lcd.clear(), lcd.setCursor() και lcd.print(“ “) με κατάλληλη σειρά ώστε να εμφανιστεί στην LCD οθόνη το μήνυμα (TUNE FINISHED!). Το μήνυμα αυτό παραμένει στην οθόνη για 5sec εξαιτίας της εντολής καθυστέρησης που εκτελείται delay(5000). Έπειτα, γίνεται κλήση της συνάρτησης Menu (), ώστε να εμφανιστεί στον χρήστη μέσω της LCD οθόνης το μενού επιλογών.

Πίνακας 3.6: Ο κώδικας για το κούρδισμα της κιθάρας ενός ημιτόνου κάτω (Half-Step Down Tune) [32].

Κώδικας Προγραμματισμού (Μέρος 6ο)

else if (menu==2) //Else if menu value is 2 then
{
lcd.clear(); //Clear LCD screen
lcd.setCursor(0,0); //Set cursor at first column and first line

```
    lcd.print(" Half Step Down "); //Print message at LCD screen
    lcd.setCursor(4,1); //Set cursor at fifth column and first line
    lcd.print("Selected"); //Print message at LCD screen
    delay(1000); //Delay 1 sec
    lcd.clear(); //Clear LCD screen
    lcd.setCursor(1,0); //Set Cursor at third column and first line
    lcd.print("Hit Eb string!"); //Print message at LCD screen
    delay(1000); //Delay 1 sec
    lcd.clear(); //Clear LCD screen
    lcd.setCursor(1,0); //Set Cursor at third column and first line
    lcd.print("Eb -> 77.78 Hz"); //Print message at LCD screen
    lcd.setCursor(7,1); //Set Cursor at first column and second line
    lcd.print("Hz   Note"); //print message at LCD screen
    //frequency = 0; //Reset the value at frequency variable
    k = 21; //Store value at k variable
    string_6 = 0;//Reset the value at string_6
    string_5 = 0;//Reset the value at string_5
    string_4 = 0;//Reset the value at string_4
    string_3 = 0;//Reset the value at string_3
    string_2 = 0;//Reset the value at string_2
    string_1 = 0;//Reset the value at string_1
    while(string_6 != 1) //While string_6 flag isn't set repeat
    {
     if (k>20) //If k variable is bigger than 20, then
     {
      pinMode(39,OUTPUT); //Determine digital pin 39 as Output
      digitalWrite(39,LOW); //Set low voltage (GND) at pin 39
      delay(10); //Delay 10 msec
      pinMode(39,INPUT); //Determine digital pin 39 as Input
      k = 0; //Reset k variable
     } //if (k>20)
     FREQ_DETECTION(); //Call FREQ_DETECTION function (Change analog port for
the income signal)
     delay(500); //Delay 0.5 sec
     FREQ_DETECTION1(); //Call FREQ_DETECTION1 function (Calculate frequency
and stop interrupts by ADC)
     FREQ_DETECTION_STOP(); //Call FREQ_DETECTION_STOP function (Stop
frequency detection)
     Serial.println("Eb String"); //Print message at Serial Port
     Serial.println(frequency); //Print frequency value at Serial Port
```

```
      lcd.setCursor(0,1); //Set cursor at first column and second line
      lcd.print("       "); //Print message at lcd screen
      lcd.setCursor(0,1); //Set cursor at first column and second line
      lcd.print(frequency); //Print frequency value at LCD screen
      k++; //Increase variable k value by one
      if (frequency < 77.68 && frequency > 65 && frequency > 0) //If frequency value is
smaller than 77.68Hz and bigger than 65Hz and bigger than 0Hz, then
      {
       frequency = 0.00; //Reset frequency value
       for (i=0; i<=10; i++) //For 11 times repeat
       {
        Counterclockwise_EStep_Motor (); //Call function for E bass string step motor
       }
       delay(2); //Delay 2 msec
       EStep_Motor_Stop (); //Disable E bass string step motor
      } //if (frequency...
      else if (frequency > 77.88 && frequency < 103 && frequency > 0) //Else if
frequency value is bigger than 77.88Hz and smaller than 103Hz and bigger than 0Hz, then
      {
       frequency = 0.00; //Reset frequency value
       for(i=0; i<=10; i++) //For 11 times repeat
       {
        Clockwise_EStep_Motor (); //Call function for E bass string step motor
       }
       delay(2); //Delay 2 msec
       EStep_Motor_Stop (); //Disable E bass string step motor
      } //else if (frequency...
      else if (frequency >=77.68 && frequency <=77.88) //Else if frequency value is bigger
or equal with 77.68Hz and smaller or equal with 77.88Hz, then Eb bass string has been
tuned
      {
       string_6 = 1; //Set string_6 flag to finish tuning with Eb bass string
       k = 21;  //Store value at k variable
       frequency = 0.00; //Reset frequency value
       lcd.clear(); //Clear LCD screen
       lcd.setCursor(1,0); //Set Cursor at third column and first line
       lcd.print("Hit Ab string!"); //Print message at LCD screen
       delay(1000); //Delay 1 sec
       lcd.clear(); //Clear LCD screen
       lcd.setCursor(1,0); //Set Cursor at third column and first line
```

```
        lcd.print("Ab -> 103.83 Hz"); //Print message at LCD screen
        lcd.setCursor(7,1); //Set Cursor at first column and second line
        lcd.print("Hz   Note"); //print message at LCD screen
       } //else if (frequency...
       else //Else
       {
        frequency = 0.00; //Reset frequency value
       }
     } //while(string_6...
     while(string_5 != 1) //While string_5 flag isn't set repeat
     {
      if (k>20) //If k variable is bigger than 20, then
      {
       pinMode(39,OUTPUT); //Determine digital pin 39 as Output
       digitalWrite(39,LOW); //Set low voltage (GND) at pin 39
       delay(10); //Delay 10 msec
       pinMode(39,INPUT); //Determine digital pin 39 as Input
       k = 0; //Reset k variable
      } //if (k>20)
      FREQ_DETECTION(); //Call FREQ_DETECTION function (Change analog port for the income signal)
      delay(500); //Delay 0.5 sec
      FREQ_DETECTION1(); //Call FREQ_DETECTION1 function (Calculate frequency and stop interrupts by ADC)
      FREQ_DETECTION_STOP(); //Call FREQ_DETECTION_STOP function (Stop frequency detection)
      Serial.println("Ab String"); //Print message at Serial Port
      Serial.println(frequency); //Print frequency value at Serial Port
      lcd.setCursor(0,1); //Set cursor at first column and second line
      lcd.print("       "); //Print message at lcd screen
      lcd.setCursor(0,1); //Set cursor at first column and second line
      lcd.print(frequency); //Print frequency value at LCD screen
      k++; //Increase variable k value by one
      if (frequency < 103.67 && frequency > 92 && frequency > 0) //If frequency value is smaller than 103.67Hz and bigger than 92Hz and bigger than 0Hz, then
      {
       frequency = 0.00; //Reset frequency value
       for (i=0; i<=20; i++) //For 21 times repeat
       {
        Counterclockwise_AStep_Motor (); //Call function for A string step motor
```

```
        }
        delay(2); //Delay 2 msec
        AStep_Motor_Stop (); //Disable A string step motor
      } //if (frequency...
      else if (frequency > 103.96 && frequency < 130 && frequency > 0) //Else if frequency value is bigger than 103.96Hz and smaller than 130Hz and bigger than 0Hz, then
      {
        frequency = 0.00; //Reset frequency value
        for(i=0; i<=20; i++) //For 21 times repeat
        {
          Clockwise_AStep_Motor (); //Call function for A string step motor
        }
        delay(2); //Delay 2 msec
        AStep_Motor_Stop (); //Disable A string step motor
      } //else if (frequency...
      else if (frequency >=103.66 && frequency <=103.96) //Else if frequency value is bigger or equal with 103.66Hz and smaller or equal with 103.96Hz, then Ab string has been tuned
      {
        string_5 = 1; //Set string_5 flag to finish tuning with Ab string
        k = 21; //Store value at k variable
        frequency = 0.00; //Reset frequency value
        lcd.clear(); //Clear LCD screen
        lcd.setCursor(1,0); //Set Cursor at third column and first line
        lcd.print("Hit Db string!"); //Print message at LCD screen
        delay(1000); //Delay 1 sec
        lcd.clear(); //Clear LCD screen
        lcd.setCursor(1,0); //Set Cursor at third column and first line
        lcd.print("Db -> 138.59 Hz"); //Print message at LCD screen
        lcd.setCursor(7,1); //Set Cursor at first column and second line
        lcd.print("Hz   Note"); //print message at LCD screen
      } //else if (frequency...
      else //Else
      {
        frequency = 0.00; //Reset frequency value
      }
    } //while(string_5...
    while(string_4 != 1) //While string_4 flag isn't set repeat
    {
```

```
    if (k>20) //If k variable is bigger than 20, then
    {
     pinMode(39,OUTPUT); //Determine digital pin 39 as Output
     digitalWrite(39,LOW); //Set low voltage (GND) at pin 39
     delay(10); //Delay 10 msec
     pinMode(39,INPUT); //Determine digital pin 39 as Input
     k = 0;//Reset k variable
    } //if (k>20)
    FREQ_DETECTION(); //Call FREQ_DETECTION function (Change analog port for the income signal)
    delay(500); //Delay 0.5 sec
    FREQ_DETECTION1(); //Call FREQ_DETECTION1 function (Calculate frequency and stop interrupts by ADC)
    FREQ_DETECTION_STOP(); //Call FREQ_DETECTION_STOP function (Stop frequency detection)
    Serial.println("Db String"); //Print message at Serial Port
    Serial.println(frequency); //Print frequency value at Serial Port
    lcd.setCursor(0,1); //Set cursor at first column and second line
    lcd.print("       "); //Print message at lcd screen
    lcd.setCursor(0,1); //Set cursor at first column and second line
    lcd.print(frequency); //Print frequency value at LCD screen
    k++; //Increase variable k value by one
    if (frequency < 138.35 && frequency > 123 && frequency > 0) //If frequency value is smaller than 138.35z and bigger than 123Hz and bigger than 0Hz, then
    {
     frequency = 0.00; //Reset frequency value
     for (i=0; i<=30; i++) //For 31 times repeat
     {
      Counterclockwise_DStep_Motor (); //Call function for D string step motor
     }
     delay(2); //Delay 2 msec
     DStep_Motor_Stop (); //Disable D string step motor
    } //if (frequency...
    else if (frequency > 138.85 && frequency < 174 && frequency > 0) //Else if frequency value is bigger than 138.85Hz and smaller than 174Hz and bigger than 0Hz, then
    {
     frequency = 0.00; //Reset frequency value
     for(i=0; i<=30; i++) //For 31 times repeat
     {
```

```
          Clockwise_DStep_Motor (); //Call function for D string step motor
         }
         delay(2); //Delay 2 msec
         DStep_Motor_Stop (); //Disable D string step motor
       } //else if (frequency...
       else if (frequency >=138.35 && frequency <=138.85) //Else if frequency value is
bigger or equal with 138.35Hz and smaller or equal with 138.85Hz, then Db string has
been tuned
       {
         string_4 = 1; //Set string_4 flag to finish tuning with Db string
         k = 21;  //Store value at k variable
         frequency = 0.00; //Reset frequency value
         lcd.clear(); //Clear LCD screen
         lcd.setCursor(1,0); //Set Cursor at third column and first line
         lcd.print("Hit Gb string"); //Print message at LCD screen
         delay(1000); //Delay 1 sec
         lcd.clear(); //Clear LCD screen
         lcd.setCursor(1,0); //Set Cursor at third column and first line
         lcd.print("Gb -> 185 Hz"); //Print message at LCD screen
         lcd.setCursor(7,1); //Set Cursor at first column and second line
         lcd.print("Hz   Note"); //print message at LCD screen
       } //else if (frequency...
       else //Else
       {
         frequency = 0.00; //Reset frequency value
       }
     } //while(string_4...*/
     while(string_3 != 1) //While string_3 flag isn't set repeat
     {
       if (k>20) //If k variable is bigger than 20, then
       {
         pinMode(39,OUTPUT); //Determine digital pin 39 as Output
         digitalWrite(39,LOW); //Set low voltage (GND) at pin 39
         delay(10); //Delay 10 msec
         pinMode(39,INPUT); //Determine digital pin 39 as Input
         k = 0; //Reset k variable
       } //if (k>20)
       FREQ_DETECTION(); //Call FREQ_DETECTION function (Change analog port for
the income signal)
       delay(500); //Delay 0.5 sec
```

```
    FREQ_DETECTION1(); //Call FREQ_DETECTION1 function (Calculate frequency and stop interrupts by ADC)
    FREQ_DETECTION_STOP(); //Call FREQ_DETECTION_STOP function (Stop frequency detection)
    Serial.println("Gb String"); //Print message at Serial Port
    Serial.println(frequency); //Print frequency value at Serial Port
    lcd.setCursor(0,1); //Set cursor at first column and second line
    lcd.print("       "); //Print message at lcd screen
    lcd.setCursor(0,1); //Set cursor at first column and second line
    lcd.print(frequency); //Print frequency value at LCD screen
    k++; //Increase variable k value by one
    if (frequency < 184.90 && frequency > 164 && frequency > 0)  //If frequency value is smaller than 184.90Hz and bigger than 164Hz and bigger than 0Hz, then
    {
     frequency = 0.00; //Reset frequency value
     for (i=0; i<=15; i++) //For 16 times repeat
     {
       Clockwise_GStep_Motor (); //Call function for G string step motor
     }
     delay(2); //Delay 2 msec
     GStep_Motor_Stop (); //Disable G string step motor
    } //if (frequency...
    else if (frequency > 185.10 && frequency < 233 && frequency > 0) //Else if frequency value is bigger than 185.10Hz and smaller than 233Hz and bigger than 0Hz, then
    {
     frequency = 0.00; //Reset frequency value
     for(i=0; i<=15; i++) //For 16 times repeat
     {
       Counterclockwise_GStep_Motor (); //Call function for G string step motor
     }
     delay(2); //Delay 2 msec
     GStep_Motor_Stop (); //Disable G string step motor
    } //else if (frequency...
    else if (frequency >=184.90 && frequency <=185.10)//Else if frequency value is bigger or equal with 184.90Hz and smaller or equal with 185.10Hz, then Gb string has been tuned
    {
     string_3 = 1; //Set string_3 flag to finish tuning with Gb string
     k = 21;  //Store value at k variable
```

```
        frequency = 0.00; //Reset frequency value
        lcd.clear(); //Clear LCD screen
        lcd.setCursor(1,0); //Set Cursor at third column and first line
        lcd.print("Hit Bb string"); //Print message at LCD screen
        delay(1000); //Delay 1 sec
        lcd.clear(); //Clear LCD screen
        lcd.setCursor(1,0); //Set Cursor at third column and first line
        lcd.print("Bb -> 233.08 Hz"); //Print message at LCD screen
        lcd.setCursor(7,1); //Set Cursor at first column and second line
        lcd.print("Hz   Note"); //print message at LCD screen
      } //else if (frequency...
      else //Else
      {
        frequency = 0.00;
      }
    } //while(string_3...*/
    while(string_2 != 1) //While string_2 flag isn't set repeat
    {
      if (k>20) //If k variable is bigger than 20, then
      {
        pinMode(39,OUTPUT); //Determine digital pin 39 as Output
        digitalWrite(39,LOW); //Set low voltage (GND) at pin 39
        delay(10); //Delay 10 msec
        pinMode(39,INPUT); //Determine digital pin 39 as Input
        k = 0; //Reset k variable
      } //if (k>20)
      FREQ_DETECTION(); //Call FREQ_DETECTION function (Change analog port for
the income signal)
      delay(500); //Delay 0.5 sec
      FREQ_DETECTION1(); //Call FREQ_DETECTION1 function (Calculate frequency
and stop interrupts by ADC)
      FREQ_DETECTION_STOP(); //Call FREQ_DETECTION_STOP function (Stop
frequency detection)
      Serial.println("Bb String"); //Print message at Serial Port
      Serial.println(frequency); //Print frequency value at Serial Port
      lcd.setCursor(0,1); //Set cursor at first column and second line
      lcd.print("       "); //Print message at lcd screen
      lcd.setCursor(0,1); //Set cursor at first column and second line
      lcd.print(frequency); //Print frequency value at LCD screen
      k++; //Increase variable k value by one
```

```
    if (frequency < 232.98 && frequency > 207 && frequency > 0) //If frequency value is smaller than 232.98Hz and bigger than 207Hz and bigger than 0Hz, then
    {
     frequency = 0.00; //Reset frequency value
     for (i=0; i<=20; i++) //For 21 times repeat
     {
      Clockwise_BStep_Motor (); //Call function for B string step motor
     }
     delay(2); //Delay 2 msec
     BStep_Motor_Stop (); //Disable B string step motor
    } //if (frequency...
    else if (frequency > 233.18 && frequency < 293 && frequency > 0) //Else if frequency value is bigger than 233.18Hz and smaller than 293Hz and bigger than 0Hz, then
    {
     frequency = 0.00; //Reset frequency value
     for(i=0; i<=20; i++) //For 21 times repeat
     {
      Counterclockwise_BStep_Motor (); //Call function for B string step motor
     }
     delay(2); //Delay 2 msec
     BStep_Motor_Stop (); //Disable B string step motor
    } //else if (frequency...
    else if (frequency >=232.98 && frequency <=233.18) //Else if frequency value is bigger or equal with 232.98Hz and smaller or equal with 233.18Hz, then Bb string has been tuned
    {
     string_2 = 1; //Set string_2 flag to finish tuning with Bb string
     k = 21;  //Store value at k variable
     frequency = 0.00; //Reset frequency value
     lcd.clear(); //Clear LCD screen
     lcd.setCursor(2,0); //Set Cursor at third column and first line
     lcd.print("Hit eb string"); //Print message at LCD screen
     delay(1000); //Delay 1 sec
     lcd.clear(); //Clear LCD screen
     lcd.setCursor(1,0); //Set Cursor at third column and first line
     lcd.print("eb -> 311.13 Hz"); //Print message at LCD screen
     lcd.setCursor(7,1); //Set Cursor at first column and second line
     lcd.print("Hz   Note"); //print message at LCD screen
    } //else if (frequency...
```

```
      else //Else
      {
        frequency = 0.00; //Reset frequency value
      }
    } //while(string_2...*/
    while(string_1 != 1) //While string_1 flag isn't set repeat
    {
      if (k>20) //If k variable is bigger than 20, then
      {
        pinMode(39,OUTPUT); //Determine digital pin 39 as Output
        digitalWrite(39,LOW); //Set low voltage (GND) at pin 39
        delay(10); //Delay 10 msec
        pinMode(39,INPUT); //Determine digital pin 39 as Input
        k = 0;  //Reset k variable
      } //if (k>20)
      FREQ_DETECTION(); //Call FREQ_DETECTION function (Change analog port for the income signal)
      delay(500); //Delay 0.5 sec
      FREQ_DETECTION1(); //Call FREQ_DETECTION1 function (Calculate frequency and stop interrupts by ADC)
      FREQ_DETECTION_STOP(); //Call FREQ_DETECTION_STOP function (Stop frequency detection)
      Serial.println("eb String"); //Print message at Serial Port
      Serial.println(frequency); //Print frequency value at Serial Port
      lcd.setCursor(0,1); //Set cursor at first column and second line
      lcd.print("       "); //Print message at lcd screen
      lcd.setCursor(0,1); //Set cursor at first column and second line
      lcd.print(frequency); //Print frequency value at LCD screen
      k++; //Increase variable k value by one
      if (frequency < 310.18 && frequency > 277 && frequency > 0) //If frequency value is smaller than 310.18Hz and bigger than 277Hz and bigger than 0Hz, then
      {
        frequency = 0.00; //Reset frequency value
        for (i=0; i<=20; i++) //For 21 times repeat
        {
          Clockwise_eStep_Motor (); //Call function for e string step motor
        }
        delay(2); //Delay 2 msec
        eStep_Motor_Stop (); //Disable e string step motor
      } //if (frequency...
```

```
      else if (frequency > 312.70 && frequency < 392 && frequency > 0) //Else if
frequency value is bigger than 312.70Hz and smaller than 392Hz and bigger than 0Hz,
then
      {
        frequency = 0.00; //Reset frequency value
        for(i=0; i<=20; i++) //For 21 times repeat
        {
          Counterclockwise_eStep_Motor (); //Call function for e string step motor
        }
        delay(2); //Delay 2 msec
        eStep_Motor_Stop (); //Disable e string step motor
      } //else if (frequency...
      else if (frequency >=310.18 && frequency <=312.70) //Else if frequency value is
bigger or equal with 310.18Hz and smaller or equal with 312.70Hz, then eb string has been
tuned
      {
        string_1 = 1; //Set string_1 flag to finish tuning with eb string
        k = 21;  //Store value at k variable
        frequency = 0.00; //Reset frequency value
      } //else if (frequency...
      else //Else
      {
        frequency = 0.00; //Reset frequency value
      }
    } //while(string_1...*/
    lcd.clear(); //Clear LCD screen
    lcd.setCursor(1,0); //Set cursor at second column and first line
    lcd.print("Tune finished!"); //Print message at LCD screen
    delay(5000); //Delay 5 sec
    Menu(); //Call Menu() function
  } //else if (menu==2)
```

Σύμφωνα με όσα αναφέρθηκαν προηγουμένως, εντός του μπλοκ εντολών του πλήκτρου SELECT, χρησιμοποιείται η τεχνική (if – else if) ώστε να μπορέσει το σύστημα να αναγνωρίσει την επιλογή που έχει κάνει ο χρήστης μέσα από το μενού επιλογής. Στον προηγούμενο πίνακα (Πίνακα 3.6) παρουσιάζεται ο κώδικας, ο οποίος αφορά την συνθήκη ελέγχου else if(menu = = 2), η οποία ελέγχει αν η μεταβλητή (menu) είναι ίση με την τιμή δύο και αντίστοιχα εκτελεί το μπλοκ εντολών για την επίτευξη του κουρδίσματος ενός ημιτόνου κάτω στην κιθάρα (Half-Step Down Tune). Στην περίπτωση όμως, που δεν ικανοποιείται η συνθήκη ελέγχου (else if), δηλαδή δεν έχει επιλεχτεί το κούρδισμα ενός ημιτόνου κάτω (Half-Step Down Tune), τότε εκτελείται η αμέσως επόμενη εντολή

ελέγχου else if(menu = = 3). Η συγκεκριμένη συνθήκη ελέγχου (else if) ελέγχει αν η τιμή της μεταβλητής (menu) είναι ίση με τρία. Συνεπώς, σε περίπτωση που ικανοποιείται η συγκεκριμένη συνθήκη, τότε εκτελείτε το μπλοκ εντολών της συγκεκριμένης συνθήκης ελέγχου (else if), γεγονός που σημαίνει ότι έχει επιλεχτεί το κούρδισμα της κιθάρας σε «πεσμένη» Ρε (Drop D Tune) (Πίνακας 3.7)

Πιο συγκεκριμένα, οι πρώτες εντολές που εκτελούνται στο μπλοκ εντολών της συνθήκης (else if) έχουν να κάνουν με το σβήσιμο χαρακτήρων στην LCD οθόνη μέσω της εντολής lcd.clear(), με την μετακίνηση του κέρσορα σε συγκεκριμένη στήλη και γραμμή μέσω της εντολής lcd.setCursor(), αλλά και με την εκτύπωση μηνυμάτων στην LCD οθόνης μέσω της εντολής lcd.print(“ “). Πιο συγκεκριμένα, πρώτα σβήνονται όλοι οι χαρακτήρες και ύστερα εμφανίζεται το μήνυμα (Drop D Tune Selected) για 1 sec εξαιτίας της εντολής καθυστέρησης delay(1000) που εκτελείται αμέσως μετά. Σκοπός του μηνύματος αυτού είναι να ενημερωθεί ο χρήστης από το σύστημα ότι το κούρδισμα που έχει επιλεχτεί για την κιθάρα είναι αυτό της «πεσμένης» Ρε (Drop D Tune). Έπειτα, σβήνονται ξανά οι χαρακτήρες από την LCD οθόνη και εμφανίζεται το μήνυμα (HIT D STRING) για 1 sec εξαιτίας της εντολής καθυστέρησης delay(1000) που εκτελείται αμέσως μετά. Το μήνυμα αυτό έχει σκοπό να προτρέψει τον χρήστη να διεγείρει την χορδή Ρε μπάσο (D). Ύστερα, σβήνονται οι χαρακτήρες από την LCD οθόνη και εμφανίζεται το μήνυμα (D -> 73.42 Hz) στην πρώτη γραμμή, το οποίο δηλώνει την επιθυμητή τιμή της θεμελιώδης συχνότητας ταλάντωσης της χορδής Ρε μπάσο (D). Επίσης, στην δεύτερη γραμμή εμφανίζεται το μήνυμα (Hz Note) σε συγκεκριμένες θέσεις, ώστε κατά την διάρκεια του κουρδίσματος να εκτυπώνεται η πραγματική συχνότητα πριν το (Hz) και η νότα στην οποία αντιστοιχεί η συχνότητα ταλάντωσης της χορδής πριν το (Note). Έπειτα, ακολουθούν κάποιες αρχικοποιήσεις μεταβλητών πριν την έναρξη της διαδικασίας κουρδίσματος της χορδής Ρε μπάσο (D). Πιο συγκεκριμένα, ο μετρητής (k) αρχικοποιείτε με την τιμή είκοσι ένα, ενώ στις σημαίες (flags) με ονομασίες (string_6, string_5, string_4, string_3, string_2 & string_1) γίνεται εναπόθεση της τιμής μηδέν.

Μετά την ολοκλήρωση των αρχικοποιήσεων, εφαρμόζεται ο βρόγχος επανάληψης (while()) ο οποίος στο μπλοκ εντολών του περιλαμβάνει τις απαραίτητες εντολές για το κούρδισμα της χορδής Ρε μπάσο (D). Για την εφαρμογή του βρόγχου επανάληψης (while), θα πρέπει εντός των παρενθέσεων να τοποθετηθεί μία συνθήκη. Όσο η συνθήκη αυτή ικανοποιείται, τόσο θα συνεχίσει να εκτελείται το μπλοκ εντολών του βρόγχου επανάληψης (while). Στην συγκεκριμένη περίπτωση, εντός των παρενθέσεων βρίσκεται η συνθήκη (string_6 != 1) η οποία σημαίνει ότι όσο η σημαία (flag) με όνομα (string_6) δεν είναι ίση με ένα, τόσο θα συνεχίζει να εκτελείται το μπλοκ εντολών του βρόγχου επανάληψης. Πιο συγκεκριμένα, όσο η σημαία (flag) με όνομα (string_6) είναι μηδέν αυτό σημαίνει ότι η χορδή Ρε μπάσο (D) είναι ξεκούρδιστη. Μόλις όμως η σημαία (flag) με όνομα (string_6) γίνει ένα, τότε αυτό σημαίνει ότι η χορδή Ρε μπάσο (D) έχει κουρδιστεί και το πρόγραμμα μπορεί να συνεχίσει στο κούρδισμα της επόμενης χορδής. Επειδή όμως η συγκεκριμένη σημαία (flag) έχει αρχικοποιηθεί με την τιμή μηδέν, το μπλοκ εντολών του βρόγχου επανάληψης (while) θα εκτελεστεί μία φορά τουλάχιστον.

Εντός λοιπόν του μπλοκ εντολών του βρόγχου επανάληψης (while), εκτελούνται οι εντολές οι οποίες είναι απαραίτητες για το κούρδισμα της χορδής Ρε μπάσο (D). Πιο συγκεκριμένα, υπάρχει μία συνθήκη ελέγχου (if), η οποία ελέγχει την τιμή του μετρητή (k) αν είναι μεγαλύτερη από είκοσι. Ο μετρητής (k) έχει αρχικοποιηθεί πριν τον βρόγχο

επανάληψης (while) με την τιμή είκοσι ένα, με αποτέλεσμα να εκτελείται το μπλοκ εντολών της συνθήκης ελέγχου (if). Πιο συγκεκριμένα, στο μπλοκ εντολών της συνθήκης (if) εκτελείτε πρώτα η εντολή pinMode(39, OUTPUT), η οποία καθορίζει τον ψηφιακό ακροδέκτη τριάντα εννιά ως έξοδο. Έπειτα, εκτελείται η εντολή digitalWrite(39, LOW), όπου ο ψηφιακός ακροδέκτης τριάντα εννιά μεταβαίνει σε κατάσταση χαμηλού δυναμικού για 10msec, διότι αμέσως μετά εκτελείται η εντολή χρονικής καθυστέρησης delay(10). Μετά εκτελείται η εντολή pinMode(39, INPUT), η οποία καθορίζει τον ψηφιακό ακροδέκτη τριάντα εννιά ως είσοδο. Ύστερα, το μπλοκ εντολών της συνθήκης ελέγχου (if) κλείνει με την εντολή (k = 0), όπου γίνεται μηδενισμός του μετρητή (k). Σκοπός της συγκεκριμένης συνθήκης ελέγχου, είναι η γείωση (GND) του σήματος που προέρχεται από το μικρόφωνο επαφής ανά τακτά χρονικά διαστήματα, ώστε να μην υπάρχουν παράσιτα τα οποία παραμορφώνουν το σήμα και οδηγούν σε λανθασμένους υπολογισμούς της θεμελιώδης συχνότητας ταλάντωσης.

Στην συνέχεια, μετά την συνθήκη ελέγχου (if), γίνεται κλήση της συνάρτησης FREQ_DETECTION(), η οποία ξεκινά την διαδικασία δειγματοληψίας του σήματος από το μικρόφωνο επαφής και αναλύεται σε επόμενη ενότητα. Η διαδικασία δειγματοληψίας, διαρκεί για 0.5sec εξαιτίας της εντολής καθυστέρησης delay(500) που εκτελείται αμέσως μετά. Ύστερα, γίνεται κλήση της συνάρτησης FREQ_DETECTION1(), η οποία αναλύεται σε επόμενη ενότητα και έχει σκοπό τόσο τον υπολογισμό της θεμελιώδης συχνότητας ταλάντωσης της χορδής όσο και την εύρεση της νότας στην οποία αντιστοιχεί η συγκεκριμένη συχνότητα. Έπειτα, γίνεται κλήση της συνάρτησης FREQ_DETECTION_STOP() η οποία αναλύεται και αυτή σε επόμενη ενότητα και αποσκοπεί στην παύση της δειγματοληψίας του σήματος από το μικρόφωνο επαφής απενεργοποιώντας τον αναλογικό-ψηφιακό μετατροπέα.

Μετά λοιπόν την ολοκλήρωση της δειγματοληψίας και αφού έχει υπολογιστεί η θεμελιώδης συχνότητα ταλάντωσης της χορδής, εμφανίζεται στην σειριακή οθόνη το μήνυμα (D String) και η τιμή της συχνότητας που έχει υπολογιστεί. Αυτό επιτυγχάνεται με την χρήση της εντολής Serial.println(). Στην συνέχεια, χρησιμοποιούνται οι εντολές lcd.setCursor() και lcd.print(“ “) η οποίες έχουν σκοπό την μετακίνηση του κέρσορα στην πρώτη στήλη της δεύτερης γραμμής της LCD οθόνης, το σβήσιμο της παλιάς τιμής της συχνότητας και την εμφάνιση της πραγματικής τιμής της συχνότητας που υπολογίστηκε. Με αυτόν τον τρόπο, ο χρήστης μπορεί να δει στην LCD οθόνη την πραγματική τιμή της συχνότητας ταλάντωσης της χορδής Ρε μπάσο (D). Αφότου ολοκληρωθεί η εμφάνιση της συχνότητας στην LCD οθόνη, εκτελείται η εντολή (k++), η οποία έχει σκοπό την αύξηση της τιμής του μετρητή (k) κατά μία μονάδα.

Εν συνεχεία, εφαρμόζεται η τεχνική ελέγχου (if – else if – else), με σκοπό τον έλεγχο της συχνότητας ταλάντωσης της χορδής. Πιο συγκεκριμένα, με την εντολή ελέγχου if (frequency < 73.37 && frequency > 65 && frequency > 0), ελέγχεται αν η τιμή της μεταβλητής (frequency) είναι μικρότερη από 73.37Hz και μεγαλύτερη από 65Hz και μεγαλύτερη από 0Hz. Επομένως, αν ικανοποιείται αυτή η συνθήκη σημαίνει ότι η συχνότητα ταλάντωσης της χορδής Ρε μπάσο (D) είναι χαμηλότερη από την επιθυμητή συχνότητα και για να διορθωθεί θα πρέπει να εκτελεστεί το μπλοκ εντολών της συνθήκης ελέγχου (if). Εντός λοιπόν του μπλοκ εντολών, γίνεται μηδενισμός της μεταβλητής (frequency) και ύστερα εκτελείται ένας βρόγχος επανάληψης (for). Ο βρόγχος επανάληψης (for) επαναλαμβάνει το μπλοκ εντολών του έντεκα φορές. Εντός του μπλοκ εντολών του βρόγχου επανάληψης (for) γίνεται κλήση της συνάρτησης

(Counterclockwise_EStep_Motor ()). Η συνάρτηση αυτή αναλύεται σε επόμενη ενότητα και έχει σκοπό την αντί-ωρολογιακή περιστροφή του άξονα του βηματικού κινητήρα και κατά συνέπεια την αντί-ωρολογιακή περιστροφή του κλειδιού της χορδής Ρε μπάσο (D). Μετά την ολοκλήρωση του βρόγχου επανάληψης (for), εκτελείται η εντολή delay(2) η οποία δημιουργεί μια χρονική καθυστέρηση των 2msec. Το μπλοκ εντολών της συνθήκης ελέγχου (if) ολοκληρώνεται με την κλήση της συνάρτησης (EStep_Motor_Stop ()), η οποία αναλύεται σε επόμενο κεφάλαιο και σκοπός της είναι να απενεργοποιήσει τον βηματικό κινητήρα. Στην περίπτωση όμως που δεν ικανοποιείται η συνθήκη ελέγχου (if), τότε εκτελείται η εντολή ελέγχου else if (frequency > 73.47 && frequency < 103 && frequency > 0), όπου ελέγχεται αν η τιμή της μεταβλητής (frequency) είναι μεγαλύτερη από 73.47Hz και μικρότερη από 103Hz και μεγαλύτερη από 0Hz. Συνεπώς, αν ικανοποιείται αυτή η συνθήκη σημαίνει ότι η συχνότητα ταλάντωσης της χορδής Ρε μπάσο (D) είναι υψηλότερη από την επιθυμητή συχνότητα και για να διορθωθεί θα πρέπει να εκτελεστεί το μπλοκ εντολών της συνθήκης ελέγχου (else if). Εντός λοιπόν του μπλοκ εντολών, γίνεται μηδενισμός της μεταβλητής (frequency) και ύστερα εκτελείται ένας βρόγχος επανάληψης (for). Εντός του μπλοκ εντολών του βρόγχου επανάληψης (for) γίνεται κλήση της συνάρτησης (Clockwise_EStep_Motor ()), η οποία αναλύεται σε επόμενη ενότητα. Σκοπός της συνάρτησης (Clockwise_EStep_Motor ()), είναι να περιστρέψει τον άξονα του βηματικού κινητήρα ωρολογιακά και κατά συνέπεια να περιστρέψει ωρολογιακά και το κλειδί της χορδής Ρε μπάσο (D). Αφότου ολοκληρωθεί ο βρόγχος επανάληψης (for), εκτελείται η εντολή delay(2) η οποία δημιουργεί μια χρονική καθυστέρηση των 2msec. Η τελευταία εντολή που εκτελείται στο μπλοκ εντολών της συνθήκης ελέγχου (else if) αφορά την κλήση της συνάρτησης (EStep_Motor_Stop ()), η οποία έχει σκοπό να απενεργοποιήσει τον βηματικό κινητήρα. Αν όμως δεν ικανοποιείται η προηγούμενη συνθήκη (else if), τότε εκτελείται η εντολή ελέγχου else if (frequency >=73.37 && frequency <=73.47), η οποία ελέγχει αν η τιμή της μεταβλητής (frequency) είναι μεγαλύτερη ή ίση από 73.37Hz και μικρότερη ή ίση από 73.47Hz. Επομένως, αν ικανοποιείται αυτή η συνθήκη τότε η πραγματική συχνότητα ταλάντωσης της χορδής Ρε μπάσο (D) βρίσκεται εντός του εύρους των επιθυμητών συχνοτήτων της χορδής και εκτελείται το αντίστοιχο μπλοκ εντολών. Εντός λοιπόν του συγκεκριμένου μπλοκ εντολών, αποθηκεύεται στην σημαία (flag) με όνομα (string_6) η τιμή ένα. Με αυτόν τον τρόπο υποδηλώνεται στο πρόγραμμα η ολοκλήρωση του κουρδίσματος της χορδής Ρε μπάσο (D). Ύστερα, εκτελείται η εντολή (k = 21), όπου πραγματοποιείται αρχικοποίηση του μετρητή (k) με την τιμή είκοσι ένα και αμέσως μετά, πραγματοποιείται μηδενισμός της μεταβλητής (frequency) μέσω της εντολής (frequency = 0.00). Έπειτα, εκτελούνται οι εντολές lcd.clear(), lcd.setCursor() και lcd.print(“ “) με την κατάλληλη σειρά, ώστε να εμφανιστεί το μήνυμα (Hit A string) στην LCD οθόνη. Το μήνυμα αυτό παραμένει στην LCD οθόνη για 1sec εξαιτίας της εντολής (delay1000) και σκοπό έχει να προτρέψει τον χρήστη να διεγείρει την επόμενη χορδή η οποία είναι η Λα (A). Εν συνεχεία, εκτελούνται ξανά οι εντολές lcd.clear(), lcd.setCursor() και lcd.print(“ “) με την κατάλληλη σειρά ώστε, να εμφανιστεί στην πρώτη γραμμή της LCD οθόνης το μήνυμα (A -> 110 Hz), το οποίο υποδηλώνει την επιθυμητή τιμή της συχνότητας της χορδής Λα (A) και στην δεύτερη γραμμή το μήνυμα (Hz Note) σε συγκεκριμένες θέσης. Σκοπός του μηνύματος της δεύτερης γραμμής είναι να εκτυπώνεται η πραγματική συχνότητα πριν το (Hz) και η νότα στην οποία αντιστοιχεί η συχνότητα ταλάντωσης της χορδής πριν το (Note) κατά την διάρκεια κουρδίσματος της χορδής Λα (A). Αν όμως, δεν ικανοποιείται καμία συνθήκη από τις εντολές ελέγχου (if-else if), τότε εκτελείται το μπλοκ εντολών της συνθήκης (else).

Πιο συγκεκριμένα, το μπλοκ εντολών της συνθήκης (else) περιλαμβάνει μία μόνο εντολή η οποία μηδενίζει την τιμή της μεταβλητής (frequency). Με αυτό τον τρόπο, αντιμετωπίζονται οι συχνότητες που μπορούν να προκύψουν από την ταυτόχρονη διέγερση δύο χορδών ή από την διέγερση λανθασμένης χορδής. Σε αυτό το σημείο ολοκληρώνεται το μπλοκ εντολών του βρόγχου επανάληψης (while), το οποίο έχει να κάνει με το κούρδισμα της χορδής Ρε μπάσο (D). Η διαδικασία που μόλις περιγράφτηκε, εκτελείτε ξανά μέχρι να πραγματοποιηθεί το κούρδισμα της χορδής Ρε μπάσο (D) εντός του εύρος των συχνοτήτων ταλάντωσης που έχει οριστεί για την συγκεκριμένη χορδή.

Με την ολοκλήρωση του κουρδίσματος της χορδής Ρε μπάσο (D), σειρά έχει το κούρδισμα της χορδής Λα (A). Αυτό επιτυγχάνεται με την χρήση του βρόγχου επανάληψης while(), ο οποίος στο μπλοκ εντολών του περιλαμβάνει τις απαραίτητες εντολές για το κούρδισμα της χορδής Λα (A). Η συνθήκη που βρίσκεται εντός των παρενθέσεων του βρόγχου επανάληψης (while) είναι η (string_5 != 1) η οποία σημαίνει ότι όσο η σημαία (flag) με όνομα (string_5) δεν είναι ίση με ένα, τόσο θα συνεχίζει να εκτελείται το μπλοκ εντολών του βρόγχου επανάληψης. Πιο συγκεκριμένα, όσο η σημαία (flag) με όνομα (string_5) είναι μηδέν αυτό σημαίνει ότι η χορδή Λα (A) είναι ξεκούρδιστη. Μόλις όμως η σημαία (flag) με όνομα (string_5) γίνει ένα, τότε αυτό σημαίνει ότι η χορδή Λα (A) έχει κουρδιστεί και το πρόγραμμα μπορεί να συνεχίσει στο κούρδισμα της επόμενης χορδής. Επειδή όμως η συγκεκριμένη σημαία (flag) έχει αρχικοποιηθεί με την τιμή μηδέν, το μπλοκ εντολών του βρόγχου επανάληψης (while) θα εκτελεστεί μία φορά τουλάχιστον.

Ουσιαστικά, εντός του μπλοκ εντολών του βρόγχου επανάληψης (while), εκτελούνται οι εντολές οι οποίες είναι απαραίτητες για το κούρδισμα της χορδής Λα (A). Όπως αναφέρθηκε και προηγουμένως, υπάρχει μία συνθήκη ελέγχου (if), η οποία ελέγχει την τιμή του μετρητή (k) αν είναι μεγαλύτερη από είκοσι. Ο μετρητής (k) έχει αρχικοποιηθεί στον προηγούμενο βρόγχο επανάληψης (while) με την τιμή είκοσι ένα, με αποτέλεσμα να εκτελείται το μπλοκ εντολών της συνθήκης ελέγχου (if). Ουσιαστικά, στο μπλοκ εντολών της συνθήκης (if) εκτελείτε πρώτα η εντολή pinMode(39, OUTPUT), η οποία καθορίζει τον ψηφιακό ακροδέκτη τριάντα εννιά ως έξοδο. Έπειτα, εκτελείται η εντολή digitalWrite(39, LOW), όπου ο ψηφιακός ακροδέκτης τριάντα εννιά μεταβαίνει σε κατάσταση χαμηλού δυναμικού για 10msec, διότι αμέσως μετά εκτελείται η εντολή χρονικής καθυστέρησης delay(10). Ύστερα, εκτελείται η εντολή pinMode(39, INPUT), η οποία καθορίζει τον ψηφιακό ακροδέκτη τριάντα εννιά ως είσοδο. Το μπλοκ εντολών της συνθήκης ελέγχου (if) ολοκληρώνεται με την εντολή (k = 0), όπου γίνεται μηδενισμός του μετρητή (k). Σκοπός της συγκεκριμένης συνθήκης ελέγχου είναι η γείωση (GND) του σήματος που προέρχεται από το μικρόφωνο επαφής ανά τακτά χρονικά διαστήματα, ώστε να μην υπάρχουν παράσιτα τα οποία παραμορφώνουν το σήμα και οδηγούν σε λανθασμένους υπολογισμούς της συχνότητας ταλάντωσης της χορδής.

Αφότου ολοκληρωθεί η εντολή της συνθήκης ελέγχου (if), γίνεται κλήση της συνάρτησης FREQ_DETECTION(), η οποία αναλύεται σε επόμενη ενότητα και αποσκοπεί στην εκκίνηση της διαδικασίας της δειγματοληψίας του σήματος από το μικρόφωνο επαφής. Η διαδικασία δειγματοληψίας, διαρκεί για 0.5sec εξαιτίας της εντολής καθυστέρησης delay(500) που εκτελείται αμέσως μετά. Ύστερα, γίνεται κλήση της συνάρτησης FREQ_DETECTION1(), η οποία έχει σκοπό τόσο τον υπολογισμό της θεμελιώδης συχνότητας ταλάντωσης της χορδής όσο και την εύρεση της νότας στην οποία αντιστοιχεί η συγκεκριμένη συχνότητα και η ανάλυσή της γίνεται σε επόμενη ενότητα.

Έπειτα, γίνεται κλήση της συνάρτησης FREQ_DETECTION_STOP() η οποία αναλύεται και αυτή σε επόμενη ενότητα και αποσκοπεί στην παύση της δειγματοληψίας του σήματος από το μικρόφωνο επαφής απενεργοποιώντας τον αναλογικό-ψηφιακό μετατροπέα.

Εν συνεχεία, αφού έχει υπολογιστεί η θεμελιώδης συχνότητα ταλάντωσης της χορδής, εμφανίζεται στην σειριακή οθόνη το μήνυμα (A String) και η τιμή της συχνότητας που έχει υπολογιστεί. Αυτό επιτυγχάνεται με την χρήση της εντολής Serial.println(). Στην συνέχεια, χρησιμοποιούνται οι εντολές lcd.setCursor() και lcd.print(" ") η οποίες έχουν σκοπό την μετακίνηση του κέρσορα στην πρώτη στήλη της δεύτερης γραμμής της LCD οθόνης, το σβήσιμο της παλιάς τιμής της συχνότητας και την εμφάνιση της πραγματικής τιμής της συχνότητας που υπολογίστηκε. Με αυτόν τον τρόπο, ο χρήστης μπορεί να δει στην LCD οθόνη την πραγματική τιμή της συχνότητας ταλάντωσης της χορδής Λα (A). Αφότου ολοκληρωθεί η εμφάνιση της συχνότητας στην LCD οθόνη, εκτελείται η εντολή (k++), η οποία έχει σκοπό την αύξηση της τιμής του μετρητή (k) κατά μία μονάδα.

Έπειτα, εφαρμόζεται η τεχνική ελέγχου (if – else if – else), η οποία αποσκοπεί στον έλεγχο της συχνότητας ταλάντωσης της χορδής. Ουσιαστικά, με την εκτέλεση της εντολής ελέγχου if (frequency < 109.80 && frequency > 92 && frequency > 0), ελέγχεται αν η τιμή της μεταβλητής (frequency) είναι μικρότερη από 109.8Hz και μεγαλύτερη από 92Hz και μεγαλύτερη από 0Hz. Συνεπώς, αν ικανοποιείται αυτή η συνθήκη σημαίνει ότι η συχνότητα ταλάντωσης της χορδής Λα (A) είναι χαμηλότερη από την επιθυμητή συχνότητα και για να διορθωθεί θα πρέπει να εκτελεστεί το μπλοκ εντολών της συνθήκης ελέγχου (if). Εντός λοιπόν του μπλοκ εντολών, γίνεται μηδενισμός της μεταβλητής (frequency) και ύστερα εκτελείται ένας βρόγχος επανάληψης (for) ο οποίος επαναλαμβάνει το μπλοκ εντολών του είκοσι ένα φορές. Όσον αφορά το μπλοκ εντολών του βρόγχου επανάληψης (for), γίνεται κλήση της συνάρτησης (Counterclockwise_AStep_Motor ()). Η συνάρτηση αυτή αναλύεται σε επόμενη ενότητα και έχει σκοπό την αντί-ωρολογιακή περιστροφή του άξονα του βηματικού κινητήρα και κατά συνέπεια την αντί-ωρολογιακή περιστροφή του κλειδιού της χορδής Λα (A). Μετά την ολοκλήρωση του βρόγχου επανάληψης (for), εκτελείται η εντολή delay(2) η οποία δημιουργεί μια χρονική καθυστέρηση των 2msec. Το μπλοκ εντολών της συνθήκης ελέγχου (if) ολοκληρώνεται με την κλήση της συνάρτησης (AStep_Motor_Stop ()), η οποία αναλύεται σε επόμενο κεφάλαιο και σκοπός της είναι να απενεργοποιήσει τον βηματικό κινητήρα. Αν όμως δεν ικανοποιείται η συνθήκη ελέγχου (if), τότε εκτελείται η εντολή ελέγχου else if (frequency > 110.21 && frequency < 130 && frequency > 0), όπου ελέγχεται αν η τιμή της μεταβλητής (frequency) είναι μεγαλύτερη από 110.21Hz και μικρότερη από 130Hz και μεγαλύτερη από 0Hz. Συνεπώς, αν ικανοποιείται αυτή η συνθήκη σημαίνει ότι η συχνότητα ταλάντωσης της χορδής Λα (A) είναι υψηλότερη από την επιθυμητή συχνότητα και για να διορθωθεί θα πρέπει να εκτελεστεί το μπλοκ εντολών της συνθήκης ελέγχου (else if). Ουσιαστικά, εντός του μπλοκ εντολών γίνεται μηδενισμός της μεταβλητής (frequency) και ύστερα εκτελείται ένας βρόγχος επανάληψης (for). Εντός του μπλοκ εντολών του βρόγχου επανάληψης (for) γίνεται κλήση της συνάρτησης (Clockwise_AStep_Motor ()), της οποίας η ανάλυση πραγματοποιείται σε επόμενη ενότητα. Η κλήση της συνάρτησης (Clockwise_AStep_Motor ()), αποσκοπεί στην ωρολογιακή περιστροφή του άξονα του βηματικού κινητήρα και κατά συνέπεια στην ωρολογιακή περιστροφή του κλειδιού της χορδής Λα (A). Μετά την ολοκλήρωση του βρόγχου επανάληψης (for), εκτελείται η εντολή delay(2) η οποία δημιουργεί μια χρονική

καθυστέρηση των 2msec. Η τελευταία εντολή που εκτελείται στο μπλοκ εντολών της συνθήκης ελέγχου (else if) αφορά την κλήση της συνάρτησης (AStep_Motor_Stop ()), η οποία αποσκοπεί στην απενεργοποίηση του βηματικού κινητήρα. Σε περίπτωση όμως που δεν ικανοποιείται η προηγούμενη συνθήκη (else if), τότε εκτελείται η εντολή ελέγχου else if (frequency >=109.80 && frequency <=110.21), η οποία ελέγχει αν η τιμή της μεταβλητής (frequency) είναι μεγαλύτερη ή ίση από 109.80Hz και μικρότερη ή ίση από 110.21Hz. Συνεπώς, αν ικανοποιείται αυτή η συνθήκη τότε η πραγματική συχνότητα ταλάντωσης της χορδής Λα (A) βρίσκεται εντός του εύρους των επιθυμητών συχνοτήτων της χορδής και εκτελείται το αντίστοιχο μπλοκ εντολών. Πιο συγκεκριμένα, εντός του συγκεκριμένου μπλοκ εντολών, αποθηκεύεται στην σημαία (flag) με όνομα (string_5) η τιμή ένα. Με αυτόν τον τρόπο υποδηλώνεται στο πρόγραμμα η ολοκλήρωση του κουρδίσματος της χορδής Λα (A). Έπειτα, εκτελείται η εντολή (k = 21), όπου πραγματοποιείται αρχικοποίηση του μετρητή (k) με την τιμή είκοσι ένα και αμέσως μετά, εκτελείται η εντολή (frequency = 0.00) η οποία μηδενίζει την τιμή της μεταβλητής (frequency). Ύστερα, πραγματοποιείται εκτέλεση των εντολών lcd.clear(), lcd.setCursor() και lcd.print(“ “) με την κατάλληλη σειρά, ώστε να εμφανιστεί το μήνυμα (Hit D string) στην LCD οθόνη. Το μήνυμα αυτό παραμένει στην LCD οθόνη για 1sec εξαιτίας της εντολής (delay1000) και έχει σκοπό να προτρέψει τον χρήστη να διεγείρει την επόμενη χορδή η οποία είναι η Ρε (D). Στην συνεχεία, εκτελούνται ξανά οι εντολές lcd.clear(), lcd.setCursor() και lcd.print(“ “) με την κατάλληλη σειρά ώστε, να εμφανιστεί στην πρώτη γραμμή της LCD οθόνης το μήνυμα (D -> 146.83 Hz), το οποίο υποδηλώνει την επιθυμητή τιμή της συχνότητας της χορδής Ρε (D) και στην δεύτερη γραμμή το μήνυμα (Hz Note) σε συγκεκριμένες θέσης. Πιο συγκεκριμένα, το μήνυμα της δεύτερης γραμμής αποσκοπεί στην εκτύπωση της πραγματική συχνότητα πριν το (Hz) και η νότα στην οποία αντιστοιχεί η συχνότητα ταλάντωσης της χορδής πριν το (Note) κατά την διάρκεια κουρδίσματος της χορδής Ρε (D). Στην περίπτωση όμως που δεν ικανοποιείται καμία συνθήκη από τις εντολές ελέγχου (if-else if), τότε εκτελείται το μπλοκ εντολών της συνθήκης (else). Ουσιαστικά, το μπλοκ εντολών της συνθήκης (else) αποτελείται από μία μόνο εντολή η οποία μηδενίζει την τιμή της μεταβλητής (frequency). Με αυτό τον τρόπο, αντιμετωπίζονται οι συχνότητες που μπορούν να προκύψουν από την ταυτόχρονη διέγερση δύο χορδών ή από την διέγερση λανθασμένης χορδής. Σε αυτό το σημείο ολοκληρώνεται το μπλοκ εντολών του βρόγχου επανάληψης (while), το οποίο έχει να κάνει με το κούρδισμα της χορδής Λα (A). Η διαδικασία που μόλις περιγράφτηκε, επαναλαμβάνεται μέχρι να επιτευχθεί το κούρδισμα της χορδής Λα (A) εντός του εύρος των συχνοτήτων ταλάντωσης που έχει οριστεί για την συγκεκριμένη χορδή.

Αφού ολοκληρωθεί το κούρδισμα της χορδής Λα (A), το πρόγραμμα συνεχίζει στην πραγματοποίηση του κουρδίσματος της χορδής Ρε (D). Αυτό επιτυγχάνεται με την χρήση του βρόγχου επανάληψης while(), ο οποίος στο μπλοκ εντολών του περιλαμβάνει τις απαραίτητες εντολές για το κούρδισμα της χορδής Ρε (D). Επιπλέον, η συνθήκη που βρίσκεται εντός των παρενθέσεων του βρόγχου επανάληψης (while) είναι η (string_4 != 1), η οποία δηλώνει ότι όσο η σημαία (flag) με όνομα (string_4) δεν είναι ίση με ένα, τόσο θα συνεχίζει να εκτελείται το μπλοκ εντολών του βρόγχου επανάληψης. Πιο συγκεκριμένα, όσο η σημαία (flag) με όνομα (string_4) είναι μηδέν αυτό σημαίνει ότι η χορδή Ρε (D) είναι ξεκούρδιστη. Μόλις όμως, η σημαία (flag) με όνομα (string_4) γίνει ένα, τότε αυτό σημαίνει ότι η χορδή Ρε (D) έχει κουρδιστεί και το πρόγραμμα μπορεί να συνεχίσει στο κούρδισμα της επόμενης χορδής. Επειδή όμως η συγκεκριμένη σημαία

(flag) έχει αρχικοποιηθεί με την τιμή μηδέν, το μπλοκ εντολών του βρόγχου επανάληψης (while) θα εκτελεστεί μία φορά τουλάχιστον.

Εντός λοιπόν του μπλοκ εντολών του βρόγχου επανάληψης (while), εκτελούνται οι εντολές οι οποίες είναι απαραίτητες για το κούρδισμα της χορδής Ρε (D). Πιο συγκεκριμένα, υπάρχει μία συνθήκη ελέγχου (if), η οποία ελέγχει την τιμή του μετρητή (k) αν είναι μεγαλύτερη από είκοσι. Ο μετρητής (k) έχει αρχικοποιηθεί στον προηγούμενο βρόγχο επανάληψης (while) με την τιμή είκοσι ένα, με αποτέλεσμα να εκτελείται το μπλοκ εντολών της συνθήκης ελέγχου (if). Πιο συγκεκριμένα, εντός του μπλοκ εντολών της συνθήκης (if), εκτελείτε πρώτα η εντολή pinMode(39, OUTPUT), η οποία καθορίζει τον ψηφιακό ακροδέκτη τριάντα εννιά ως έξοδο. Έπειτα, εκτελείται η εντολή digitalWrite(39, LOW), όπου ο ψηφιακός ακροδέκτης τριάντα εννιά μεταβαίνει σε κατάσταση χαμηλού δυναμικού για 10msec, διότι αμέσως μετά εκτελείται η εντολή χρονικής καθυστέρησης delay(10). Μετά εκτελείται η εντολή pinMode(39, INPUT), η οποία καθορίζει τον ψηφιακό ακροδέκτη τριάντα εννιά ως είσοδο. Ύστερα, το μπλοκ εντολών της συνθήκης ελέγχου (if) κλείνει με την εντολή (k = 0), όπου γίνεται μηδενισμός του μετρητή (k). Σκοπός της συγκεκριμένης συνθήκης ελέγχου, είναι η γείωση (GND) του σήματος που προέρχεται από το μικρόφωνο επαφής ανά τακτά χρονικά διαστήματα, ώστε να μην υπάρχουν παράσιτα τα οποία παραμορφώνουν το σήμα και οδηγούν σε λανθασμένους υπολογισμούς της θεμελιώδης συχνότητας ταλάντωσης.

Στην συνέχεια, μετά την συνθήκη ελέγχου (if), γίνεται κλήση της συνάρτησης FREQ_DETECTION(), η οποία ξεκινά την διαδικασία δειγματοληψίας του σήματος από το μικρόφωνο επαφής και αναλύεται σε επόμενη ενότητα. Η διαδικασία δειγματοληψίας, διαρκεί για 0.5sec εξαιτίας της εντολής καθυστέρησης delay(500) που εκτελείται αμέσως μετά. Ύστερα, γίνεται κλήση της συνάρτησης FREQ_DETECTION1(), η οποία αναλύεται σε επόμενη ενότητα και έχει σκοπό τόσο τον υπολογισμό της θεμελιώδης συχνότητας ταλάντωσης της χορδής όσο και την εύρεση της νότας στην οποία αντιστοιχεί η συγκεκριμένη συχνότητα. Έπειτα, γίνεται κλήση της συνάρτησης FREQ_DETECTION_STOP() η οποία αναλύεται και αυτή σε επόμενη ενότητα και αποσκοπεί στην παύση της δειγματοληψίας του σήματος από το μικρόφωνο επαφής απενεργοποιώντας τον αναλογικό-ψηφιακό μετατροπέα.

Μετά λοιπόν την ολοκλήρωση της δειγματοληψίας και αφού έχει υπολογιστεί η θεμελιώδης συχνότητα ταλάντωσης της χορδής, εμφανίζεται στην σειριακή οθόνη το μήνυμα (D String) και η τιμή της συχνότητας που έχει υπολογιστεί. Αυτό επιτυγχάνεται με την χρήση της εντολής Serial.println(). Στην συνέχεια, χρησιμοποιούνται οι εντολές lcd.setCursor() και lcd.print(" ") η οποίες έχουν σκοπό την μετακίνηση του κέρσορα στην πρώτη στήλη της δεύτερης γραμμής της LCD οθόνης, το σβήσιμο της παλιάς τιμής της συχνότητας και την εμφάνιση της πραγματικής τιμής της θεμελιώδης συχνότητας ταλάντωσης της χορδής που υπολογίστηκε. Με αυτόν τον τρόπο, ο χρήστης μπορεί να δει στην LCD οθόνη την πραγματική τιμή της συχνότητας ταλάντωσης της χορδής Ρε (D). Αφότου ολοκληρωθεί η εμφάνιση της συχνότητας στην LCD οθόνη, εκτελείται η εντολή (k++), η οποία έχει σκοπό την αύξηση της τιμής του μετρητή (k) κατά μία μονάδα.

Εν συνεχεία, εφαρμόζεται η τεχνική ελέγχου (if – else if – else), με σκοπό τον έλεγχο της συχνότητας ταλάντωσης της χορδής. Πιο συγκεκριμένα, με την εντολή ελέγχου if (frequency < 146.73 && frequency > 123 && frequency > 0), ελέγχεται αν η τιμή της μεταβλητής (frequency) είναι μικρότερη από 146.73Hz και μεγαλύτερη από 123Hz και μεγαλύτερη από 0Hz. Επομένως, αν ικανοποιείται αυτή η συνθήκη σημαίνει ότι η

συχνότητα ταλάντωσης της χορδής Ρε (D) είναι χαμηλότερη από την επιθυμητή συχνότητα και για να διορθωθεί θα πρέπει να εκτελεστεί το μπλοκ εντολών της συνθήκης ελέγχου (if). Εντός λοιπόν του μπλοκ εντολών, γίνεται μηδενισμός της μεταβλητής (frequency) και ύστερα εκτελείται ένας βρόγχος επανάληψης (for). Ο βρόγχος επανάληψης (for) επαναλαμβάνει το μπλοκ εντολών του τριάντα ένα φορές. Εντός του μπλοκ εντολών του βρόγχου επανάληψης (for) γίνεται κλήση της συνάρτησης (Counterclockwise_DStep_Motor ()). Η συνάρτηση αυτή αναλύεται σε επόμενη ενότητα και έχει σκοπό την αντί-ωρολογιακή περιστροφή του άξονα του βηματικού κινητήρα και κατά συνέπεια την αντί-ωρολογιακή περιστροφή του κλειδιού της χορδής Ρε (D). Μετά την ολοκλήρωση του βρόγχου επανάληψης (for), εκτελείται η εντολή delay(2) η οποία δημιουργεί μια χρονική καθυστέρηση των 2msec. Το μπλοκ εντολών της συνθήκης ελέγχου (if) ολοκληρώνεται με την κλήση της συνάρτησης (DStep_Motor_Stop ()), η οποία αναλύεται σε επόμενο κεφάλαιο και σκοπός της είναι να απενεργοποιήσει τον βηματικό κινητήρα. Στην περίπτωση όμως, που δεν ικανοποιείται η συνθήκη ελέγχου (if), τότε εκτελείται η εντολή ελέγχου else if (frequency > 146.93 && frequency < 174 && frequency > 0), όπου ελέγχεται αν η τιμή της μεταβλητής (frequency) είναι μεγαλύτερη από 146.93Hz και μικρότερη από 174Hz και μεγαλύτερη από 0Hz. Συνεπώς, αν ικανοποιείται αυτή η συνθήκη σημαίνει ότι η συχνότητα ταλάντωσης της χορδής Ρε (D) είναι υψηλότερη από την επιθυμητή συχνότητα και για να διορθωθεί θα πρέπει να εκτελεστεί το μπλοκ εντολών της συνθήκης ελέγχου (else if). Εντός λοιπόν του μπλοκ εντολών, γίνεται μηδενισμός της μεταβλητής (frequency) και ύστερα εκτελείται ένας βρόγχος επανάληψης (for). Εντός του μπλοκ εντολών του βρόγχου επανάληψης (for) γίνεται κλήση της συνάρτησης (Clockwise_DStep_Motor ()), η οποία αναλύεται σε επόμενη ενότητα. Σκοπός της συνάρτησης (Clockwise_DStep_Motor ()), είναι να περιστρέψει τον άξονα του βηματικού κινητήρα ωρολογιακά και κατά συνέπεια να περιστρέψει ωρολογιακά και το κλειδί της χορδής Ρε (D). Αφότου ολοκληρωθεί ο βρόγχος επανάληψης (for), εκτελείται η εντολή delay(2) η οποία δημιουργεί μια χρονική καθυστέρηση των 2msec. Η τελευταία εντολή που εκτελείται στο μπλοκ εντολών της συνθήκης ελέγχου (else if) αφορά την κλήση της συνάρτησης (DStep_Motor_Stop ()), η οποία έχει σκοπό να απενεργοποιήσει τον βηματικό κινητήρα. Αν όμως δεν ικανοποιείται η προηγούμενη συνθήκη (else if), τότε εκτελείται η εντολή ελέγχου else if (frequency >=146.73 && frequency <=146.93), η οποία ελέγχει αν η τιμή της μεταβλητής (frequency) είναι μεγαλύτερη ή ίση από 146.73Hz και μικρότερη ή ίση από 146.93Hz. Επομένως, αν ικανοποιείται αυτή η συνθήκη τότε η πραγματική συχνότητα ταλάντωσης της χορδής Ρε (D) βρίσκεται εντός του εύρους των επιθυμητών συχνοτήτων της χορδής και εκτελείται το αντίστοιχο μπλοκ εντολών. Εντός λοιπόν του συγκεκριμένου μπλοκ εντολών, αποθηκεύεται στην σημαία (flag) με όνομα (string_4) η τιμή ένα. Με αυτόν τον τρόπο υποδηλώνεται στο πρόγραμμα η ολοκλήρωση του κουρδίσματος της χορδής Ρε (D). Ύστερα, εκτελείται η εντολή (k = 21), όπου πραγματοποιείται αρχικοποίηση του μετρητή (k) με την τιμή είκοσι ένα και αμέσως μετά, πραγματοποιείται μηδενισμός της μεταβλητής (frequency) μέσω της εντολής (frequency = 0.00). Έπειτα, εκτελούνται οι εντολές lcd.clear(), lcd.setCursor() και lcd.print(“ “) με την κατάλληλη σειρά, ώστε να εμφανιστεί το μήνυμα (Hit G string) στην LCD οθόνη. Το μήνυμα αυτό παραμένει στην LCD οθόνη για 1sec εξαιτίας της εντολής (delay1000) και σκοπό έχει να προτρέψει τον χρήστη να διεγείρει την επόμενη χορδή η οποία είναι η Σολ (G). Εν συνεχεία, εκτελούνται ξανά οι εντολές lcd.clear(), lcd.setCursor() και lcd.print(“ “) με την κατάλληλη σειρά ώστε, να εμφανιστεί στην πρώτη γραμμή της LCD οθόνης το μήνυμα (G -> 196 Hz), το οποίο

υποδηλώνει την επιθυμητή τιμή της συχνότητας της χορδής Σολ (G) και στην δεύτερη γραμμή το μήνυμα (Hz Note) σε συγκεκριμένες θέσης. Σκοπός του μηνύματος της δεύτερης γραμμής είναι να εκτυπώνεται η πραγματική συχνότητα πριν το (Hz) και η νότα στην οποία αντιστοιχεί η συχνότητα ταλάντωσης της χορδής πριν το (Note) κατά την διάρκεια κουρδίσματος της χορδής Σολ (G). Αν όμως, δεν ικανοποιείται καμία συνθήκη από τις εντολές ελέγχου (if-else if), τότε εκτελείται το μπλοκ εντολών της συνθήκης (else). Πιο συγκεκριμένα, το μπλοκ εντολών της συνθήκης (else) περιλαμβάνει μία μόνο εντολή η οποία μηδενίζει την τιμή της μεταβλητής (frequency). Με αυτό τον τρόπο, αντιμετωπίζονται οι συχνότητες που μπορούν να προκύψουν από την ταυτόχρονη διέγερση δύο χορδών ή από την διέγερση λανθασμένης χορδής. Σε αυτό το σημείο ολοκληρώνεται το μπλοκ εντολών του βρόγχου επανάληψης (while), το οποίο έχει να κάνει με το κούρδισμα της χορδής Ρε (D). Η διαδικασία που μόλις περιγράφτηκε, εκτελείτε ξανά μέχρι να πραγματοποιηθεί το κούρδισμα της χορδής Ρε (D) εντός του εύρος των συχνοτήτων ταλάντωσης που έχει οριστεί για την συγκεκριμένη χορδή.

Με την ολοκλήρωση του κουρδίσματος της χορδής Ρε (D), σειρά έχει το κούρδισμα της χορδής Σολ (G). Αυτό επιτυγχάνεται με την χρήση του βρόγχου επανάληψης while(), ο οποίος στο μπλοκ εντολών του περιλαμβάνει τις απαραίτητες εντολές για το κούρδισμα της χορδής Σολ (G). Η συνθήκη που βρίσκεται εντός των παρενθέσεων του βρόγχου επανάληψης (while) είναι η (string_3 != 1) η οποία σημαίνει ότι όσο η σημαία (flag) με όνομα (string_3) δεν είναι ίση με ένα, τόσο θα συνεχίζει να εκτελείται το μπλοκ εντολών του βρόγχου επανάληψης. Πιο συγκεκριμένα, όσο η σημαία (flag) με όνομα (string_3) είναι μηδέν αυτό σημαίνει ότι η χορδή Σολ (G) είναι ξεκούρδιστη. Μόλις όμως η σημαία (flag) με όνομα (string_3) γίνει ένα, τότε αυτό σημαίνει ότι η χορδή Σολ (G) έχει κουρδιστεί και το πρόγραμμα μπορεί να συνεχίσει στο κούρδισμα της επόμενης χορδής. Επειδή όμως η συγκεκριμένη σημαία (flag) έχει αρχικοποιηθεί με την τιμή μηδέν, το μπλοκ εντολών του βρόγχου επανάληψης (while) θα εκτελεστεί μία φορά τουλάχιστον.

Ουσιαστικά, εντός του μπλοκ εντολών του βρόγχου επανάληψης (while), εκτελούνται οι εντολές οι οποίες είναι απαραίτητες για το κούρδισμα της χορδής Σολ (G). Όπως αναφέρθηκε και προηγουμένως, υπάρχει μία συνθήκη ελέγχου (if), η οποία ελέγχει την τιμή του μετρητή (k) αν είναι μεγαλύτερη από είκοσι. Ο μετρητής (k) έχει αρχικοποιηθεί στον προηγούμενο βρόγχο επανάληψης (while) με την τιμή είκοσι ένα, με αποτέλεσμα να εκτελείται το μπλοκ εντολών της συνθήκης ελέγχου (if). Ουσιαστικά, στο μπλοκ εντολών της συνθήκης (if) εκτελείτε πρώτα η εντολή pinMode(39, OUTPUT), η οποία καθορίζει τον ψηφιακό ακροδέκτη τριάντα εννιά ως έξοδο. Έπειτα, εκτελείται η εντολή digitalWrite(39, LOW), όπου ο ψηφιακός ακροδέκτης τριάντα εννιά μεταβαίνει σε κατάσταση χαμηλού δυναμικού για 10msec, διότι αμέσως μετά εκτελείται η εντολή χρονικής καθυστέρησης delay(10). Ύστερα, εκτελείται η εντολή pinMode(39, INPUT), η οποία καθορίζει τον ψηφιακό ακροδέκτη τριάντα εννιά ως είσοδο. Το μπλοκ εντολών της συνθήκης ελέγχου (if) ολοκληρώνεται με την εντολή (k = 0), όπου γίνεται μηδενισμός του μετρητή (k). Σκοπός της συγκεκριμένης συνθήκης ελέγχου είναι η γείωση (GND) του σήματος που προέρχεται από το μικρόφωνο επαφής ανά τακτά χρονικά διαστήματα, ώστε να μην υπάρχουν παράσιτα τα οποία παραμορφώνουν το σήμα και οδηγούν σε λανθασμένους υπολογισμούς της συχνότητας ταλάντωσης της χορδής..

Αφότου ολοκληρωθεί η εντολή της συνθήκης ελέγχου (if), γίνεται κλήση της συνάρτησης FREQ_DETECTION(), η οποία αναλύεται σε επόμενη ενότητα και αποσκοπεί στην εκκίνηση της διαδικασίας της δειγματοληψίας του σήματος από το

μικρόφωνο επαφής. Η διαδικασία δειγματοληψίας, διαρκεί για 0.5sec εξαιτίας της εντολής καθυστέρησης delay(500) που εκτελείται αμέσως μετά. Ύστερα, γίνεται κλήση της συνάρτησης FREQ_DETECTION1(), η οποία έχει σκοπό τόσο τον υπολογισμό της θεμελιώδης συχνότητας ταλάντωσης της χορδής όσο και την εύρεση της νότας στην οποία αντιστοιχεί η συγκεκριμένη συχνότητα και η ανάλυσή της γίνεται σε επόμενη ενότητα. Έπειτα, γίνεται κλήση της συνάρτησης FREQ_DETECTION_STOP() η οποία αναλύεται και αυτή σε επόμενη ενότητα και αποσκοπεί στην παύση της δειγματοληψίας του σήματος από το μικρόφωνο επαφής απενεργοποιώντας τον αναλογικό-ψηφιακό μετατροπέα.

Εν συνεχεία, αφού έχει υπολογιστεί η θεμελιώδης συχνότητα ταλάντωσης της χορδής, εμφανίζεται στην σειριακή οθόνη το μήνυμα (G String) και η τιμή της συχνότητας που έχει υπολογιστεί. Αυτό επιτυγχάνεται με την χρήση της εντολής Serial.println(). Στην συνέχεια, χρησιμοποιούνται οι εντολές lcd.setCursor() και lcd.print(" ") η οποίες έχουν σκοπό την μετακίνηση του κέρσορα στην πρώτη στήλη της δεύτερης γραμμής της LCD οθόνης, το σβήσιμο της παλιάς τιμής της συχνότητας και την εμφάνιση της πραγματικής τιμής της συχνότητας που υπολογίστηκε. Με αυτόν τον τρόπο, ο χρήστης μπορεί να δει στην LCD οθόνη την πραγματική τιμή της συχνότητας ταλάντωσης της χορδής Σολ (G). Αφότου ολοκληρωθεί η εμφάνιση της συχνότητας στην LCD οθόνη, εκτελείται η εντολή (k++), η οποία έχει σκοπό την αύξηση της τιμής του μετρητή (k) κατά μία μονάδα.

Έπειτα, εφαρμόζεται η τεχνική ελέγχου (if – else if – else), η οποία αποσκοπεί στον έλεγχο της συχνότητας ταλάντωσης της χορδής. Ουσιαστικά, με την εκτέλεση της εντολής ελέγχου if (frequency < 195.90 && frequency > 164 && frequency > 0), ελέγχεται αν η τιμή της μεταβλητής (frequency) είναι μικρότερη από 195.90Hz και μεγαλύτερη από 164Hz και μεγαλύτερη από 0Hz. Συνεπώς, αν ικανοποιείται αυτή η συνθήκη σημαίνει ότι η συχνότητα ταλάντωσης της χορδής Σολ (G) είναι χαμηλότερη από την επιθυμητή συχνότητα και για να διορθωθεί θα πρέπει να εκτελεστεί το μπλοκ εντολών της συνθήκης ελέγχου (if). Εντός λοιπόν του μπλοκ εντολών, γίνεται μηδενισμός της μεταβλητής (frequency) και ύστερα εκτελείται ένας βρόγχος επανάληψης (for) ο οποίος επαναλαμβάνει το μπλοκ εντολών του δέκα έξι φορές. Όσον αφορά το μπλοκ εντολών του βρόγχου επανάληψης (for), γίνεται κλήση της συνάρτησης (Counterclockwise_GStep_Motor ()). Η συνάρτηση αυτή αναλύεται σε επόμενη ενότητα και έχει σκοπό την αντί-ωρολογιακή περιστροφή του άξονα του βηματικού κινητήρα και κατά συνέπεια την αντί-ωρολογιακή περιστροφή του κλειδιού της χορδής Σολ (G). Μετά την ολοκλήρωση του βρόγχου επανάληψης (for), εκτελείται η εντολή delay(2) η οποία δημιουργεί μια χρονική καθυστέρηση των 2msec. Το μπλοκ εντολών της συνθήκης ελέγχου (if) ολοκληρώνεται με την κλήση της συνάρτησης (GStep_Motor_Stop ()), η οποία αναλύεται σε επόμενο κεφάλαιο και σκοπός της είναι να απενεργοποιήσει τον βηματικό κινητήρα. Αν όμως δεν ικανοποιείται η συνθήκη ελέγχου (if), τότε εκτελείται η εντολή ελέγχου else if (frequency > 196.25 && frequency < 233 && frequency > 0), όπου ελέγχεται αν η τιμή της μεταβλητής (frequency) είναι μεγαλύτερη από 196.25Hz και μικρότερη από 233Hz και μεγαλύτερη από 0Hz. Συνεπώς, αν ικανοποιείται αυτή η συνθήκη σημαίνει ότι η συχνότητα ταλάντωσης της χορδής Σολ (G) είναι υψηλότερη από την επιθυμητή συχνότητα και για να διορθωθεί θα πρέπει να εκτελεστεί το μπλοκ εντολών της συνθήκης ελέγχου (else if). Ουσιαστικά, εντός του μπλοκ εντολών γίνεται μηδενισμός της μεταβλητής (frequency) και ύστερα εκτελείται ένας βρόγχος επανάληψης (for). Εντός του μπλοκ εντολών του βρόγχου επανάληψης (for) γίνεται κλήση της συνάρτησης

(Clockwise_GStep_Motor ()), της οποίας η ανάλυση πραγματοποιείται σε επόμενη ενότητα. Η κλήση της συνάρτησης (Clockwise_GStep_Motor ()), αποσκοπεί στην ωρολογιακή περιστροφή του άξονα του βηματικού κινητήρα και κατά συνέπεια στην ωρολογιακή περιστροφή του κλειδιού της χορδής Σολ (G). Μετά την ολοκλήρωση του βρόγχου επανάληψης (for), εκτελείται η εντολή delay(2) η οποία δημιουργεί μια χρονική καθυστέρηση των 2msec. Η τελευταία εντολή που εκτελείται στο μπλοκ εντολών της συνθήκης ελέγχου (else if) αφορά την κλήση της συνάρτησης (GStep_Motor_Stop ()), η οποία αποσκοπεί στην απενεργοποίηση του βηματικού κινητήρα. Σε περίπτωση όμως που δεν ικανοποιείται η προηγούμενη συνθήκη (else if), τότε εκτελείται η εντολή ελέγχου else if (frequency >=195.90 && frequency <=196.25), η οποία ελέγχει αν η τιμή της μεταβλητής (frequency) είναι μεγαλύτερη ή ίση από 195.90Hz και μικρότερη ή ίση από 196.25Hz. Συνεπώς, αν ικανοποιείται αυτή η συνθήκη τότε η πραγματική συχνότητα ταλάντωσης της χορδής Σολ (G) βρίσκεται εντός του εύρους των επιθυμητών συχνοτήτων της χορδής και εκτελείται το αντίστοιχο μπλοκ εντολών. Πιο συγκεκριμένα, εντός του συγκεκριμένου μπλοκ εντολών, αποθηκεύεται στην σημαία (flag) με όνομα (string_3) η τιμή ένα. Με αυτόν τον τρόπο υποδηλώνεται στο πρόγραμμα η ολοκλήρωση του κουρδίσματος της χορδής Σολ (G). Έπειτα, εκτελείται η εντολή (k = 21), όπου πραγματοποιείται αρχικοποίηση του μετρητή (k) με την τιμή είκοσι ένα και αμέσως μετά, εκτελείται η εντολή (frequency = 0.00) η οποία μηδενίζει την τιμή της μεταβλητής (frequency). Ύστερα, πραγματοποιείται εκτέλεση των εντολών lcd.clear(), lcd.setCursor() και lcd.print(“ “) με την κατάλληλη σειρά, ώστε να εμφανιστεί το μήνυμα (Hit B string) στην LCD οθόνη. Το μήνυμα αυτό παραμένει στην LCD οθόνη για 1sec εξαιτίας της εντολής (delay1000) και έχει σκοπό να προτρέψει τον χρήστη να διεγείρει την επόμενη χορδή η οποία είναι η Σι (B). Στην συνεχεία, εκτελούνται ξανά οι εντολές lcd.clear(), lcd.setCursor() και lcd.print(“ “) με την κατάλληλη σειρά ώστε, να εμφανιστεί στην πρώτη γραμμή της LCD οθόνης το μήνυμα (B -> 246.94 Hz), το οποίο υποδηλώνει την επιθυμητή τιμή της συχνότητας της χορδής Σι (B) και στην δεύτερη γραμμή το μήνυμα (Hz Note) σε συγκεκριμένες θέσης. Πιο συγκεκριμένα, το μήνυμα της δεύτερης γραμμής αποσκοπεί στην εκτύπωση της πραγματική συχνότητα πριν το (Hz) και η νότα στην οποία αντιστοιχεί η συχνότητα ταλάντωσης της χορδής πριν το (Note) κατά την διάρκεια κουρδίσματος της χορδής Σι (B). Στην περίπτωση όμως που δεν ικανοποιείται καμία συνθήκη από τις εντολές ελέγχου (if-else if), τότε εκτελείται το μπλοκ εντολών της συνθήκης (else). Ουσιαστικά, το μπλοκ εντολών της συνθήκης (else) αποτελείται από μία μόνο εντολή η οποία μηδενίζει την τιμή της μεταβλητής (frequency). Με αυτό τον τρόπο, αντιμετωπίζονται οι συχνότητες που μπορούν να προκύψουν από την ταυτόχρονη διέγερση δύο χορδών ή από την διέγερση λανθασμένης χορδής. Σε αυτό το σημείο ολοκληρώνεται το μπλοκ εντολών του βρόγχου επανάληψης (while), το οποίο έχει να κάνει με το κούρδισμα της χορδής Σολ (G). Η διαδικασία που μόλις περιγράφτηκε, επαναλαμβάνεται μέχρι να επιτευχθεί το κούρδισμα της χορδής Σολ (G) εντός του εύρος των συχνοτήτων ταλάντωσης που έχει οριστεί για την συγκεκριμένη χορδή.

Αφού ολοκληρωθεί το κούρδισμα της χορδής Σολ (G), το πρόγραμμα συνεχίζει στην πραγματοποίηση του κουρδίσματος της χορδής Σι (B). Αυτό επιτυγχάνεται με την χρήση του βρόγχου επανάληψης while(), ο οποίος στο μπλοκ εντολών του περιλαμβάνει τις απαραίτητες εντολές για το κούρδισμα της χορδής Σι (B). Επιπλέον, η συνθήκη που βρίσκεται εντός των παρενθέσεων του βρόγχου επανάληψης (while) είναι η (string_2 != 1), η οποία δηλώνει ότι όσο η σημαία (flag) με όνομα (string_2) δεν είναι ίση με ένα, τόσο θα συνεχίζει να εκτελείται το μπλοκ εντολών του βρόγχου επανάληψης. Πιο

συγκεκριμένα, όσο η σημαία (flag) με όνομα (string_2) είναι μηδέν αυτό σημαίνει ότι η χορδή Σι (B) είναι ξεκούρδιστη. Μόλις όμως, η σημαία (flag) με όνομα (string_2) γίνει ένα, τότε αυτό σημαίνει ότι η χορδή Σι (B) έχει κουρδιστεί και το πρόγραμμα μπορεί να συνεχίσει στο κούρδισμα της επόμενης χορδής. Επειδή όμως η συγκεκριμένη σημαία (flag) έχει αρχικοποιηθεί με την τιμή μηδέν, το μπλοκ εντολών του βρόγχου επανάληψης (while) θα εκτελεστεί μία φορά τουλάχιστον.

Εντός λοιπόν του μπλοκ εντολών του βρόγχου επανάληψης (while), εκτελούνται οι εντολές οι οποίες είναι απαραίτητες για το κούρδισμα της χορδής Σι (B). Πιο συγκεκριμένα, υπάρχει μία συνθήκη ελέγχου (if), η οποία ελέγχει την τιμή του μετρητή (k) αν είναι μεγαλύτερη από είκοσι. Ο μετρητής (k) έχει αρχικοποιηθεί στον προηγούμενο βρόγχο επανάληψης (while) με την τιμή είκοσι ένα, με αποτέλεσμα να εκτελείται το μπλοκ εντολών της συνθήκης ελέγχου (if). Πιο συγκεκριμένα, εντός του μπλοκ εντολών της συνθήκης (if), εκτελείτε πρώτα η εντολή pinMode(39, OUTPUT), η οποία καθορίζει τον ψηφιακό ακροδέκτη τριάντα εννιά ως έξοδο. Έπειτα, εκτελείται η εντολή digitalWrite(39, LOW), όπου ο ψηφιακός ακροδέκτης τριάντα εννιά μεταβαίνει σε κατάσταση χαμηλού δυναμικού για 10msec, διότι αμέσως μετά εκτελείται η εντολή χρονικής καθυστέρησης delay(10). Μετά εκτελείται η εντολή pinMode(39, INPUT), η οποία καθορίζει τον ψηφιακό ακροδέκτη τριάντα εννιά ως είσοδο. Ύστερα, το μπλοκ εντολών της συνθήκης ελέγχου (if) κλείνει με την εντολή (k = 0), όπου γίνεται μηδενισμός του μετρητή (k). Σκοπός της συγκεκριμένης συνθήκης ελέγχου, είναι η γείωση (GND) του σήματος που προέρχεται από το μικρόφωνο επαφής ανά τακτά χρονικά διαστήματα, ώστε να μην υπάρχουν παράσιτα τα οποία παραμορφώνουν το σήμα και οδηγούν σε λανθασμένους υπολογισμούς της θεμελιώδης συχνότητας ταλάντωσης.

Στην συνέχεια, μετά την συνθήκη ελέγχου (if), γίνεται κλήση της συνάρτησης FREQ_DETECTION(), η οποία ξεκινά την διαδικασία δειγματοληψίας του σήματος από το μικρόφωνο επαφής και αναλύεται σε επόμενη ενότητα. Η διαδικασία δειγματοληψίας, διαρκεί για 0.5sec εξαιτίας της εντολής καθυστέρησης delay(500) που εκτελείται αμέσως μετά. Ύστερα, γίνεται κλήση της συνάρτησης FREQ_DETECTION1(), η οποία αναλύεται σε επόμενη ενότητα και έχει σκοπό τόσο τον υπολογισμό της θεμελιώδης συχνότητας ταλάντωσης της χορδής όσο και την εύρεση της νότας στην οποία αντιστοιχεί η συγκεκριμένη συχνότητα. Έπειτα, γίνεται κλήση της συνάρτησης FREQ_DETECTION_STOP() η οποία αναλύεται και αυτή σε επόμενη ενότητα και αποσκοπεί στην παύση της δειγματοληψίας του σήματος από το μικρόφωνο επαφής απενεργοποιώντας τον αναλογικό-ψηφιακό μετατροπέα.

Μετά λοιπόν την ολοκλήρωση της δειγματοληψίας και αφού έχει υπολογιστεί η θεμελιώδης συχνότητα ταλάντωσης της χορδής, εμφανίζεται στην σειριακή οθόνη το μήνυμα (B String) και η τιμή της συχνότητας που έχει υπολογιστεί. Αυτό επιτυγχάνεται με την χρήση της εντολής Serial.println(). Στην συνέχεια, χρησιμοποιούνται οι εντολές lcd.setCursor() και lcd.print(“ “) η οποίες έχουν σκοπό την μετακίνηση του κέρσορα στην πρώτη στήλη της δεύτερης γραμμής της LCD οθόνης, το σβήσιμο της παλιάς τιμής της συχνότητας και την εμφάνιση της πραγματικής τιμής της θεμελιώδης συχνότητας ταλάντωσης της χορδής που υπολογίστηκε. Με αυτόν τον τρόπο, ο χρήστης μπορεί να δει στην LCD οθόνη την πραγματική τιμή της συχνότητας ταλάντωσης της χορδής Σι (B). Αφότου ολοκληρωθεί η εμφάνιση της συχνότητας στην LCD οθόνη, εκτελείται η εντολή (k++), η οποία έχει σκοπό την αύξηση της τιμής του μετρητή (k) κατά μία μονάδα.

Εν συνεχεία, εφαρμόζεται η τεχνική ελέγχου (if – else if – else), με σκοπό τον έλεγχο της συχνότητας ταλάντωσης της χορδής. Πιο συγκεκριμένα, με την εντολή ελέγχου if (frequency < 246.54 && frequency > 207 && frequency > 0), ελέγχεται αν η τιμή της μεταβλητής (frequency) είναι μικρότερη από 246.54Hz και μεγαλύτερη από 207Hz και μεγαλύτερη από 0Hz. Επομένως, αν ικανοποιείται αυτή η συνθήκη σημαίνει ότι η συχνότητα ταλάντωσης της χορδής Σι (B) είναι χαμηλότερη από την επιθυμητή συχνότητα και για να διορθωθεί θα πρέπει να εκτελεστεί το μπλοκ εντολών της συνθήκης ελέγχου (if). Εντός λοιπόν του μπλοκ εντολών, γίνεται μηδενισμός της μεταβλητής (frequency) και ύστερα εκτελείται ένας βρόγχος επανάληψης (for). Ο βρόγχος επανάληψης (for) επαναλαμβάνει το μπλοκ εντολών του είκοσι ένα φορές. Εντός του μπλοκ εντολών του βρόγχου επανάληψης (for) γίνεται κλήση της συνάρτησης (Counterclockwise_BStep_Motor ()). Η συνάρτηση αυτή αναλύεται σε επόμενη ενότητα και έχει σκοπό την αντί-ωρολογιακή περιστροφή του άξονα του βηματικού κινητήρα και κατά συνέπεια την αντί-ωρολογιακή περιστροφή του κλειδιού της χορδής Σι (B). Μετά την ολοκλήρωση του βρόγχου επανάληψης (for), εκτελείται η εντολή delay(2) η οποία δημιουργεί μια χρονική καθυστέρηση των 2msec. Το μπλοκ εντολών της συνθήκης ελέγχου (if) ολοκληρώνεται με την κλήση της συνάρτησης (BStep_Motor_Stop ()), η οποία αναλύεται σε επόμενο κεφάλαιο και σκοπός της είναι να απενεργοποιήσει τον βηματικό κινητήρα. Στην περίπτωση όμως, που δεν ικανοποιείται η συνθήκη ελέγχου (if), τότε εκτελείται η εντολή ελέγχου else if (frequency > 247.04 && frequency < 293 && frequency > 0), όπου ελέγχεται αν η τιμή της μεταβλητής (frequency) είναι μεγαλύτερη από 247.04Hz και μικρότερη από 293Hz και μεγαλύτερη από 0Hz. Συνεπώς, αν ικανοποιείται αυτή η συνθήκη σημαίνει ότι η συχνότητα ταλάντωσης της χορδής Σι (B) είναι υψηλότερη από την επιθυμητή συχνότητα και για να διορθωθεί θα πρέπει να εκτελεστεί το μπλοκ εντολών της συνθήκης ελέγχου (else if). Εντός λοιπόν του μπλοκ εντολών, γίνεται μηδενισμός της μεταβλητής (frequency) και ύστερα εκτελείται ένας βρόγχος επανάληψης (for). Εντός του μπλοκ εντολών του βρόγχου επανάληψης (for) γίνεται κλήση της συνάρτησης (Clockwise_BStep_Motor ()), η οποία αναλύεται σε επόμενη ενότητα. Σκοπός της συνάρτησης (Clockwise_BStep_Motor ()), είναι να περιστρέψει τον άξονα του βηματικού κινητήρα ωρολογιακά και κατά συνέπεια να περιστρέψει ωρολογιακά και το κλειδί της χορδής Σι (B). Αφότου ολοκληρωθεί ο βρόγχος επανάληψης (for), εκτελείται η εντολή delay(2) η οποία δημιουργεί μια χρονική καθυστέρηση των 2msec. Η τελευταία εντολή που εκτελείται στο μπλοκ εντολών της συνθήκης ελέγχου (else if) αφορά την κλήση της συνάρτησης (BStep_Motor_Stop ()), η οποία έχει σκοπό να απενεργοποιήσει τον βηματικό κινητήρα. Αν όμως δεν ικανοποιείται η προηγούμενη συνθήκη (else if), τότε εκτελείται η εντολή ελέγχου else if (frequency >= frequency >=246.55 && frequency <=247.04), η οποία ελέγχει αν η τιμή της μεταβλητής (frequency) είναι μεγαλύτερη ή ίση από 246.55Hz και μικρότερη ή ίση από 247.04Hz. Επομένως, αν ικανοποιείται αυτή η συνθήκη τότε η πραγματική συχνότητα ταλάντωσης της χορδής Σι (B) βρίσκεται εντός του εύρους των επιθυμητών συχνοτήτων της χορδής και εκτελείται το αντίστοιχο μπλοκ εντολών. Εντός λοιπόν του συγκεκριμένου μπλοκ εντολών, αποθηκεύεται στην σημαία (flag) με όνομα (string_2) η τιμή ένα. Με αυτόν τον τρόπο υποδηλώνεται στο πρόγραμμα η ολοκλήρωση του κουρδίσματος της χορδής Σι (B). Ύστερα, εκτελείται η εντολή (k = 21), όπου πραγματοποιείται αρχικοποίηση του μετρητή (k) με την τιμή είκοσι ένα και αμέσως μετά, πραγματοποιείται μηδενισμός της μεταβλητής (frequency) μέσω της εντολής (frequency = 0.00). Έπειτα, εκτελούνται οι εντολές lcd.clear(), lcd.setCursor() και lcd.print(“ “) με την κατάλληλη σειρά, ώστε να εμφανιστεί

το μήνυμα (Hit E string) στην LCD οθόνη. Το μήνυμα αυτό παραμένει στην LCD οθόνη για 1sec εξαιτίας της εντολής (delay1000) και σκοπό έχει να προτρέψει τον χρήστη να διεγείρει την επόμενη χορδή η οποία είναι η Μι καντίνι (e). Εν συνεχεία, εκτελούνται ξανά οι εντολές lcd.clear(), lcd.setCursor() και lcd.print(“ “) με την κατάλληλη σειρά ώστε, να εμφανιστεί στην πρώτη γραμμή της LCD οθόνης το μήνυμα (e -> 329.63 Hz), το οποίο υποδηλώνει την επιθυμητή τιμή της συχνότητας της χορδής Μι (e) και στην δεύτερη γραμμή το μήνυμα (Hz Note) σε συγκεκριμένες θέσης. Σκοπός του μηνύματος της δεύτερης γραμμής είναι να εκτυπώνεται η πραγματική συχνότητα πριν το (Hz) και η νότα στην οποία αντιστοιχεί η συχνότητα ταλάντωσης της χορδής πριν το (Note) κατά την διάρκεια κουρδίσματος της χορδής Μι (e). Αν όμως, δεν ικανοποιείται καμία συνθήκη από τις εντολές ελέγχου (if-else if), τότε εκτελείται το μπλοκ εντολών της συνθήκης (else). Πιο συγκεκριμένα, το μπλοκ εντολών της συνθήκης (else) περιλαμβάνει μία μόνο εντολή η οποία μηδενίζει την τιμή της μεταβλητής (frequency). Με αυτό τον τρόπο, αντιμετωπίζονται οι συχνότητες που μπορούν να προκύψουν από την ταυτόχρονη διέγερση δύο χορδών ή από την διέγερση λανθασμένης χορδής. Σε αυτό το σημείο ολοκληρώνεται το μπλοκ εντολών του βρόγχου επανάληψης (while), το οποίο έχει να κάνει με το κούρδισμα της χορδής Σι (B). Η διαδικασία που μόλις περιγράφτηκε, εκτελείτε ξανά μέχρι να πραγματοποιηθεί το κούρδισμα της χορδής Σι (B) εντός του εύρος των συχνοτήτων ταλάντωσης που έχει οριστεί για την συγκεκριμένη χορδή.

Με την ολοκλήρωση του κουρδίσματος της χορδής Σι (B), σειρά έχει το κούρδισμα της χορδής Μι καντίνι (e). Αυτό επιτυγχάνεται με την χρήση του βρόγχου επανάληψης while(), ο οποίος στο μπλοκ εντολών του περιλαμβάνει τις απαραίτητες εντολές για το κούρδισμα της χορδής Μι καντίνι (e). Η συνθήκη που βρίσκεται εντός των παρενθέσεων του βρόγχου επανάληψης (while) είναι η (string_1 != 1) η οποία σημαίνει ότι όσο η σημαία (flag) με όνομα (string_1) δεν είναι ίση με ένα, τόσο θα συνεχίζει να εκτελείται το μπλοκ εντολών του βρόγχου επανάληψης. Πιο συγκεκριμένα, όσο η σημαία (flag) με όνομα (string_1) είναι μηδέν αυτό σημαίνει ότι η χορδή Μι καντίνι (e) είναι ξεκούρδιστη. Μόλις όμως η σημαία (flag) με όνομα (string_1) γίνει ένα, τότε αυτό σημαίνει ότι η χορδή Μι καντίνι (e) έχει κουρδιστεί. Επειδή όμως η συγκεκριμένη σημαία (flag) έχει αρχικοποιηθεί με την τιμή μηδέν, το μπλοκ εντολών του βρόγχου επανάληψης (while) θα εκτελεστεί μία φορά τουλάχιστον.

Ουσιαστικά, εντός του μπλοκ εντολών του βρόγχου επανάληψης (while), εκτελούνται οι εντολές οι οποίες είναι απαραίτητες για το κούρδισμα της χορδής Μι καντίνι (e). Όπως αναφέρθηκε και προηγουμένως, υπάρχει μία συνθήκη ελέγχου (if), η οποία ελέγχει την τιμή του μετρητή (k) αν είναι μεγαλύτερη από είκοσι. Ο μετρητής (k) έχει αρχικοποιηθεί στον προηγούμενο βρόγχο επανάληψης (while) με την τιμή είκοσι ένα, με αποτέλεσμα να εκτελείται το μπλοκ εντολών της συνθήκης ελέγχου (if). Ουσιαστικά, στο μπλοκ εντολών της συνθήκης (if) εκτελείτε πρώτα η εντολή pinMode(39, OUTPUT), η οποία καθορίζει τον ψηφιακό ακροδέκτη τριάντα εννιά ως έξοδο. Έπειτα, εκτελείται η εντολή digitalWrite(39, LOW), όπου ο ψηφιακός ακροδέκτης τριάντα εννιά μεταβαίνει σε κατάσταση χαμηλού δυναμικού για 10msec, διότι αμέσως μετά εκτελείται η εντολή χρονικής καθυστέρησης delay(10). Ύστερα, εκτελείται η εντολή pinMode(39, INPUT), η οποία καθορίζει τον ψηφιακό ακροδέκτη τριάντα εννιά ως είσοδο. Το μπλοκ εντολών της συνθήκης ελέγχου (if) ολοκληρώνεται με την εντολή (k = 0), όπου γίνεται μηδενισμός του μετρητή (k). Σκοπός της συγκεκριμένης συνθήκης ελέγχου είναι η γείωση (GND) του σήματος που προέρχεται από το μικρόφωνο επαφής ανά τακτά χρονικά διαστήματα, ώστε

να μην υπάρχουν παράσιτα τα οποία παραμορφώνουν το σήμα και οδηγούν σε λανθασμένους υπολογισμούς της συχνότητας ταλάντωσης της χορδής..

Αφότου ολοκληρωθεί η εντολή της συνθήκης ελέγχου (if), γίνεται κλήση της συνάρτησης FREQ_DETECTION(), η οποία αναλύεται σε επόμενη ενότητα και αποσκοπεί στην εκκίνηση της διαδικασίας της δειγματοληψίας του σήματος από το μικρόφωνο επαφής. Η διαδικασία δειγματοληψίας, διαρκεί για 0.5sec εξαιτίας της εντολής καθυστέρησης delay(500) που εκτελείται αμέσως μετά. Ύστερα, γίνεται κλήση της συνάρτησης FREQ_DETECTION1(), η οποία έχει σκοπό τόσο τον υπολογισμό της θεμελιώδης συχνότητας ταλάντωσης της χορδής όσο και την εύρεση της νότας στην οποία αντιστοιχεί η συγκεκριμένη συχνότητα και η ανάλυσή της γίνεται σε επόμενη ενότητα. Έπειτα, γίνεται κλήση της συνάρτησης FREQ_DETECTION_STOP() η οποία αναλύεται και αυτή σε επόμενη ενότητα και αποσκοπεί στην παύση της δειγματοληψίας του σήματος από το μικρόφωνο επαφής απενεργοποιώντας τον αναλογικό-ψηφιακό μετατροπέα.

Εν συνεχεία, αφού έχει υπολογιστεί η θεμελιώδης συχνότητα ταλάντωσης της χορδής, εμφανίζεται στην σειριακή οθόνη το μήνυμα (e String) και η τιμή της συχνότητας που έχει υπολογιστεί. Αυτό επιτυγχάνεται με την χρήση της εντολής Serial.println(). Στην συνέχεια, χρησιμοποιούνται οι εντολές lcd.setCursor() και lcd.print(“ “) η οποίες έχουν σκοπό την μετακίνηση του κέρσορα στην πρώτη στήλη της δεύτερης γραμμής της LCD οθόνης, το σβήσιμο της παλιάς τιμής της συχνότητας και την εμφάνιση της πραγματικής τιμής της συχνότητας που υπολογίστηκε. Με αυτόν τον τρόπο, ο χρήστης μπορεί να δει στην LCD οθόνη την πραγματική τιμή της συχνότητας ταλάντωσης της χορδής Μι καντίνι (e). Αφότου ολοκληρωθεί η εμφάνιση της συχνότητας στην LCD οθόνη, εκτελείται η εντολή (k++), η οποία έχει σκοπό την αύξηση της τιμής του μετρητή (k) κατά μία μονάδα.

Έπειτα, εφαρμόζεται η τεχνική ελέγχου (if – else if – else), η οποία αποσκοπεί στον έλεγχο της συχνότητας ταλάντωσης της χορδής. Ουσιαστικά, με την εκτέλεση της εντολής ελέγχου if (frequency < 328.73 && frequency > 277 && frequency > 0), ελέγχεται αν η τιμή της μεταβλητής (frequency) είναι μικρότερη από 328.73Hz και μεγαλύτερη από 277Hz και μεγαλύτερη από 0Hz. Συνεπώς, αν ικανοποιείται αυτή η συνθήκη σημαίνει ότι η συχνότητα ταλάντωσης της χορδής Μι καντίνι (e) είναι χαμηλότερη από την επιθυμητή συχνότητα και για να διορθωθεί θα πρέπει να εκτελεστεί το μπλοκ εντολών της συνθήκης ελέγχου (if). Εντός λοιπόν του μπλοκ εντολών, γίνεται μηδενισμός της μεταβλητής (frequency) και ύστερα εκτελείται ένας βρόγχος επανάληψης (for) ο οποίος επαναλαμβάνει το μπλοκ εντολών του είκοσι ένα φορές. Όσον αφορά το μπλοκ εντολών του βρόγχου επανάληψης (for), γίνεται κλήση της συνάρτησης (Counterclockwise_eStep_Motor ()). Η συνάρτηση αυτή αναλύεται σε επόμενη ενότητα και έχει σκοπό την αντί-ωρολογιακή περιστροφή του άξονα του βηματικού κινητήρα και κατά συνέπεια την αντί-ωρολογιακή περιστροφή του κλειδιού της χορδής Μι καντίνι (e). Μετά την ολοκλήρωση του βρόγχου επανάληψης (for), εκτελείται η εντολή delay(2) η οποία δημιουργεί μια χρονική καθυστέρηση των 2msec. Το μπλοκ εντολών της συνθήκης ελέγχου (if) ολοκληρώνεται με την κλήση της συνάρτησης (eStep_Motor_Stop ()), η οποία αναλύεται σε επόμενο κεφάλαιο και σκοπός της είναι να απενεργοποιήσει τον βηματικό κινητήρα. Αν όμως δεν ικανοποιείται η συνθήκη ελέγχου (if), τότε εκτελείται η εντολή ελέγχου else if (frequency > 331.58 && frequency < 392 && frequency > 0), όπου ελέγχεται αν η τιμή της μεταβλητής (frequency) είναι μεγαλύτερη από 331.58Hz και μικρότερη από 392Hz και μεγαλύτερη από 0Hz. Συνεπώς, αν ικανοποιείται αυτή η συνθήκη σημαίνει ότι η συχνότητα ταλάντωσης της χορδής Μι καντίνι (e) είναι

υψηλότερη από την επιθυμητή συχνότητα και για να διορθωθεί θα πρέπει να εκτελεστεί το μπλοκ εντολών της συνθήκης ελέγχου (else if). Ουσιαστικά, εντός του μπλοκ εντολών γίνεται μηδενισμός της μεταβλητής (frequency) και ύστερα εκτελείται ένας βρόγχος επανάληψης (for). Εντός του μπλοκ εντολών του βρόγχου επανάληψης (for) γίνεται κλήση της συνάρτησης (Clockwise_eStep_Motor ()), της οποίας η ανάλυση πραγματοποιείται σε επόμενη ενότητα. Η κλήση της συνάρτησης (Clockwise_eStep_Motor ()), αποσκοπεί στην ωρολογιακή περιστροφή του άξονα του βηματικού κινητήρα και κατά συνέπεια στην ωρολογιακή περιστροφή του κλειδιού της χορδής Μι καντίνι (e). Μετά την ολοκλήρωση του βρόγχου επανάληψης (for), εκτελείται η εντολή delay(2) η οποία δημιουργεί μια χρονική καθυστέρηση των 2msec. Η τελευταία εντολή που εκτελείται στο μπλοκ εντολών της συνθήκης ελέγχου (else if) αφορά την κλήση της συνάρτησης (eStep_Motor_Stop ()), η οποία αποσκοπεί στην απενεργοποίηση του βηματικού κινητήρα. Σε περίπτωση όμως που δεν ικανοποιείται η προηγούμενη συνθήκη (else if), τότε εκτελείται η εντολή ελέγχου else if (frequency >=328.73 && frequency <=331.58), η οποία ελέγχει αν η τιμή της μεταβλητής (frequency) είναι μεγαλύτερη ή ίση από 328.73Hz και μικρότερη ή ίση από 331.58Hz. Συνεπώς, αν ικανοποιείται αυτή η συνθήκη τότε η πραγματική συχνότητα ταλάντωσης της χορδής Μι καντίνι (e) βρίσκεται εντός του εύρους των επιθυμητών συχνοτήτων της χορδής και εκτελείται το αντίστοιχο μπλοκ εντολών. Πιο συγκεκριμένα, εντός του συγκεκριμένου μπλοκ εντολών, αποθηκεύεται στην σημαία (flag) με όνομα (string_1) η τιμή ένα. Με αυτόν τον τρόπο υποδηλώνεται στο πρόγραμμα η ολοκλήρωση του κουρδίσματος της χορδής Μι καντίνι (e). Έπειτα, εκτελείται η εντολή (k = 21), όπου πραγματοποιείται αρχικοποίηση του μετρητή (k) με την τιμή είκοσι ένα και αμέσως μετά, εκτελείται η εντολή (frequency = 0.00) η οποία μηδενίζει την τιμή της μεταβλητής (frequency). Στην περίπτωση όμως που δεν ικανοποιείται καμία συνθήκη από τις εντολές ελέγχου (if-else if), τότε εκτελείται το μπλοκ εντολών της συνθήκης (else). Ουσιαστικά, το μπλοκ εντολών της συνθήκης (else) αποτελείται από μία μόνο εντολή η οποία μηδενίζει την τιμή της μεταβλητής (frequency). Με αυτό τον τρόπο, αντιμετωπίζονται οι συχνότητες που μπορούν να προκύψουν από την ταυτόχρονη διέγερση δύο χορδών ή από την διέγερση λανθασμένης χορδής. Σε αυτό το σημείο ολοκληρώνεται το μπλοκ εντολών του βρόγχου επανάληψης (while), το οποίο έχει να κάνει με το κούρδισμα της χορδής Μι καντίνι (e). Η διαδικασία που μόλις περιγράφτηκε, επαναλαμβάνεται μέχρι να επιτευχθεί το κούρδισμα της χορδής Μι καντίνι (e) εντός του εύρος των συχνοτήτων ταλάντωσης που έχει οριστεί για την συγκεκριμένη χορδή.

Αφού λοιπόν κουρδιστεί και η τελευταία χορδή της κιθάρας, εκτελούνται οι τελευταίες εντολές του μπλοκ εντολών της συνθήκης ελέγχου (if). Πιο συγκεκριμένα, εκτελούνται οι εντολές lcd.clear(), lcd.setCursor() και lcd.print(“ “) με κατάλληλη σειρά ώστε να εμφανιστεί στην LCD οθόνη το μήνυμα (TUNE FINISHED!). Το μήνυμα αυτό παραμένει στην οθόνη για 5sec εξαιτίας της εντολής καθυστέρησης που εκτελείται delay(5000). Έπειτα, γίνεται κλήση της συνάρτησης Menu (), ώστε να εμφανιστεί στον χρήστη μέσω της LCD οθόνης το μενού επιλογών.

Πίνακας 3.7: Ο κώδικας για το κούρδισμα της κιθάρας σε «πεσμένη» Ρε (Drop D Tune) [33].

Κώδικας Προγραμματισμού (Μέρος 7ο)
else if (menu==3) //Else if menu variable is 3

```
  {
    lcd.clear(); //Clear LCD screen
    lcd.setCursor(2,0); //Set cursor at first column and first line
    lcd.print("Drop D Tune"); //Print message at LCD screen
    lcd.setCursor(4,1); //Set cursor at fifth column and first line
    lcd.print("Selected"); //Print message at LCD screen
    delay(1000); //Delay 1 sec
    lcd.clear(); //Clear LCD screen
    lcd.setCursor(2,0); //Set Cursor at third column and first line
    lcd.print("Hit D string"); //Print message at LCD screen
    delay(1000); //Delay 1 sec
    lcd.clear(); //Clear LCD screen
    lcd.setCursor(1,0); //Set Cursor at third column and first line
    lcd.print("D -> 73.42 Hz"); //Print message at LCD screen
    lcd.setCursor(7,1); //Set Cursor at first column and second line
    lcd.print("Hz   Note"); //print message at LCD screen
    //frequency = 0; //Reset the value at frequency variable
    k = 20;  //Store value at k variable
    string_6 = 0;//Reset the value at string_6
    string_5 = 0;//Reset the value at string_5
    string_4 = 0;//Reset the value at string_4
    string_3 = 0;//Reset the value at string_3
    string_2 = 0;//Reset the value at string_2
    string_1 = 0;//Reset the value at string_1
    while(string_6 != 1) //While string_6 flag isn't set repeat
    {
      if (k>20) //If k variable is bigger than 20, then
      {
        pinMode(39,OUTPUT); //Determine digital pin 39 as Output
        digitalWrite(39,LOW); //Set low voltage (GND) at pin 39
        delay(10); //Delay 10 msec
        pinMode(39,INPUT); //Determine digital pin 39 as Input
        k = 0; //Reset k variable
      } //if (k>20)
      FREQ_DETECTION(); //Call FREQ_DETECTION function (Change analog port for
the income signal)
      delay(500); //Delay 0.5 sec
      FREQ_DETECTION1(); //Call FREQ_DETECTION1 function (Calculate frequency
and stop interrupts by ADC)
```

```
    FREQ_DETECTION_STOP(); //Call FREQ_DETECTION_STOP function (Stop frequency detection)
    Serial.println("D String"); //Print message at Serial Port
    Serial.println(frequency); //Print frequency value at Serial Port
    lcd.setCursor(0,1); //Set cursor at first column and second line
    lcd.print("       "); //Print message at lcd screen
    lcd.setCursor(0,1); //Set cursor at first column and second line
    lcd.print(frequency); //Print frequency value at LCD screen
    k++; //Increase variable k value by one
    if (frequency < 73.37 && frequency > 65 && frequency > 0) //If frequency value is smaller than 73.37Hz and bigger than 65Hz and bigger than 0Hz, then
    {
     frequency = 0.00; //Reset frequency value
     for (i=0; i<=10; i++) //For 11 times repeat
     {
      Counterclockwise_EStep_Motor (); //Call function for E bass string step motor
     }
     delay(2); //Delay 2 msec
     EStep_Motor_Stop (); //Disable E bass string step motor
    } //if (frequency...
    else if (frequency > 73.47 && frequency < 103 && frequency > 0) //Else if frequency value is bigger than 73.47Hz and smaller than 103Hz and bigger than 0Hz, then
    {
     frequency = 0.00; //Reset frequency value
     for(i=0; i<=10; i++) //For 11 times repeat
     {
      Clockwise_EStep_Motor (); //Call function for E bass string step motor
     }
     delay(2); //Delay 2 msec
     EStep_Motor_Stop (); //Disable E bass string step motor
    } //else if (frequency...
    else if (frequency >=73.37 && frequency <=73.47) //Else if frequency value is bigger or equal with 73.37Hz and smaller or equal with 73.47Hz, then D bass string has been tuned
    {
     string_6 = 1; //Set string_6 flag to finish tuning with D bass string
     k = 21;  //Store value at k variable
     frequency = 0.00; //Reset frequency value
     lcd.clear(); //Clear LCD screen
     lcd.setCursor(2,0); //Set Cursor at third column and first line
```

```
        lcd.print("Hit A string"); //Print message at LCD screen
        delay(1000); //Delay 1 sec
        lcd.clear(); //Clear LCD screen
        lcd.setCursor(1,0); //Set Cursor at third column and first line
        lcd.print("a -> 110 Hz"); //Print message at LCD screen
        lcd.setCursor(7,1); //Set Cursor at first column and second line
        lcd.print("Hz   Note"); //print message at LCD screen
       } //else if (frequency...
       else //Else
       {
        frequency = 0.00; //Reset frequency value
       }
     } //while(string_6...
     while(string_5 != 1) //While string_5 flag isn't set repeat
     {
      if (k>20) //If k variable is bigger than 20, then
      {
       pinMode(39,OUTPUT); //Determine digital pin 39 as Output
       digitalWrite(39,LOW); //Set low voltage (GND) at pin 39
       delay(10); //Delay 10 msec
       pinMode(39,INPUT); //Determine digital pin 39 as Input
       k = 0; //Reset k variable
      } //if (k>20)
      FREQ_DETECTION(); //Call FREQ_DETECTION function (Change analog port for the income signal)
      delay(500); //Delay 0.5 sec
      FREQ_DETECTION1(); //Call FREQ_DETECTION1 function (Calculate frequency and stop interrupts by ADC)
      FREQ_DETECTION_STOP(); //Call FREQ_DETECTION_STOP function (Stop frequency detection)
      Serial.println("A String"); //Print message at Serial Port
      Serial.println(frequency); //Print frequency value at Serial Port
      lcd.setCursor(0,1); //Set cursor at first column and second line
      lcd.print("       "); //Print message at lcd screen
      lcd.setCursor(0,1); //Set cursor at first column and second line
      lcd.print(frequency); //Print frequency value at LCD screen
      k++; //Increase variable k value by one
      if (frequency < 109.80 && frequency > 92 && frequency > 0) //If frequency value is smaller than 109.80Hz and bigger than 92Hz and bigger than 0Hz, then
      {
```

```
        frequency = 0.00; //Reset frequency value
        for (i=0; i<=20; i++) //For 21 times repeat
        {
          Counterclockwise_AStep_Motor (); //Call function for A string step motor
        }
        delay(2); //Delay 2 msec
        AStep_Motor_Stop (); //Disable A string step motor
      } //if (frequency...
      else if (frequency > 110.21 && frequency < 130 && frequency > 0) //Else if
frequency value is bigger than 110.21Hz and smaller than 130Hz and bigger than 0Hz,
then
      {
        frequency = 0.00; //Reset frequency value
        for(i=0; i<=20; i++) //For 21 times repeat
        {
          Clockwise_AStep_Motor (); //Call function for A string step motor
        }
        delay(2); //Delay 2 msec
        AStep_Motor_Stop (); //Disable A string step motor
      } //else if (frequency...
      else if (frequency >=109.80 && frequency <=110.21) //Else if frequency value is
bigger or equal with 109.80Hz and smaller or equal with 110.21Hz, then A string has been
tuned
      {
        string_5 = 1; //Set string_5 flag to finish tuning with A string
        k = 21;  //Store value at k variable
        frequency = 0.00; //Reset frequency value
        lcd.clear(); //Clear LCD screen
        lcd.setCursor(2,0); //Set Cursor at third column and first line
        lcd.print("Hit D string"); //Print message at LCD screen
        delay(1000); //Delay 1 sec
        lcd.clear(); //Clear LCD screen
        lcd.setCursor(1,0); //Set Cursor at third column and first line
        lcd.print("D -> 146.83 Hz"); //Print message at LCD screen
        lcd.setCursor(7,1); //Set Cursor at first column and second line
        lcd.print("Hz   Note"); //print message at LCD screen
      } //else if (frequency...
      else  //Else
      {
        frequency = 0.00; //Reset frequency value
```

```
      }
    } //while(string_5...
    while(string_4 != 1) //While string_4 flag isn't set repeat
    {
     if (k>20) //If k variable is bigger than 20, then
     {
      pinMode(39,OUTPUT); //Determine digital pin 39 as Output
      digitalWrite(39,LOW); //Set low voltage (GND) at pin 39
      delay(10); //Delay 10 msec
      pinMode(39,INPUT); //Determine digital pin 39 as Input
      k = 0; //Reset k variable
     } //if (k>20)
     FREQ_DETECTION(); //Call FREQ_DETECTION function (Change analog port for the income signal)
     delay(500); //Delay 0.5 sec
     FREQ_DETECTION1(); //Call FREQ_DETECTION1 function (Calculate frequency and stop interrupts by ADC)
     FREQ_DETECTION_STOP(); //Call FREQ_DETECTION_STOP function (Stop frequency detection)
     Serial.println("D String"); //Print message at Serial Port
     Serial.println(frequency); //Print frequency value at Serial Port
     lcd.setCursor(0,1); //Set cursor at first column and second line
     lcd.print("       "); //Print message at lcd screen
     lcd.setCursor(0,1); //Set cursor at first column and second line
     lcd.print(frequency); //Print frequency value at LCD screen
     k++; //Increase variable k value by one
     if (frequency < 146.73 && frequency > 123 && frequency > 0)  //If frequency value is smaller than 146.73Hz and bigger than 123Hz and bigger than 0Hz, then
     {
      frequency = 0.00; //Reset frequency value
      for (i=0; i<=30; i++) //For 31 times repeat
      {
       Counterclockwise_DStep_Motor (); //Call function for D string step motor
      }
      delay(2); //Delay 2 msec
      DStep_Motor_Stop (); //Disable D string step motor
     } //if (frequency...
     else if (frequency > 146.93 && frequency < 174 && frequency > 0) //Else if frequency value is bigger than 146.93Hz and smaller than 174Hz and bigger than 0Hz, then
```

```
      {
       frequency = 0.00; //Reset frequency value
       for(i=0; i<=30; i++) //For 31 times repeat
       {
        Clockwise_DStep_Motor (); //Call function for D string step motor
       }
       delay(2); //Delay 2 msec
       DStep_Motor_Stop (); //Disable D string step motor
      } //else if (frequency...
      else if (frequency >=146.73 && frequency <=146.93) //Else if frequency value is
bigger or equal with 146.73Hz and smaller or equal with 146.93Hz, then D string has been
tuned
      {
       string_4 = 1; //Set string_4 flag to finish tuning with D string
       k = 21;  //Store value at k variable
       frequency = 0.00; //Reset frequency value
       lcd.clear(); //Clear LCD screen
       lcd.setCursor(2,0); //Set Cursor at third column and first line
       lcd.print("Hit G string"); //Print message at LCD screen
       delay(1000); //Delay 1 sec
       lcd.clear(); //Clear LCD screen
       lcd.setCursor(1,0); //Set Cursor at third column and first line
       lcd.print("G -> 196 Hz"); //Print message at LCD screen
       lcd.setCursor(7,1); //Set Cursor at first column and second line
       lcd.print("Hz   Note"); //print message at LCD screen
      } //else if (frequency...
      else  //Else
      {
       frequency = 0.00; //Reset frequency value
      }
     } //while(string_4...*/
     while(string_3 != 1) //While string_3 flag isn't set repeat
     {
      if (k>20) //If k variable is bigger than 20, then
      {
       pinMode(39,OUTPUT); //Determine digital pin 39 as Output
       digitalWrite(39,LOW); //Set low voltage (GND) at pin 39
       delay(10); //Delay 10 msec
       pinMode(39,INPUT); //Determine digital pin 39 as Input
```

```
      k = 0; //Reset k variable
    } //if (k>20)
    FREQ_DETECTION(); //Call FREQ_DETECTION function (Change analog port for the income signal)
    delay(500); //Delay 0.5 sec
    FREQ_DETECTION1(); //Call FREQ_DETECTION1 function (Calculate frequency and stop interrupts by ADC)
    FREQ_DETECTION_STOP(); //Call FREQ_DETECTION_STOP function (Stop frequency detection)
    Serial.println("G String"); //Print message at Serial Port
    Serial.println(frequency); //Print frequency value at Serial Port
    lcd.setCursor(0,1); //Set cursor at first column and second line
    lcd.print("       "); //Print message at lcd screen
    lcd.setCursor(0,1); //Set cursor at first column and second line
    lcd.print(frequency); //Print frequency value at LCD screen
    k++; //Increase variable k value by one
    if (frequency < 195.90 && frequency > 164 && frequency > 0) //If frequency value is smaller than 195.90Hz and bigger than 164Hz and bigger than 0Hz, then
    {
      frequency = 0.00; //Reset frequency value
      for (i=0; i<=15; i++) //For 16 times repeat
      {
        Clockwise_GStep_Motor (); //Call function for G string step motor
      }
      delay(2); //Delay 2 msec
      GStep_Motor_Stop (); //Disable G string step motor
    } //if (frequency...
    else if (frequency > 196.25 && frequency < 233 && frequency > 0) //Else if frequency value is bigger than 196.25Hz and smaller than 233Hz and bigger than 0Hz, then
    {
      frequency = 0.00; //Reset frequency value
      for(i=0; i<=15; i++) //For 16 times repeat
      {
        Counterclockwise_GStep_Motor (); //Call function for G string step motor
      }
      delay(2); //Delay 2 msec
      GStep_Motor_Stop (); //Disable G string step motor
    } //else if (frequency...
    else if (frequency >=195.90 && frequency <=196.25) //Else if frequency value is
```

```
bigger or equal with 195.90Hz and smaller or equal with 196.25Hz, then G string has been tuned
      {
        string_3 = 1; //Set string_3 flag to finish tuning with G string
        k = 21;  //Store value at k variable
        frequency = 0.00; //Reset frequency value
        lcd.clear(); //Clear LCD screen
        lcd.setCursor(2,0); //Set Cursor at third column and first line
        lcd.print("Hit B string"); //Print message at LCD screen
        delay(1000); //Delay 1 sec
        lcd.clear(); //Clear LCD screen
        lcd.setCursor(1,0); //Set Cursor at third column and first line
        lcd.print("B -> 246.94 Hz"); //Print message at LCD screen
        lcd.setCursor(7,1); //Set Cursor at first column and second line
        lcd.print("Hz   Note"); //print message at LCD screen
      } //else if (frequency...
      else  //Else
      {
        frequency = 0.00; //Reset frequency value
      }
    } //while(string_3...*/
    while(string_2 != 1) //While string_2 flag isn't set repeat
    {
      if (k>20) //If k variable is bigger than 20, then
      {
        pinMode(39,OUTPUT); //Determine digital pin 39 as Output
        digitalWrite(39,LOW); //Set low voltage (GND) at pin 39
        delay(10); //Delay 10 msec
        pinMode(39,INPUT); //Determine digital pin 39 as Input
        k = 0; //Reset k variable
      } //if (k>20)
      FREQ_DETECTION(); //Call FREQ_DETECTION function (Change analog port for the income signal)
      delay(500); //Delay 0.5 sec
      FREQ_DETECTION1(); //Call FREQ_DETECTION1 function (Calculate frequency and stop interrupts by ADC)
      FREQ_DETECTION_STOP(); //Call FREQ_DETECTION_STOP function (Stop frequency detection)
      Serial.println("B String"); //Print message at Serial Port
      Serial.println(frequency); //Print frequency value at Serial Port
```

```
      lcd.setCursor(0,1); //Set cursor at first column and second line
      lcd.print("       "); //Print message at lcd screen
      lcd.setCursor(0,1); //Set cursor at first column and second line
      lcd.print(frequency); //Print frequency value at LCD screen
      k++; //Increase variable k value by one
      if (frequency < 246.54 && frequency > 207 && frequency > 0) //If frequency value
is smaller than 246.54Hz and bigger than 207Hz and bigger than 0Hz, then
      {
       frequency = 0.00; //Reset frequency value
       for (i=0; i<=20; i++) //For 21 times repeat
       {
        Clockwise_BStep_Motor (); //Call function for B string step motor
       }
       delay(2); //Delay 2 msec
       BStep_Motor_Stop (); //Disable B string step motor
      } //if (frequency...
      else if (frequency > 247.04 && frequency < 293 && frequency > 0) //Else if
frequency value is bigger than 247.04Hz and smaller than 293Hz and bigger than 0Hz,
then
      {
       frequency = 0.00; //Reset frequency value
       for(i=0; i<=20; i++) //For 21 times repeat
       {
        Counterclockwise_BStep_Motor (); //Call function for B string step motor
       }
       delay(2); //Delay 2 msec
       BStep_Motor_Stop (); //Disable B string step motor
      } //else if (frequency...
      else if (frequency >=246.55 && frequency <=247.04) //Else if frequency value is
bigger or equal with 246.55Hz and smaller or equal with 247.04Hz, then B string has been
tuned
      {
       string_2 = 1; //Set string_2 flag to finish tuning with B string
       k = 21;  //Store value at k variable
       frequency = 0.00; //Reset frequency value
       lcd.clear(); //Clear LCD screen
       lcd.setCursor(2,0); //Set Cursor at third column and first line
       lcd.print("Hit e string"); //Print message at LCD screen
       delay(1000); //Delay 1 sec
       lcd.clear(); //Clear LCD screen
```

```
        lcd.setCursor(1,0); //Set Cursor at third column and first line
        lcd.print("e -> 329.63 Hz"); //Print message at LCD screen
        lcd.setCursor(7,1); //Set Cursor at first column and second line
        lcd.print("Hz   Note"); //print message at LCD screen
       } //else if (frequency...
       else  //Else
       {
        frequency = 0.00; //Reset frequency value
       }
     } //while(string_2...*/
     while(string_1 != 1) //While string_1 flag isn't set repeat
     {
      if (k>20) //If k variable is bigger than 20, then
      {
       pinMode(39,OUTPUT); //Determine digital pin 39 as Output
       digitalWrite(39,LOW); //Set low voltage (GND) at pin 39
       delay(10); //Delay 10 msec
       pinMode(39,INPUT); //Determine digital pin 39 as Input
       k = 0; //Reset k variable
      } //if (k>20)
      FREQ_DETECTION(); //Call FREQ_DETECTION function (Change analog port for the income signal)
      delay(500); //Delay 0.5 sec
      FREQ_DETECTION1(); //Call FREQ_DETECTION1 function (Calculate frequency and stop interrupts by ADC)
      FREQ_DETECTION_STOP(); //Call FREQ_DETECTION_STOP function (Stop frequency detection)
      Serial.println("e String"); //Print message at Serial Port
      Serial.println(frequency); //Print frequency value at Serial Port
      lcd.setCursor(0,1); //Set cursor at first column and second line
      lcd.print("       "); //Print message at lcd screen
      lcd.setCursor(0,1);  //Set cursor at first column and second line
      lcd.print(frequency); //Print frequency value at LCD screen
      k++; //Increase variable k value by one
      if (frequency < 328.73 && frequency > 277 && frequency > 0) //If frequency value is smaller than 328.73Hz and bigger than 277Hz and bigger than 0Hz, then
      {
       frequency = 0.00; //Reset frequency value
       for (i=0; i<=20; i++) //For 21 times repeat
       {
```

```
          Clockwise_eStep_Motor (); //Call function for e string step motor
         }
         delay(2); //Delay 2 msec
         eStep_Motor_Stop (); //Disable e string step motor
       } //if (frequency...
       else if (frequency > 331.58 && frequency < 392 && frequency > 0) //Else if
frequency value is bigger than 331.58Hz and smaller than 392Hz and bigger than 0Hz,
then
       {
         frequency = 0.00; //Reset frequency value
         for(i=0; i<=20; i++) //For 21 times repeat
         {
          Counterclockwise_eStep_Motor (); //Call function for e string step motor
         }
         delay(2); //Delay 2 msec
         eStep_Motor_Stop (); //Disable e string step motor
       } //else if (frequency...
       else if (frequency >=328.73 && frequency <=331.58) //Else if frequency value is
bigger or equal with 328.73Hz and smaller or equal with 331.58Hz, then e string has been
tuned
       {
         string_1 = 1; //Set string_1 flag to finish tuning with e string
         k = 21;  //Store value at k variable
         frequency = 0.00; //Reset frequency value
       } //else if (frequency...
       else //Else
       {
         frequency = 0.00; //Reset frequency value
       }
     } //while(string_1...*/
     lcd.clear(); //Clear LCD screen
     lcd.setCursor(1,0); //Set cursor at second column and first line
     lcd.print("Tune finished!"); //Print message at LCD screen
     delay(5000); //Delay 5 sec
     Menu(); //Call Menu() function
   } //else if(menu = = 3)
```

Εν συνεχεία, ο κώδικας προγραμματισμού που βρίσκεται εντός του μπλοκ εντολών του πλήκτρου SELECT, όπως προαναφέρθηκε χρησιμοποιεί την τεχνική ελέγχου (if – else if) για να μπορέσει το σύστημα να αναγνωρίσει την επιλογή που έχει κάνει ο χρήστης

μέσα από το μενού επιλογής. Στον προηγούμενο πίνακα (Πίνακας 3.7) παρουσιάζεται ο κώδικας ο οποίος αφορά την συνθήκη ελέγχου else if(menu = = 3), η οποία ελέγχει αν η μεταβλητή (menu) είναι ίση με την τιμή τρία και αντίστοιχα εκτελεί το μπλοκ εντολών για την επίτευξη του κουρδίσματος της κιθάρας σε «πεσμένη» Ρε (Drop D Tune). Αν όμως, δεν ικανοποιείται ούτε αυτή η συνθήκη ελέγχου (else if), τότε αυτό σημαίνει ότι δεν έχει επιλεχτεί το κούρδισμα της κιθάρας σε «πεσμένη» Ρε (Drop D Tune). Πιο συγκεκριμένα, θα μπορούσε να ειπωθεί ότι δεν έχει επιλεχτεί κανένα από τα τρία κουρδίσματα που εμφανίζονται στο μενού επιλογής. Σε αυτήν την περίπτωση, εκτελείται η τελευταία εντολή ελέγχου (else if) που βρίσκεται εντός του μπλοκ εντολών του πλήκτρου SELECT. Πιο συγκεκριμένα, εκτελείται η εντολή ελέγχου else if (menu = = 4), η οποία ελέγχει την τιμή της μεταβλητής (menu) αν είναι ίση με τέσσερα. Επομένως, στην περίπτωση που ικανοποιείται η συγκεκριμένη συνθήκη ελέγχου (else if) τότε εκτελείται το αντίστοιχο μπλοκ εντολών, γεγονός που σημαίνει ότι έχει επιλεχτεί ο χειροκίνητος έλεγχος του συστήματος (Πίνακας 3.8).

Εντός του μπλοκ εντολών της συνθήκης ελέγχου (else if) η οποία αφορά τον χειροκίνητο έλεγχο του συστήματος, γίνονται πρώτα κάποιες αρχικοποιήσεις. Πιο συγκεκριμένα, η σημαία (flag) με όνομα (Manual_Finish) λαμβάνει την τιμή μηδέν και σκοπός της είναι να προσδιορίσει στο σύστημα πότε έχει ολοκληρωθεί ο χειροκίνητος έλεγχος του συστήματος. Έπειτα, η μεταβλητή (manual_menu), λαμβάνει την τιμή ένα και αποσκοπεί στον έλεγχο του χειροκίνητου μενού επιλογής. Ύστερα, γίνεται εκτελέσει των εντολών lcd.clear(), lcd.setCursor() και lcd.print(“ “) με την κατάλληλη σειρά ώστε να εμφανιστεί το μήνυμα (Choose string!) στην LCD οθόνη. Σκοπός του μηνύματος αυτού είναι να προτρέψει τον χρήστη να επιλέξει την χορδή που επιθυμεί να ελέγξει μέσω του χειροκίνητου ελέγχου του συστήματος. Το μήνυμα αυτό παραμένει στην LCD οθόνη για 2.5sec εξαιτίας της εντολής καθυστέρησης delay(2500) που εκτελείται αμέσως μετά. Ύστερα, εκτελείται η εντολή lcd.clear() ώστε να σβηστούν οι χαρακτήρες από την LCD οθόνη.

Εν συνεχεία, εκτελείται η εντολή επανάληψης while (Manual_ Finish == 0) η οποία ελέγχει αν η τιμή της σημαίας (flag) με όνομα (Manual_Finish) είναι ίση με το μηδέν. Πιο συγκεκριμένα, το μπλοκ εντολών του βρόγχου επανάληψης (while) επαναλαμβάνεται όσο συνεχίζει να ικανοποιείται η συγκεκριμένη συνθήκη. Ουσιαστικά, όσο εκτελείται το μπλοκ εντολών του συγκεκριμένου βρόγχου επανάληψης αυτό σημαίνει ότι η εκτέλεση του προγράμματος βρίσκεται εντός του μενού επιλογών του χειροκίνητου ελέγχου του συστήματος.

Εντός λοιπόν του μπλοκ εντολών του βρόγχου επανάληψης (while), γίνεται πρώτα κλήση της συνάρτησης Manual_Button_Read() της οποίας η ανάλυση γίνεται σε επόμενη ενότητα. Ουσιαστικά η συγκεκριμένη συνάρτηση, αποσκοπεί στην αναγνώριση των πλήκτρων (UP, DOWN, LEFT, RIGHT & SELECT) που βρίσκονται ενσωματωμένα στην πλακέτα της LCD οθόνης. Ύστερα, εφαρμόζεται η τεχνική (if – else if) η οποία έχει κύριο στόχο την εμφάνιση της επιλογής που έχει πραγματοποιήσει ο χρήστης μέσω του χειροκίνητου μενού. Πιο συγκεκριμένα, εκτελείται η εντολή ελέγχου if(manual_menu = = 1), η οποία ελέγχει αν η τιμή της μεταβλητής (manual_menu) είναι ίση με ένα. Αν ικανοποιείται η συγκεκριμένη συνθήκη, τότε εκτελείται το μπλοκ εντολών της συνθήκης ελέγχου (if). Αυτό σημαίνει ότι ο χρήστης έχει επιλέξει στο μενού τον χειροκίνητο έλεγχο της χορδής Μι μπάσο (Ε), χωρίς όμως να έχει οριστικοποιήσει την επιλογή του! Πιο συγκεκριμένα, εντός του μπλοκ εντολών της συνθήκης ελέγχου (if), εκτελούνται με

κατάλληλή σειρά οι εντολές lcd.setcursor() και lcd.print(“ “) η οποίες εμφανίζουν στις κατάλληλες στήλες και στις δύο γραμμές το χειροκίνητο μενού (>E A D G B e Back) με την επιλογή να βρίσκεται στην χορδή Μι μπάσο (E). Έπειτα, εντός του μπλοκ εντολών της συνθήκης (if), εκτελείται η εντολή ελέγχου if(Manual_Select = = 1) η οποία αποτελεί ορολογιακά μία φωλιασμένη συνθήκη ελέγχου (if) και ελέγχει αν η σημαία (flag) με όνομα (Manual_Select) είναι ίση με την τιμή ένα. Ουσιαστικά η φωλιασμένη συνθήκη ελέγχου (if) ελέγχει αν ο χρήστης έχει οριστικοποιήσει την επιλογή για τον χειροκίνητο έλεγχο της χορδής Μι μπάσο (E). Αν λοιπόν ικανοποιείται η συγκεκριμένη συνθήκη ελέγχου της φωλιασμένης (if), τότε ο χρήστης έχει οριστικοποιήσει την επιλογή του και εκτελείται το μπλοκ εντολών της φωλιασμένης συνθήκης ελέγχου (if). Εντός λοιπόν του μπλοκ εντολών της φωλιασμένης συνθήκης ελέγχου (if), γίνονται πρώτα κάποιες αρχικοποιήσεις οι οποίες αφορούν τον μηδενισμό της σημαίας (flag) με όνομα (Manual_Select) καθώς και της σημαίας (flag) με όνομα (Finish_Control). Ύστερα, εκτελούνται με κατάλληλη σειρά οι εντολές lcd.clear(), lcd.setCursor() και lcd.print(“ “), ώστε να εμφανιστεί το μήνυμα (Press Right or Left Buttons) στην LCD οθόνη. Το μήνυμα αυτό παραμένει στην LCD οθόνη για 3sec εξαιτίας της εντολής delay(3000) που εκτελείται αμέσως μετά. Σκοπός του μηνύματος αυτού, είναι να πληροφορήσει τον χρήστη ότι για να ελέγξει χειροκίνητα τον βηματικό κινητήρα της χορδής Μι μπάσο (E), θα πρέπει να χρησιμοποιήσει τα πλήκτρα LEFT και RIGHT που βρίσκονται ενσωματωμένα στο πληκτρολόγιο της πλακέτας της LCD οθόνης. Στην συνέχεια, εκτελούνται ξανά οι εντολές lcd.clear(), lcd.setCursor() και lcd.print(“ “) με την κατάλληλη σειρά, ώστε να εμφανιστεί το μήνυμα (Press SELECT to go back) στην LCD οθόνη. Σκοπός του μηνύματος αυτού είναι να πληροφορήσει τον χρήστη ότι πιέζοντας το πλήκτρο SELECT, θα επιστρέψει στο χειροκίνητο μενού και ότι η διαδικασία του χειροκίνητου ελέγχου της χορδής Μι μπάσο (E) θα σταματήσει. Ύστερα, εκτελείται η εντολή επανάληψης while (Finish_Control = = 0), η οποία ελέγχει αν η τιμή της σημαίας (flag) με όνομα (Finish_Control) είναι ίση με μηδέν. Ουσιαστικά, το μπλοκ εντολών του βρόγχου επανάληψης (while) επαναλαμβάνεται όσο η συνθήκη συνεχίζει να ικανοποιείται. Η σημαία (flag) με όνομα (Finish_Control) έχει αρχικοποιηθεί πριν τον βρόγχο επανάληψης (while) και έχει λάβει την τιμή μηδέν. Συνεπώς, η εκτέλεση του προγράμματος θα συνεχίσει εντός του μπλοκ εντολών του βρόγχου επανάληψης (while) το οποίο είναι υπεύθυνο για τον χειροκίνητο έλεγχο της χορδής Μι μπάσο (E). Πιο συγκεκριμένα, εντός του μπλοκ εντολών του βρόγχου επανάληψης (while) γίνεται κλήση της συνάρτησης Manual_Step_Motor_Control(), η οποία είναι υπεύθυνη για την ανάγνωση των πλήκτρων LEFT, RIGHT, UP, DOWN και SELECT και αναλύεται σε επόμενη ενότητα. Έπειτα, πραγματοποιείται μια καθυστέρηση των 50msec εξαιτίας της εντολής καθυστέρησης delay(50) που εκτελείται αμέσως μετά. Εν συνεχεία, εφαρμόζεται ξανά η τεχνική ελέγχου (if – else if) ώστε να περιστραφεί ο βηματικός κινητήρας της χορδής Μι μπάσο (E) είτε ωρολογιακά, είτε αντί-ωρολογιακά. Πιο συγκεκριμένα, εκτελείται η εντολή ελέγχου if (Turn_Right = = 1) η οποία ελέγχει αν η τιμή της σημαίας (flag) με όνομα (Turn_Right) είναι ίση με ένα. Αν λοιπόν έχει πατηθεί το πλήκτρο RIGHT, τότε η συνθήκη ελέγχου (if) ικανοποιείται και εκτελείται το αντίστοιχο μπλοκ εντολών. Εντός του μπλοκ εντολών της συνθήκης ελέγχου (if), εκτελείται η συνθήκη επανάληψης (for), η οποία εκτελεί το μπλοκ εντολών της είκοσι ένα φορές. Στο εσωτερικό του μπλοκ εντολών του βρόγχου επανάληψης (for) γίνεται κλήση της συνάρτησης Clockwise_EStep_Motor (), η οποία έχει σκοπό την ωρολογιακή περιστροφή του άξονα του βηματικού κινητήρα και κατά συνέπεια την ωρολογιακή περιστροφή του κλειδιού της

χορδής Μι μπάσο (Ε). Αφού ολοκληρωθεί η εκτέλεση του μπλοκ εντολών του βρόγχου επανάληψης (for), τότε γίνεται μηδενισμός της σημαίας (flag) με όνομα (Turn_Right) και έτσι ολοκληρώνεται το μπλοκ εντολών της εντολής ελέγχου if (Turn_Right = = 1). Στην περίπτωση όμως που δεν ικανοποιείτε η συνθήκη ελέγχου (if), τότε εκτελείται η εντολή ελέγχου else if (Turn_Left = = 1), η οποία ελέγχει αν η τιμή της σημαίας (flag) με όνομα (Turn_Left) είναι ίση με ένα. Επομένως, αν πατηθεί το πλήκτρο LEFT, τότε η συνθήκη ελέγχου (else if) ικανοποιείται και εκτελείται το αντίστοιχο μπλοκ εντολών. Πιο συγκεκριμένα, εντός του μπλοκ εντολών της συνθήκης ελέγχου (else if), εκτελείται η συνθήκη επανάληψης (for), η οποία εκτελεί το μπλοκ εντολών της είκοσι ένα φορές. Στο εσωτερικό του μπλοκ εντολών του βρόγχου επανάληψης (for) γίνεται κλήση της συνάρτησης Counterclockwise_EStep_Motor (), η οποία έχει σκοπό την αντί-ωρολογιακή περιστροφή του άξονα του βηματικού κινητήρα και κατά συνέπεια την αντί-ωρολογιακή περιστροφή του κλειδιού της χορδής Μι μπάσο (Ε). Αφού ολοκληρωθεί η εκτέλεση του μπλοκ εντολών του βρόγχου επανάληψης (for), τότε γίνεται μηδενισμός της σημαίας (flag) με όνομα (Turn_Left) και έτσι ολοκληρώνεται το μπλοκ εντολών της εντολής ελέγχου else if (Turn_Left = = 1). Πάραυτα, αν δε πατηθεί κανένα κουμπί η εκτέλεση του προγράμματος παραμένει εντός του βρόγχου (while). Στην περίπτωση όμως που πατηθεί το πλήκτρο SELECT, όπως θα αναλυθεί σε επόμενη ενότητα, η σημαία (flag) με όνομα (Finish_Control) λαμβάνει την τιμή ένα και έτσι η εκτέλεση του προγράμματος επιστρέφει στο χειροκίνητο μενού. Με αυτόν τον τρόπο, ολοκληρώνεται ο χειροκίνητος έλεγχος του κλειδιού της χορδής Μι μπάσο (Ε).

Στην περίπτωση όμως που δεν ικανοποιείται η συνθήκη if(manual_menu = = 1), τότε εκτελείται η εντολή ελέγχου else if (manual_menu = = 2), η οποία ελέγχει αν η τιμή της μεταβλητής (manual_menu) είναι ίση με δύο. Επομένως, αν ικανοποιείται η συγκεκριμένη συνθήκη, τότε εκτελείται το μπλοκ εντολών της συνθήκης ελέγχου (else if). Αυτό σημαίνει ότι ο χρήστης έχει επιλέξει στο μενού τον χειροκίνητο έλεγχο της χορδής Λα (Α), χωρίς όμως να έχει οριστικοποιήσει ακόμα την επιλογή του! Ουσιαστικά, εντός του μπλοκ εντολών της συνθήκης ελέγχου (else if), εκτελούνται με κατάλληλή σειρά οι εντολές lcd.setcursor() και lcd.print(“ “) η οποίες εμφανίζουν στις κατάλληλες στήλες και στις δύο γραμμές το χειροκίνητο μενού (E >A D G B e Back) με την επιλογή να βρίσκεται στην χορδή Λα (Α). Εν συνεχεία, εντός του μπλοκ εντολών της συνθήκης (else if), εκτελείται η εντολή ελέγχου if(Manual_Select = = 1), η οποία αποτελεί ορολογιακά μία φωλιασμένη συνθήκη ελέγχου (if) και ελέγχει αν η σημαία (flag) με όνομα (Manual_Select) είναι ίση με την τιμή ένα. Πιο αναλυτικά, η φωλιασμένη συνθήκη ελέγχου (if) εντοπίζει αν ο χρήστης έχει οριστικοποιήσει την επιλογή για τον χειροκίνητο έλεγχο της χορδής Λα (Α). Αν λοιπόν ικανοποιείται η συγκεκριμένη συνθήκη ελέγχου της φωλιασμένης (if), τότε ο χρήστης έχει οριστικοποιήσει την επιλογή του και εκτελείται το μπλοκ εντολών της φωλιασμένης συνθήκης ελέγχου (if). Πιο συγκεκριμένα, εντός του μπλοκ εντολών της φωλιασμένης συνθήκης ελέγχου (if), γίνονται πρώτα κάποιες αρχικοποιήσεις οι οποίες αφορούν τον μηδενισμό της σημαίας (flag) με όνομα (Manual_Select) καθώς και της σημαίας (flag) με όνομα (Finish_Control). Έπειτα, εκτελούνται οι εντολές lcd.clear(), lcd.setCursor() και lcd.print(“ “) με κατάλληλη σειρά, ώστε να εμφανιστεί το μήνυμα (Press Right or Left Buttons) στην LCD οθόνη. Το μήνυμα αυτό παραμένει στην LCD οθόνη για 3sec εξαιτίας της εντολής delay(3000) που εκτελείται αμέσως μετά. Σκοπός του μηνύματος αυτού, είναι να πληροφορήσει τον χρήστη ότι για να ελέγξει χειροκίνητα τον βηματικό κινητήρα της χορδής Λα (Α), θα πρέπει να χρησιμοποιήσει τα πλήκτρα LEFT και RIGHT που βρίσκονται ενσωματωμένα

στο πληκτρολόγιο της πλακέτας της LCD οθόνης. Ύστερα, εκτελούνται ξανά οι εντολές lcd.clear(), lcd.setCursor() και lcd.print(“ “) με την κατάλληλη σειρά, ώστε να εμφανιστεί το μήνυμα (Press SELECT to go back) στην LCD οθόνη. Το μήνυμα αυτό αποσκοπεί στην πληροφόρηση του χρήστη ότι πιέζοντας το πλήκτρο SELECT, θα επιστρέψει στο χειροκίνητο μενού και θα ολοκληρωθεί η διαδικασία του χειροκίνητου ελέγχου της χορδής Λα (A). Εν συνεχεία, εκτελείται η εντολή επανάληψης while (Finish_Control = = 0), η οποία ελέγχει αν η τιμή της σημαίας (flag) με όνομα (Finish_Control) είναι ίση με μηδέν. Πιο συγκεκριμένα, το μπλοκ εντολών του βρόγχου επανάληψης (while) επαναλαμβάνεται όσο η συνθήκη συνεχίζει να ικανοποιείται. Επιπλέον, η σημαία (flag) με όνομα (Finish_Control) έχει αρχικοποιηθεί πριν τον βρόγχο επανάληψης (while) και έχει λάβει την τιμή μηδέν. Επομένως, η εκτέλεση του προγράμματος θα συνεχίσει εντός του μπλοκ εντολών του βρόγχου επανάληψης (while) το οποίο είναι υπεύθυνο για τον χειροκίνητο έλεγχο της χορδής Λα (A). Πιο αναλυτικά, εντός του μπλοκ εντολών του βρόγχου επανάληψης (while) γίνεται κλήση της συνάρτησης Manual_Step_Motor_Control(), η οποία αναλύεται σε επόμενη ενότητα και είναι υπεύθυνη για την ανάγνωση των πλήκτρων LEFT, RIGHT, UP, DOWN και SELECT. Ύστερα, πραγματοποιείται μια καθυστέρηση των 50msec εξαιτίας της εντολής καθυστέρησης delay(50) που εκτελείται αμέσως μετά. Έπειτα, εφαρμόζεται ξανά η τεχνική ελέγχου (if – else if) ώστε να επιτευχθεί η περιστροφή του βηματικού κινητήρα της χορδής Λα (A) είτε ωρολογιακά, είτε αντί-ωρολογιακά. Ουσιαστικά, εκτελείται η εντολή ελέγχου if (Turn_Right = = 1), η οποία ελέγχει αν η τιμή της σημαίας (flag) με όνομα (Turn_Right) είναι ίση με ένα. Συνεπώς, αν έχει πατηθεί το πλήκτρο RIGHT, τότε η συνθήκη ελέγχου (if) ικανοποιείται και εκτελείται το αντίστοιχο μπλοκ εντολών. Πιο συγκεκριμένα, εντός του μπλοκ εντολών της συνθήκης ελέγχου (if), εκτελείται η συνθήκη επανάληψης (for), η οποία εκτελεί το μπλοκ εντολών της είκοσι ένα φορές. Όσον αφορά το μπλοκ εντολών του βρόγχου επανάληψης (for) γίνεται κλήση της συνάρτησης Clockwise_AStep_Motor (), η οποία αποσκοπεί στην ωρολογιακή περιστροφή του άξονα του βηματικού κινητήρα και κατά συνέπεια στην ωρολογιακή περιστροφή του κλειδιού της χορδής Λα (A). Αφού ολοκληρωθεί η εκτέλεση του μπλοκ εντολών του βρόγχου επανάληψης (for), τότε γίνεται μηδενισμός της σημαίας (flag) με όνομα (Turn_Right) και έτσι ολοκληρώνεται το μπλοκ εντολών της εντολής ελέγχου if (Turn_Right = = 1). Αν όμως δεν ικανοποιείτε η συνθήκη ελέγχου (if), τότε εκτελείται η εντολή ελέγχου else if (Turn_Left = = 1), η οποία ελέγχει αν η τιμή της σημαίας (flag) με όνομα (Turn_Left) είναι ίση με ένα. Συνεπώς, στην περίπτωση που πατηθεί το πλήκτρο LEFT, τότε η συνθήκη ελέγχου (else if) ικανοποιείται και εκτελείται το αντίστοιχο μπλοκ εντολών. Πιο αναλυτικά, το μπλοκ εντολών της συνθήκης ελέγχου (else if), περιλαμβάνει μία συνθήκη επανάληψης (for), η οποία εκτελεί το μπλοκ εντολών της είκοσι ένα φορές. Στο εσωτερικό του μπλοκ εντολών του βρόγχου επανάληψης (for) γίνεται κλήση της συνάρτησης Counterclockwise_AStep_Motor (), η οποία αποσκοπεί στην αντί-ωρολογιακή περιστροφή του άξονα του βηματικού κινητήρα και κατά συνέπεια στην αντί-ωρολογιακή περιστροφή του κλειδιού της χορδής Λα (A). Αφότου ολοκληρωθεί η εκτέλεση του μπλοκ εντολών του βρόγχου επανάληψης (for), τότε γίνεται μηδενισμός της σημαίας (flag) με όνομα (Turn_Left) και έτσι ολοκληρώνεται το μπλοκ εντολών της εντολής ελέγχου else if (Turn_Left = = 1). Στην περίπτωση όμως που δεν πατηθεί κανένα κουμπί, η εκτέλεση του προγράμματος παραμένει εντός του βρόγχου (while). Αν όμως, πατηθεί το πλήκτρο SELECT, όπως θα αναλυθεί σε επόμενη ενότητα, η σημαία (flag) με όνομα (Finish_Control) λαμβάνει την τιμή ένα και έτσι η εκτέλεση του προγράμματος

επιστρέφει στο χειροκίνητο μενού. Με αυτόν τον τρόπο, ολοκληρώνεται ο χειροκίνητος έλεγχος του κλειδιού της χορδής Λα (A).

Από την άλλη πλευρά, αν δεν ικανοποιηθεί η συνθήκη else if (manual_menu = = 2), τότε εκτελείται η εντολή ελέγχου else if (manual_menu = = 3), η οποία ελέγχει αν η τιμή της μεταβλητής (manual_menu) είναι ίση με τρία. Συνεπώς, αν ικανοποιείται η συγκεκριμένη συνθήκη, τότε εκτελείται το μπλοκ εντολών της συνθήκης ελέγχου (else if). Με αυτόν τον τρόπο, το σύστημα αντιλαμβάνεται ότι ο χρήστης έχει επιλέξει στο μενού τον χειροκίνητο έλεγχο της χορδής Ρε (D), χωρίς όμως να έχει οριστικοποιήσει την επιλογή του! Πιο συγκεκριμένα, εντός του μπλοκ εντολών της συνθήκης ελέγχου (if), εκτελούνται με κατάλληλή σειρά οι εντολές lcd.setcursor() και lcd.print(“ “) η οποίες εμφανίζουν στις κατάλληλες στήλες και στις δύο γραμμές το χειροκίνητο μενού (E A >D G B e Back) με την επιλογή να βρίσκεται στην χορδή Ρε (D). Έπειτα, εντός του μπλοκ εντολών της συνθήκης (else if), εκτελείται η εντολή ελέγχου if(Manual_Select = = 1) η οποία αποτελεί ορολογιακά μία φωλιασμένη συνθήκη ελέγχου (if) και ελέγχει αν η σημαία (flag) με όνομα (Manual_Select) είναι ίση με την τιμή ένα. Ουσιαστικά, η φωλιασμένη συνθήκη ελέγχου (if) ελέγχει αν ο χρήστης έχει οριστικοποιήσει την επιλογή για τον χειροκίνητο έλεγχο της χορδής Ρε (D). Αν λοιπόν ικανοποιείται η συγκεκριμένη συνθήκη ελέγχου της φωλιασμένης (if), τότε ο χρήστης έχει οριστικοποιήσει την επιλογή του και εκτελείται το μπλοκ εντολών της φωλιασμένης συνθήκης ελέγχου (if). Εντός λοιπόν του μπλοκ εντολών της φωλιασμένης συνθήκης ελέγχου (if), γίνονται πρώτα κάποιες αρχικοποιήσεις οι οποίες αφορούν τον μηδενισμό της σημαίας (flag) με όνομα (Manual_Select) καθώς και της σημαίας (flag) με όνομα (Finish_Control). Ύστερα, εκτελούνται με κατάλληλη σειρά οι εντολές lcd.clear(), lcd.setCursor() και lcd.print(“ “), ώστε να εμφανιστεί το μήνυμα (Press Right or Left Buttons) στην LCD οθόνη. Το μήνυμα αυτό παραμένει στην LCD οθόνη για 3sec εξαιτίας της εντολής delay(3000) που εκτελείται αμέσως μετά. Σκοπός του μηνύματος αυτού, είναι να πληροφορήσει τον χρήστη ότι για να ελέγξει χειροκίνητα τον βηματικό κινητήρα της χορδής Ρε (D), θα πρέπει να χρησιμοποιήσει τα πλήκτρα LEFT και RIGHT που βρίσκονται ενσωματωμένα στο πληκτρολόγιο της πλακέτας της LCD οθόνης. Στην συνέχεια, εκτελούνται ξανά οι εντολές lcd.clear(), lcd.setCursor() και lcd.print(“ “) με την κατάλληλη σειρά, ώστε να εμφανιστεί το μήνυμα (Press SELECT to go back) στην LCD οθόνη. Ουσιαστικά, σκοπός του μηνύματος αυτού είναι να πληροφορήσει τον χρήστη ότι πιέζοντας το πλήκτρο SELECT, θα επιστρέψει στο χειροκίνητο μενού και ότι η διαδικασία του χειροκίνητου ελέγχου της χορδής Ρε (D) θα σταματήσει. Ύστερα, εκτελείται η εντολή επανάληψης while (Finish_Control = = 0), η οποία ελέγχει αν η τιμή της σημαίας (flag) με όνομα (Finish_Control) είναι ίση με μηδέν. Ουσιαστικά, το μπλοκ εντολών του βρόγχου επανάληψης (while) επαναλαμβάνεται όσο η συνθήκη συνεχίζει να ικανοποιείται. Η σημαία (flag) με όνομα (Finish_Control) έχει αρχικοποιηθεί πριν τον βρόγχο επανάληψης (while) και έχει λάβει την τιμή μηδέν. Συνεπώς, η εκτέλεση του προγράμματος θα συνεχίσει εντός του μπλοκ εντολών του βρόγχου επανάληψης (while) το οποίο είναι υπεύθυνο για τον χειροκίνητο έλεγχο της χορδής Ρε (D). Πιο συγκεκριμένα, εντός του μπλοκ εντολών του βρόγχου επανάληψης (while) γίνεται κλήση της συνάρτησης Manual_Step_Motor_Control(), η οποία είναι υπεύθυνη για την ανάγνωση των πλήκτρων LEFT, RIGHT, UP, DOWN και SELECT και αναλύεται σε επόμενη ενότητα. Έπειτα, πραγματοποιείται μια καθυστέρηση των 50msec εξαιτίας της εντολής καθυστέρησης delay(50) που εκτελείται αμέσως μετά. Εν συνεχεία, εφαρμόζεται ξανά η τεχνική ελέγχου (if – else if) ώστε να περιστραφεί ο βηματικός κινητήρας της χορδής Ρε (D) είτε

ωρολογιακά, είτε αντί-ωρολογιακά. Πιο συγκεκριμένα, εκτελείται η εντολή ελέγχου if (Turn_Right = = 1), η οποία ελέγχει αν η τιμή της σημαίας (flag) με όνομα (Turn_Right) είναι ίση με ένα. Αν λοιπόν έχει πατηθεί το πλήκτρο RIGHT, τότε η συνθήκη ελέγχου (if) ικανοποιείται και εκτελείται το αντίστοιχο μπλοκ εντολών. Εντός του μπλοκ εντολών της συνθήκης ελέγχου (if), εκτελείται η συνθήκη επανάληψης (for), η οποία εκτελεί το μπλοκ εντολών της είκοσι ένα φορές. Στο εσωτερικό του μπλοκ εντολών του βρόγχου επανάληψης (for) γίνεται κλήση της συνάρτησης Clockwise_DStep_Motor (), η οποία έχει σκοπό την ωρολογιακή περιστροφή του άξονα του βηματικού κινητήρα και κατά συνέπεια την ωρολογιακή περιστροφή του κλειδιού της χορδής Ρε (D). Αφού ολοκληρωθεί η εκτέλεση του μπλοκ εντολών του βρόγχου επανάληψης (for), τότε γίνεται μηδενισμός της σημαίας (flag) με όνομα (Turn_Right) και έτσι ολοκληρώνεται το μπλοκ εντολών της εντολής ελέγχου if (Turn_Right = = 1). Στην περίπτωση όμως που δεν ικανοποιείτε η συνθήκη ελέγχου (if), τότε εκτελείται η εντολή ελέγχου else if (Turn_Left = = 1), η οποία ελέγχει αν η τιμή της σημαίας (flag) με όνομα (Turn_Left) είναι ίση με ένα. Επομένως, αν πατηθεί το πλήκτρο LEFT, τότε η συνθήκη ελέγχου (else if) ικανοποιείται και εκτελείται το αντίστοιχο μπλοκ εντολών. Πιο συγκεκριμένα, εντός του μπλοκ εντολών της συνθήκης ελέγχου (else if), εκτελείται η συνθήκη επανάληψης (for), η οποία εκτελεί το μπλοκ εντολών της είκοσι ένα φορές. Στο εσωτερικό του μπλοκ εντολών του βρόγχου επανάληψης (for) γίνεται κλήση της συνάρτησης Counterclockwise_DStep_Motor (), η οποία έχει σκοπό την αντί-ωρολογιακή περιστροφή του άξονα του βηματικού κινητήρα και κατά συνέπεια την αντί-ωρολογιακή περιστροφή του κλειδιού της χορδής Ρε (D). Αφού ολοκληρωθεί η εκτέλεση του μπλοκ εντολών του βρόγχου επανάληψης (for), τότε γίνεται μηδενισμός της σημαίας (flag) με όνομα (Turn_Left) και έτσι ολοκληρώνεται το μπλοκ εντολών της εντολής ελέγχου else if (Turn_Left = = 1). Πάραυτα, αν δε πατηθεί κανένα κουμπί η εκτέλεση του προγράμματος παραμένει εντός του βρόγχου (while). Στην περίπτωση όμως που πατηθεί το πλήκτρο SELECT, όπως θα αναλυθεί σε επόμενη ενότητα, η σημαία (flag) με όνομα (Finish_Control) λαμβάνει την τιμή ένα και έτσι η εκτέλεση του προγράμματος επιστρέφει στο χειροκίνητο μενού. Με αυτόν τον τρόπο, ολοκληρώνεται ο χειροκίνητος έλεγχος του κλειδιού της χορδής Ρε (D).

Στην περίπτωση όμως που δεν ικανοποιείται η συνθήκη else if(manual_menu = = 3), τότε εκτελείται η εντολή ελέγχου else if (manual_menu = = 4), η οποία ελέγχει αν η τιμή της μεταβλητής (manual_menu) είναι ίση με τέσσερα. Επομένως, αν ικανοποιείται η συγκεκριμένη συνθήκη, τότε εκτελείται το μπλοκ εντολών της συνθήκης ελέγχου (else if). Αυτό σημαίνει ότι ο χρήστης έχει επιλέξει στο μενού τον χειροκίνητο έλεγχο της χορδής Σολ (G), χωρίς όμως να έχει οριστικοποιήσει ακόμα την επιλογή του! Ουσιαστικά, εντός του μπλοκ εντολών της συνθήκης ελέγχου (else if), εκτελούνται με κατάλληλή σειρά οι εντολές lcd.setcursor() και lcd.print(“ “) η οποίες εμφανίζουν στις κατάλληλες στήλες και στις δύο γραμμές το χειροκίνητο μενού (E A D >G B e Back) με την επιλογή να βρίσκεται στην χορδή Σολ (G). Εν συνεχεία, εντός του μπλοκ εντολών της συνθήκης (else if), εκτελείται η εντολή ελέγχου if (Manual_Select = = 1), η οποία αποτελεί ορολογιακά μία φωλιασμένη συνθήκη ελέγχου (if) και ελέγχει αν η σημαία (flag) με όνομα (Manual_Select) είναι ίση με την τιμή ένα. Πιο αναλυτικά, η φωλιασμένη συνθήκη ελέγχου (if) εντοπίζει αν ο χρήστης έχει οριστικοποιήσει την επιλογή για τον χειροκίνητο έλεγχο της χορδής Σολ (G). Αν λοιπόν ικανοποιείται η συγκεκριμένη συνθήκη ελέγχου της φωλιασμένης (if), τότε ο χρήστης έχει οριστικοποιήσει την επιλογή του και εκτελείται το μπλοκ εντολών της φωλιασμένης συνθήκης ελέγχου (if). Πιο συγκεκριμένα, εντός του μπλοκ εντολών της φωλιασμένης συνθήκης ελέγχου (if), γίνονται πρώτα κάποιες

αρχικοποιήσεις οι οποίες αφορούν τον μηδενισμό της σημαίας (flag) με όνομα (Manual_Select) καθώς και της σημαίας (flag) με όνομα (Finish_Control). Έπειτα, εκτελούνται οι εντολές lcd.clear(), lcd.setCursor() και lcd.print(“ “) με κατάλληλη σειρά, ώστε να εμφανιστεί το μήνυμα (Press Right or Left Buttons) στην LCD οθόνη. Το μήνυμα αυτό παραμένει στην LCD οθόνη για 3sec εξαιτίας της εντολής delay(3000) που εκτελείται αμέσως μετά. Σκοπός του μηνύματος αυτού, είναι να πληροφορήσει τον χρήστη ότι για να ελέγξει χειροκίνητα τον βηματικό κινητήρα της χορδής Σολ (G), θα πρέπει να χρησιμοποιήσει τα πλήκτρα LEFT και RIGHT που βρίσκονται ενσωματωμένα στο πληκτρολόγιο της πλακέτας της LCD οθόνης. Ύστερα, εκτελούνται ξανά οι εντολές lcd.clear(), lcd.setCursor() και lcd.print(“ “) με την κατάλληλη σειρά, ώστε να εμφανιστεί το μήνυμα (Press SELECT to go back) στην LCD οθόνη. Το μήνυμα αυτό αποσκοπεί στην πληροφόρηση του χρήστη ότι πιέζοντας το πλήκτρο SELECT, θα επιστρέψει στο χειροκίνητο μενού και θα ολοκληρωθεί η διαδικασία του χειροκίνητου ελέγχου της χορδής Σολ (G). Εν συνεχεία, εκτελείται η εντολή επανάληψης while (Finish_Control = = 0), η οποία ελέγχει αν η τιμή της σημαίας (flag) με όνομα (Finish_Control) είναι ίση με μηδέν. Πιο συγκεκριμένα, το μπλοκ εντολών του βρόγχου επανάληψης (while) επαναλαμβάνεται όσο η συνθήκη συνεχίζει να ικανοποιείται. Επιπλέον, η σημαία (flag) με όνομα (Finish_Control) έχει αρχικοποιηθεί πριν τον βρόγχο επανάληψης (while) και έχει λάβει την τιμή μηδέν. Επομένως, η εκτέλεση του προγράμματος θα συνεχίσει εντός του μπλοκ εντολών του βρόγχου επανάληψης (while) το οποίο είναι υπεύθυνο για τον χειροκίνητο έλεγχο της χορδής Σολ (G). Πιο αναλυτικά, εντός του μπλοκ εντολών του βρόγχου επανάληψης (while) γίνεται κλήση της συνάρτησης Manual_Step_Motor_Control(), η οποία αναλύεται σε επόμενη ενότητα και είναι υπεύθυνη για την ανάγνωση των πλήκτρων LEFT, RIGHT, UP, DOWN και SELECT. Ύστερα, πραγματοποιείται μια καθυστέρηση των 50msec εξαιτίας της εντολής καθυστέρησης delay(50) που εκτελείται αμέσως μετά. Έπειτα, εφαρμόζεται ξανά η τεχνική ελέγχου (if – else if) ώστε να επιτευχθεί η περιστροφή του βηματικού κινητήρα της χορδής Σολ (G) είτε ωρολογιακά, είτε αντί-ωρολογιακά. Ουσιαστικά, εκτελείται η εντολή ελέγχου if (Turn_Right = = 1), η οποία ελέγχει αν η τιμή της σημαίας (flag) με όνομα (Turn_Right) είναι ίση με ένα. Συνεπώς, αν έχει πατηθεί το πλήκτρο RIGHT, τότε η συνθήκη ελέγχου (if) ικανοποιείται και εκτελείται το αντίστοιχο μπλοκ εντολών. Πιο συγκεκριμένα, εντός του μπλοκ εντολών της συνθήκης ελέγχου (if), εκτελείται η συνθήκη επανάληψης (for), η οποία εκτελεί το μπλοκ εντολών της είκοσι ένα φορές. Όσον αφορά το μπλοκ εντολών του βρόγχου επανάληψης (for) γίνεται κλήση της συνάρτησης Clockwise_GStep_Motor (), η οποία αποσκοπεί στην ωρολογιακή περιστροφή του άξονα του βηματικού κινητήρα και κατά συνέπεια στην ωρολογιακή περιστροφή του κλειδιού της χορδής Σολ (G). Αφού ολοκληρωθεί η εκτέλεση του μπλοκ εντολών του βρόγχου επανάληψης (for), τότε γίνεται μηδενισμός της σημαίας (flag) με όνομα (Turn_Right) και έτσι ολοκληρώνεται το μπλοκ εντολών της εντολής ελέγχου if (Turn_Right = = 1). Αν όμως δεν ικανοποιείτε η συνθήκη ελέγχου (if), τότε εκτελείται η εντολή ελέγχου else if (Turn_Left = = 1), η οποία ελέγχει αν η τιμή της σημαίας (flag) με όνομα (Turn_Left) είναι ίση με ένα. Συνεπώς, στην περίπτωση που πατηθεί το πλήκτρο LEFT, τότε η συνθήκη ελέγχου (else if) ικανοποιείται και εκτελείται το αντίστοιχο μπλοκ εντολών. Πιο αναλυτικά, το μπλοκ εντολών της συνθήκης ελέγχου (else if), περιλαμβάνει μία συνθήκη επανάληψης (for), η οποία εκτελεί το μπλοκ εντολών της είκοσι ένα φορές. Στο εσωτερικό του μπλοκ εντολών του βρόγχου επανάληψης (for), γίνεται κλήση της συνάρτησης Counterclockwise_GStep_Motor (), η οποία αποσκοπεί στην αντί-ωρολογιακή περιστροφή του άξονα του βηματικού κινητήρα και κατά συνέπεια

στην αντί-ωρολογιακή περιστροφή του κλειδιού της χορδής Σολ (G). Αφότου ολοκληρωθεί η εκτέλεση του μπλοκ εντολών του βρόγχου επανάληψης (for), τότε γίνεται μηδενισμός της σημαίας (flag) με όνομα (Turn_Left) και έτσι ολοκληρώνεται το μπλοκ εντολών της εντολής ελέγχου else if (Turn_Left = = 1). Στην περίπτωση όμως που δεν πατηθεί κανένα κουμπί, η εκτέλεση του προγράμματος παραμένει εντός του βρόγχου (while). Αν όμως, πατηθεί το πλήκτρο SELECT, όπως θα αναλυθεί σε επόμενη ενότητα, η σημαία (flag) με όνομα (Finish_Control) λαμβάνει την τιμή ένα και έτσι η εκτέλεση του προγράμματος επιστρέφει στο χειροκίνητο μενού. Με αυτόν τον τρόπο, ολοκληρώνεται ο χειροκίνητος έλεγχος του κλειδιού της χορδής Σολ (G).

Από την άλλη πλευρά, αν δεν ικανοποιηθεί η συνθήκη else if (manual_menu = = 4), τότε εκτελείται η εντολή ελέγχου else if (manual_menu = = 5), η οποία ελέγχει αν η τιμή της μεταβλητής (manual_menu) είναι ίση με πέντε. Συνεπώς, αν ικανοποιείται η συγκεκριμένη συνθήκη, τότε εκτελείται το μπλοκ εντολών της συνθήκης ελέγχου (else if). Με αυτόν τον τρόπο, το σύστημα αντιλαμβάνεται ότι ο χρήστης έχει επιλέξει στο μενού τον χειροκίνητο έλεγχο της χορδής Σι (B), χωρίς όμως να έχει οριστικοποιήσει την επιλογή του! Πιο συγκεκριμένα, εντός του μπλοκ εντολών της συνθήκης ελέγχου (else if), εκτελούνται με κατάλληλή σειρά οι εντολές lcd.setcursor() και lcd.print(“ “) η οποίες εμφανίζουν στις κατάλληλες στήλες και στις δύο γραμμές το χειροκίνητο μενού (E A D G >B e Back) με την επιλογή να βρίσκεται στην χορδή Σι (B). Έπειτα, εντός του μπλοκ εντολών της συνθήκης (else if), εκτελείται η εντολή ελέγχου if (Manual_Select = = 1) η οποία αποτελεί ορολογιακά μία φωλιασμένη συνθήκη ελέγχου (if) και ελέγχει αν η σημαία (flag) με όνομα (Manual_Select) είναι ίση με την τιμή ένα. Ουσιαστικά η φωλιασμένη συνθήκη ελέγχου (if) ελέγχει αν ο χρήστης έχει οριστικοποιήσει την επιλογή για τον χειροκίνητο έλεγχο της χορδής Σι (B). Αν λοιπόν ικανοποιείται η συγκεκριμένη συνθήκη ελέγχου της φωλιασμένης (if), τότε ο χρήστης έχει οριστικοποιήσει την επιλογή του και εκτελείται το μπλοκ εντολών της φωλιασμένης συνθήκης ελέγχου (if). Εντός λοιπόν του μπλοκ εντολών της φωλιασμένης συνθήκης ελέγχου (if), γίνονται πρώτα κάποιες αρχικοποιήσεις οι οποίες αφορούν τον μηδενισμό της σημαίας (flag) με όνομα (Manual_Select) καθώς και της σημαίας (flag) με όνομα (Finish_Control). Ύστερα, εκτελούνται με κατάλληλη σειρά οι εντολές lcd.clear(), lcd.setCursor() και lcd.print(“ “), ώστε να εμφανιστεί το μήνυμα (Press Right or Left Buttons) στην LCD οθόνη. Το μήνυμα αυτό παραμένει στην LCD οθόνη για 3sec εξαιτίας της εντολής delay(3000) που εκτελείται αμέσως μετά. Σκοπός του μηνύματος αυτού, είναι να πληροφορήσει τον χρήστη ότι για να ελέγξει χειροκίνητα τον βηματικό κινητήρα της χορδής Σι (B), θα πρέπει να χρησιμοποιήσει τα πλήκτρα LEFT και RIGHT που βρίσκονται ενσωματωμένα στο πληκτρολόγιο της πλακέτας της LCD οθόνης. Στην συνέχεια, εκτελούνται ξανά οι εντολές lcd.clear(), lcd.setCursor() και lcd.print(“ “) με την κατάλληλη σειρά, ώστε να εμφανιστεί το μήνυμα (Press SELECT to go back) στην LCD οθόνη. Ουσιαστικά, σκοπός του μηνύματος αυτού είναι να πληροφορήσει τον χρήστη ότι πιέζοντας το πλήκτρο SELECT, θα επιστρέψει στο χειροκίνητο μενού και ότι η διαδικασία του χειροκίνητου ελέγχου της χορδής Σι (B) θα σταματήσει. Ύστερα, εκτελείται η εντολή επανάληψης while (Finish_Control = = 0), η οποία ελέγχει αν η τιμή της σημαίας (flag) με όνομα (Finish_Control) είναι ίση με μηδέν. Ουσιαστικά, το μπλοκ εντολών του βρόγχου επανάληψης (while) επαναλαμβάνεται όσο η συνθήκη συνεχίζει να ικανοποιείται. Η σημαία (flag) με όνομα (Finish_Control) έχει αρχικοποιηθεί πριν τον βρόγχο επανάληψης (while) και έχει λάβει την τιμή μηδέν. Συνεπώς, η εκτέλεση του προγράμματος θα συνεχίσει εντός του μπλοκ εντολών του βρόγχου επανάληψης (while) το οποίο είναι

υπεύθυνο για τον χειροκίνητο έλεγχο της χορδής Σι (B). Πιο συγκεκριμένα, εντός του μπλοκ εντολών του βρόγχου επανάληψης (while) γίνεται κλήση της συνάρτησης Manual_Step_Motor_Control(), η οποία είναι υπεύθυνη για την ανάγνωση των πλήκτρων LEFT, RIGHT, UP, DOWN και SELECT και αναλύεται σε επόμενη ενότητα. Έπειτα, πραγματοποιείται μια καθυστέρηση των 50msec εξαιτίας της εντολής καθυστέρησης delay(50) που εκτελείται αμέσως μετά. Εν συνεχεία, εφαρμόζεται ξανά η τεχνική ελέγχου (if – else if) ώστε να περιστραφεί ο βηματικός κινητήρας της χορδής Σι (B) είτε ωρολογιακά, είτε αντί-ωρολογιακά. Πιο συγκεκριμένα, εκτελείται η εντολή ελέγχου if (Turn_Right = = 1), η οποία ελέγχει αν η τιμή της σημαίας (flag) με όνομα (Turn_Right) είναι ίση με ένα. Αν λοιπόν έχει πατηθεί το πλήκτρο RIGHT, τότε η συνθήκη ελέγχου (if) ικανοποιείται και εκτελείται το αντίστοιχο μπλοκ εντολών. Εντός του μπλοκ εντολών της συνθήκης ελέγχου (if), εκτελείται η συνθήκη επανάληψης (for), η οποία εκτελεί το μπλοκ εντολών της είκοσι ένα φορές. Στο εσωτερικό του μπλοκ εντολών του βρόγχου επανάληψης (for) γίνεται κλήση της συνάρτησης Clockwise_BStep_Motor (), η οποία έχει σκοπό την ωρολογιακή περιστροφή του άξονα του βηματικού κινητήρα και κατά συνέπεια την ωρολογιακή περιστροφή του κλειδιού της χορδής Σι (B). Αφού ολοκληρωθεί η εκτέλεση του μπλοκ εντολών του βρόγχου επανάληψης (for), τότε γίνεται μηδενισμός της σημαίας (flag) με όνομα (Turn_Right) και έτσι ολοκληρώνεται το μπλοκ εντολών της εντολής ελέγχου if (Turn_Right = = 1). Στην περίπτωση όμως που δεν ικανοποιείτε η συνθήκη ελέγχου (if), τότε εκτελείται η εντολή ελέγχου else if (Turn_Left = = 1), η οποία ελέγχει αν η τιμή της σημαίας (flag) με όνομα (Turn_Left) είναι ίση με ένα. Επομένως, αν πατηθεί το πλήκτρο LEFT, τότε η συνθήκη ελέγχου (else if) ικανοποιείται και εκτελείται το αντίστοιχο μπλοκ εντολών. Πιο συγκεκριμένα, εντός του μπλοκ εντολών της συνθήκης ελέγχου (else if), εκτελείται η συνθήκη επανάληψης (for), η οποία εκτελεί το μπλοκ εντολών της είκοσι ένα φορές. Στο εσωτερικό του μπλοκ εντολών του βρόγχου επανάληψης (for) γίνεται κλήση της συνάρτησης Counterclockwise_BStep_Motor (), η οποία έχει σκοπό την αντί-ωρολογιακή περιστροφή του άξονα του βηματικού κινητήρα και κατά συνέπεια την αντί-ωρολογιακή περιστροφή του κλειδιού της χορδής Σι (B). Αφού ολοκληρωθεί η εκτέλεση του μπλοκ εντολών του βρόγχου επανάληψης (for), τότε γίνεται μηδενισμός της σημαίας (flag) με όνομα (Turn_Left) και έτσι ολοκληρώνεται το μπλοκ εντολών της εντολής ελέγχου else if (Turn_Left = = 1). Πάραυτα, αν δε πατηθεί κανένα κουμπί η εκτέλεση του προγράμματος παραμένει εντός του βρόγχου (while). Στην περίπτωση όμως που πατηθεί το πλήκτρο SELECT, όπως θα αναλυθεί σε επόμενη ενότητα, η σημαία (flag) με όνομα (Finish_Control) λαμβάνει την τιμή ένα και έτσι η εκτέλεση του προγράμματος επιστρέφει στο χειροκίνητο μενού. Με αυτόν τον τρόπο, ολοκληρώνεται ο χειροκίνητος έλεγχος του κλειδιού της χορδής Σι (B).

Στην περίπτωση όμως που δεν ικανοποιείται η συνθήκη else if(manual_menu = = 5), τότε εκτελείται η εντολή ελέγχου else if (manual_menu = = 6), η οποία ελέγχει αν η τιμή της μεταβλητής (manual_menu) είναι ίση με έξι. Επομένως, αν ικανοποιείται η συγκεκριμένη συνθήκη, τότε εκτελείται το μπλοκ εντολών της συνθήκης ελέγχου (else if). Αυτό σημαίνει ότι ο χρήστης έχει επιλέξει στο μενού τον χειροκίνητο έλεγχο της χορδής Μι καντίνι (e), χωρίς όμως να έχει οριστικοποιήσει ακόμα την επιλογή του! Ουσιαστικά, εντός του μπλοκ εντολών της συνθήκης ελέγχου (else if), εκτελούνται με κατάλληλή σειρά οι εντολές lcd.setcursor() και lcd.print(“ “) η οποίες εμφανίζουν στις κατάλληλες στήλες και στις δύο γραμμές το χειροκίνητο μενού (E A D G B >e Back) με την επιλογή να βρίσκεται στην χορδή Μι καντίνι (e). Εν συνεχεία, εντός του μπλοκ εντολών της συνθήκης (else if), εκτελείται η εντολή ελέγχου if (Manual_Select = = 1), η οποία

αποτελεί ορολογιακά μία φωλιασμένη συνθήκη ελέγχου (if) και ελέγχει αν η σημαία (flag) με όνομα (Manual_Select) είναι ίση με την τιμή ένα. Πιο αναλυτικά, η φωλιασμένη συνθήκη ελέγχου (if) εντοπίζει αν ο χρήστης έχει οριστικοποιήσει την επιλογή για τον χειροκίνητο έλεγχο της χορδής Μι καντίνι (e). Αν λοιπόν ικανοποιείται η συγκεκριμένη συνθήκη ελέγχου της φωλιασμένης (if), τότε ο χρήστης έχει οριστικοποιήσει την επιλογή του και εκτελείται το μπλοκ εντολών της φωλιασμένης συνθήκης ελέγχου (if). Πιο συγκεκριμένα, εντός του μπλοκ εντολών της φωλιασμένης συνθήκης ελέγχου (if), γίνονται πρώτα κάποιες αρχικοποιήσεις οι οποίες αφορούν τον μηδενισμό της σημαίας (flag) με όνομα (Manual_Select) καθώς και της σημαίας (flag) με όνομα (Finish_Control). Έπειτα, εκτελούνται οι εντολές lcd.clear(), lcd.setCursor() και lcd.print(“ “) με κατάλληλη σειρά, ώστε να εμφανιστεί το μήνυμα (Press Right or Left Buttons) στην LCD οθόνη. Το μήνυμα αυτό παραμένει στην LCD οθόνη για 3sec εξαιτίας της εντολής delay(3000) που εκτελείται αμέσως μετά. Σκοπός του μηνύματος αυτού, είναι να πληροφορήσει τον χρήστη ότι για να ελέγξει χειροκίνητα τον βηματικό κινητήρα της χορδής Μι καντίνι (e), θα πρέπει να χρησιμοποιήσει τα πλήκτρα LEFT και RIGHT που βρίσκονται ενσωματωμένα στο πληκτρολόγιο της πλακέτας της LCD οθόνης. Ύστερα, εκτελούνται ξανά οι εντολές lcd.clear(), lcd.setCursor() και lcd.print(“ “) με την κατάλληλη σειρά, ώστε να εμφανιστεί το μήνυμα (Press SELECT to go back) στην LCD οθόνη. Το μήνυμα αυτό αποσκοπεί στην πληροφόρηση του χρήστη ότι πιέζοντας το πλήκτρο SELECT, θα επιστρέψει στο χειροκίνητο μενού και θα ολοκληρωθεί η διαδικασία του χειροκίνητου ελέγχου της χορδής Μι καντίνι (e). Εν συνεχεία, εκτελείται η εντολή επανάληψης while (Finish_Control = = 0), η οποία ελέγχει αν η τιμή της σημαίας (flag) με όνομα (Finish_Control) είναι ίση με μηδέν. Πιο συγκεκριμένα, το μπλοκ εντολών του βρόγχου επανάληψης (while) επαναλαμβάνεται όσο η συνθήκη συνεχίζει να ικανοποιείται. Επιπλέον, η σημαία (flag) με όνομα (Finish_Control) έχει αρχικοποιηθεί πριν τον βρόγχο επανάληψης (while) και έχει λάβει την τιμή μηδέν. Επομένως, η εκτέλεση του προγράμματος θα συνεχίσει εντός του μπλοκ εντολών του βρόγχου επανάληψης (while) το οποίο είναι υπεύθυνο για τον χειροκίνητο έλεγχο της χορδής Μι καντίνι (e). Πιο αναλυτικά, εντός του μπλοκ εντολών του βρόγχου επανάληψης (while) γίνεται κλήση της συνάρτησης Manual_Step_Motor_Control(), η οποία αναλύεται σε επόμενη ενότητα και είναι υπεύθυνη για την ανάγνωση των πλήκτρων LEFT, RIGHT, UP, DOWN και SELECT. Ύστερα, πραγματοποιείται μια καθυστέρηση των 50msec εξαιτίας της εντολής καθυστέρησης delay(50) που εκτελείται αμέσως μετά. Έπειτα, εφαρμόζεται ξανά η τεχνική ελέγχου (if – else if) ώστε να επιτευχθεί η περιστροφή του βηματικού κινητήρα της χορδής Μι καντίνι (e) είτε ωρολογιακά, είτε αντί-ωρολογιακά. Ουσιαστικά, εκτελείται η εντολή ελέγχου if (Turn_Right = = 1), η οποία ελέγχει αν η τιμή της σημαίας (flag) με όνομα (Turn_Right) είναι ίση με ένα. Συνεπώς, αν έχει πατηθεί το πλήκτρο RIGHT, τότε η συνθήκη ελέγχου (if) ικανοποιείται και εκτελείται το αντίστοιχο μπλοκ εντολών. Πιο συγκεκριμένα, εντός του μπλοκ εντολών της συνθήκης ελέγχου (if), εκτελείται η συνθήκη επανάληψης (for), η οποία εκτελεί το μπλοκ εντολών της είκοσι ένα φορές. Όσον αφορά το μπλοκ εντολών του βρόγχου επανάληψης (for) γίνεται κλήση της συνάρτησης Clockwise_eStep_Motor (), η οποία αποσκοπεί στην ωρολογιακή περιστροφή του άξονα του βηματικού κινητήρα και κατά συνέπεια στην ωρολογιακή περιστροφή του κλειδιού της χορδής Μι καντίνι (e). Αφού ολοκληρωθεί η εκτέλεση του μπλοκ εντολών του βρόγχου επανάληψης (for), τότε γίνεται μηδενισμός της σημαίας (flag) με όνομα (Turn_Right) και έτσι ολοκληρώνεται το μπλοκ εντολών της εντολής ελέγχου if (Turn_Right = = 1). Αν όμως δεν ικανοποιείτε η συνθήκη ελέγχου (if), τότε εκτελείται η

εντολή ελέγχου else if (Turn_Left = = 1), η οποία ελέγχει αν η τιμή της σημαίας (flag) με όνομα (Turn_Left) είναι ίση με ένα. Συνεπώς, στην περίπτωση που πατηθεί το πλήκτρο LEFT, τότε η συνθήκη ελέγχου (else if) ικανοποιείται και εκτελείται το αντίστοιχο μπλοκ εντολών. Πιο αναλυτικά, το μπλοκ εντολών της συνθήκης ελέγχου (else if), περιλαμβάνει μία συνθήκη επανάληψης (for), η οποία εκτελεί το μπλοκ εντολών της είκοσι ένα φορές. Στο εσωτερικό του μπλοκ εντολών του βρόγχου επανάληψης (for), γίνεται κλήση της συνάρτησης Counterclockwise_eStep_Motor (), η οποία αποσκοπεί στην αντί-ωρολογιακή περιστροφή του άξονα του βηματικού κινητήρα και κατά συνέπεια στην αντί-ωρολογιακή περιστροφή του κλειδιού της χορδής Μι καντίνι (e). Αφότου ολοκληρωθεί η εκτέλεση του μπλοκ εντολών του βρόγχου επανάληψης (for), τότε γίνεται μηδενισμός της σημαίας (flag) με όνομα (Turn_Left) και έτσι ολοκληρώνεται το μπλοκ εντολών της εντολής ελέγχου else if (Turn_Left = = 1). Στην περίπτωση όμως που δεν πατηθεί κανένα κουμπί, η εκτέλεση του προγράμματος παραμένει εντός του βρόγχου (while). Αν όμως, πατηθεί το πλήκτρο SELECT, όπως θα αναλυθεί σε επόμενη ενότητα, η σημαία (flag) με όνομα (Finish_Control) λαμβάνει την τιμή ένα και έτσι η εκτέλεση του προγράμματος επιστρέφει στο χειροκίνητο μενού. Με αυτόν τον τρόπο, ολοκληρώνεται ο χειροκίνητος έλεγχος του κλειδιού της χορδής Μι καντίνι (e).

Από την άλλη πλευρά, αν δεν ικανοποιηθεί καμία από τις προηγούμενες συνθήκες (else if), τότε εκτελείται η τελευταία εντολή ελέγχου else if (manual_menu = = 7), η οποία ελέγχει αν η τιμή της μεταβλητής (manual_menu) είναι ίση με επτά. Συνεπώς, αν ικανοποιείται η συγκεκριμένη συνθήκη, τότε εκτελείται το μπλοκ εντολών της συνθήκης ελέγχου (else if). Με αυτόν τον τρόπο, το σύστημα αντιλαμβάνεται ότι ο χρήστης έχει επιλέξει στο μενού την επιλογή της ολοκλήρωσης του χειροκίνητου ελέγχου του συστήματος, χωρίς όμως να έχει οριστικοποιήσει την επιλογή του! Πιο συγκεκριμένα, εντός του μπλοκ εντολών της συνθήκης ελέγχου (else if), εκτελούνται με κατάλληλή σειρά οι εντολές lcd.setcursor() και lcd.print(“ “) η οποίες εμφανίζουν στις κατάλληλες στήλες και στις δύο γραμμές το χειροκίνητο μενού (E A D G B e >Back) με το βελάκι (>) να βρίσκεται στην επιλογή (Back). Έπειτα, εντός του μπλοκ εντολών της συνθήκης (else if), εκτελείται η εντολή ελέγχου if (Manual_Select = = 1) η οποία αποτελεί ορολογιακά μία φωλιασμένη συνθήκη ελέγχου (if) και ελέγχει αν η σημαία (flag) με όνομα (Manual_Select) είναι ίση με την τιμή ένα. Ουσιαστικά, η φωλιασμένη συνθήκη ελέγχου (if) ελέγχει αν ο χρήστης έχει οριστικοποιήσει την επιλογή του για την ολοκλήρωση του χειροκίνητου ελέγχου του συστήματος. Συνεπώς, αν ο χρήστης έχει πιέσει το πλήκτρο SELECT για να οριστικοποιήσει την επιλογή του, τότε εκτελείται το μπλοκ εντολών της φωλιασμένης συνθήκης ελέγχου (if). Πιο αναλυτικά, το μπλοκ εντολών της φωλιασμένης συνθήκης ελέγχου (if) περιλαμβάνει την αρχικοποίηση της σημαίας (flag) με όνομα (Manual_Select) με την τιμή μηδέν. Με αυτόν τον τρόπο γίνεται επαναφορά του πλήκτρου SELECT από το πρόγραμμα. Αμέσως μετά, εκτελείται η τελευταία εντολή η οποία αρχικοποιεί την σημαία (flag) με όνομα (Manual_Finish) με την τιμή ένα. Αυτό έχει ως αποτέλεσμα η εντολή επανάληψης while (Manual_Finish = = 0) που εκτελείται στην αρχή, να έχει ψευδής συνθήκη. Έτσι, παύει να εκτελείται το μπλοκ εντολών της ολοκληρώνοντας κατ’ αυτόν τον τρόπο τον χειροκίνητο έλεγχο του συστήματος. Έτσι το πρόγραμμα ολοκληρώνει το μπλοκ εντολών της εντολής ελέγχου else if (manual_menu = = 7). Ύστερα, εκτελείται η τελευταία εντολή από το μπλοκ εντολών της εντολής επανάληψης while (Manual_Finish = = 0), η οποία είναι η εντολή καθυστέρησης delay(50) που προκαλεί μια καθυστέρηση των 50msec. Εφόσον, η συνθήκη εντός του

βρόγχου επανάληψης (while) δεν ικανοποιείται, τότε ολοκληρώνεται η εκτέλεση του μπλοκ εντολών του.

Τέλος, γίνεται κλήση της συνάρτησης Menu(), ώστε να εμφανιστεί στην LCD οθόνη ξανά το μενού επιλογών που περιλαμβάνει τα κουρδίσματα και τον χειροκίνητο έλεγχο. Με αυτόν τον τρόπο, το σύστημα επιτρέπει στον χρήστη να επιλέξει ξανά ποια ενέργεια επιθυμεί, κούρδισμα ή χειροκίνητο έλεγχο. Επίσης, εκτελείται η εντολή delay(200) η οποία δημιουργεί μια καθυστέρηση των 0.2msec. Έτσι, ολοκληρώνεται το μπλοκ εντολών της εντολής ελέγχου else if (x<800 && x!=689), η οποία ελέγχει ουσιαστικά πότε πατήθηκε το πλήκτρο SELECT. Επίσης, σε αυτό το σημείο ολοκληρώνεται και το μπλοκ εντολών της συνάρτησης ButtonRead(), η οποία αναλύθηκε εξονυχιστικά σε αυτό το κεφάλαιο.

Πίνακας 3.8: Ο κώδικας για τον χειροκίνητο έλεγχο του συστήματος από τον χρήστη (Manual Control) [34].

Κώδικας Προγραμματισμού (Μέρος 8ο)

```
else if (menu==4) //Else if menu variable is 4 then
   {
    Manual_Finish = 0; //Reset Manual_Finish variable
    manual_menu = 1; //Store number 1 at manual_menu variable
    lcd.clear(); //Clear LCD screen
    lcd.setCursor(0,0); //Set cursor at 1st column and 1st line
    lcd.print("Choose string!"); //Print message at LCD screen
    delay(2500); //Delay 2.5 sec
    lcd.clear(); //Clear LCD screen
    while(Manual_Finish == 0) //While Manual_Finish is 0 then
    {
     Manual_Button_Read(); //Call Function for LCD's button read
     if(manual_menu == 1) //If E string has been selected //If manual_menu variable is 1
then
      {
       lcd.setCursor(0,0); //Move the cursor to the 1st line and the 1st column
       lcd.print(">E"); //Print message to the lcd screen
       lcd.setCursor(0,1); //Move the cursor to the 2nd line and the 1st column
       lcd.print(" A"); //Print message to the lcd screen
       lcd.setCursor(2,0); //Move the cursor to the 1st line and the 3rd column
       lcd.print(" D"); //Print message to the lcd screen
       lcd.setCursor(2,1); //Move the cursor to the 2nd line and the 3rd column
       lcd.print(" G"); //Print message to the lcd screen
       lcd.setCursor(4,0); //Move the cursor to the 1st line and the 5th column
       lcd.print(" B"); //Print message to the lcd screen
       lcd.setCursor(4,1); //Move the cursor to the 2nd line and the 5th column
```

```
        lcd.print(" e"); //Print message to the lcd screen
        lcd.setCursor(6,0); //Move the cursor to the 1st line and the 7th column
        lcd.print(" Back"); //Print message to the lcd screen
        if (Manual_Select == 1) //If Manual_Select is 1, If the E string has been selected
then
        {
          Manual_Select = 0; //Reset Manual_Selected variable
          Finish_Control = 0; //Reset Finish_Control variable
          lcd.clear(); //Clear LCD screen
          lcd.setCursor(0,0); //Set cursor at 1st column and 1st line
          lcd.print(" Press Right or "); //Print message at LCD screen
          lcd.setCursor(0,1); //Set cursor at 1st column and 2nd line
          lcd.print("  Left Buttons  "); //Print message at LCD screen
          delay(3000); //Delay 3 sec
          lcd.clear(); //Clear LCD screen
          lcd.setCursor(0,0); //Set cursor at 1st column and 1st line
          lcd.print("  Press SELECT  "); //Print message at LCD screen
          lcd.setCursor(0,1); //Set cursor at 1st column and 2nd line
          lcd.print("   to go back   "); //Print message at LCD screen
          while (Finish_Control == 0) //While Finish_Control variable is 0 then
          {
            Manual_Step_Motor_Control(); //Call Function for reading LCD's shield buttons
            delay(50); //Delay 50 msec
            if (Turn_Right == 1) //If Turn_Right flag is 1 then
            {
              //Serial.println("Turn Right");
              for(i=0;i<=20;i++) //For 21 reps
              {
                Clockwise_EStep_Motor (); //Turn E string's step motor clockwise
              }
              Turn_Right = 0; //Reset Turn_Right flag
            } //if
            else if (Turn_Left == 1) //Else if Turn_Left flag is 1 then
            {
              for(i=0;i<=20;i++) //For 21 reps
              {
                Counterclockwise_EStep_Motor (); //Turn E string's step motor
counterclockwise
              }
```

```
          Turn_Left = 0; //Reset Turn_Left flag
        } //else if (Turn_Left...
      } //while (Finish_Control...
    } //if (Manual_Select...
  } //if (manual_menu...
  else if (manual_menu == 2) //If A string has been selected
  {
   lcd.setCursor(0,0); //Move the cursor to the 1st line and the 1st column
   lcd.print(" E"); //Print message to the lcd screen
   lcd.setCursor(0,1); //Move the cursor to the 2nd line and the 1st column
   lcd.print(">A"); //Print message to the lcd screen
   lcd.setCursor(2,0); //Move the cursor to the 1st line and the 3rd column
   lcd.print(" D"); //Print message to the lcd screen
   lcd.setCursor(2,1); //Move the cursor to the 2nd line and the 3rd column
   lcd.print(" G"); //Print message to the lcd screen
   lcd.setCursor(4,0); //Move the cursor to the 1st line and the 5th column
   lcd.print(" B"); //Print message to the lcd screen
   lcd.setCursor(4,1); //Move the cursor to the 2nd line and the 5th column
   lcd.print(" e"); //Print message to the lcd screen
   lcd.setCursor(6,0); //Move the cursor to the 1st line and the 7th column
   lcd.print(" Back"); //Print message to the lcd screen
   if (Manual_Select == 1) //If Manual_Select flag is 1 then
   {
    Manual_Select = 0; //Reset Manual_Select flag
    Finish_Control = 0; //Reset Finish_Control flag
    lcd.clear(); //Clear LCD screen
    lcd.setCursor(0,0); //Set cursor at 1st column and 1st line
    lcd.print(" Press Right or "); //Print message at LCD screen
    lcd.setCursor(0,1); //Set cursor at 1st column and 2nd line
    lcd.print("  Left Buttons  "); //Print message at LCD screen
    delay(3000); //Delay 3 sec
    lcd.clear(); //Clear LCD screen
    lcd.setCursor(0,0); //Set cursor at 1st column and 1st line
    lcd.print("  Press SELECT  "); //Print message at LCD screen
    lcd.setCursor(0,1); //Set cursor at 1st column and 2nd line
    lcd.print("   to go back   "); //Print message at LCD screen
    while (Finish_Control == 0) //While Finish_Control flag is 0 then
    {
     Manual_Step_Motor_Control(); //Call of Manual_Step_Motor_Control function
```

```
        delay(50); //Delay 50 msec
        if (Turn_Right == 1) //If Turn right flag is 1 then
        {
         for(i=0;i<=20;i++) //For 21 reps
         {
          Clockwise_AStep_Motor (); //Turn A Step Motor Clockwise
         }
         Turn_Right = 0; //Reset Turn_Right flag
        }//if (Turn_Right)...
        else if (Turn_Left == 1) //Else if Turn_Left flag is 1 then
        {
         for(i=0;i<=20;i++) //For 21 reps
         {
          Counterclockwise_AStep_Motor (); //Turn A step motor counter clockwise
         }
         Turn_Left = 0; //Reset Turn Left flag
        } //else if (Turn_Left)...
       } //while (Finish_Control)...
      } //if (Manual_Select == 1)...
     } //else if (manual_menu == 1)...
     else if (manual_menu == 3) //If D string has been selected
     {
      lcd.setCursor(0,0); //Move the cursor to the 1st line and the 1st column
      lcd.print(" E"); //Print message to the lcd screen
      lcd.setCursor(0,1); //Move the cursor to the 2nd line and the 1st column
      lcd.print(" A"); //Print message to the lcd screen
      lcd.setCursor(2,0); //Move the cursor to the 1st line and the 3rd column
      lcd.print(">D"); //Print message to the lcd screen
      lcd.setCursor(2,1); //Move the cursor to the 2nd line and the 3rd column
      lcd.print(" G"); //Print message to the lcd screen
      lcd.setCursor(4,0); //Move the cursor to the 1st line and the 5th column
      lcd.print(" B"); //Print message to the lcd screen
      lcd.setCursor(4,1); //Move the cursor to the 2nd line and the 5th column
      lcd.print(" e"); //Print message to the lcd screen
      lcd.setCursor(6,0); //Move the cursor to the 1st line and the 7th column
      lcd.print(" Back"); //Print message to the lcd screen
      if (Manual_Select == 1) //If Manual_Select flag is 1 then
      {
       Manual_Select = 0; //Reset Manual_Reset
```

```
        Finish_Control = 0; //Reset Finish_Control
        lcd.clear(); //Clear LCD screen
        lcd.setCursor(0,0); //Set cursor at 1st column and 1st line
        lcd.print(" Press Right or "); //Print message at LCD screen
        lcd.setCursor(0,1); //Set cursor at 1st column and 2nd line
        lcd.print("  Left Buttons  "); //Print message at LCD screen
        delay(3000); //Delay 3 sec
        lcd.clear(); //Clear LCD screen
        lcd.setCursor(0,0); //Set cursor at 1st column and 1st line
        lcd.print("  Press SELECT  "); //Print message at LCD screen
        lcd.setCursor(0,1); //Set cursor at 1st column and 2nd line
        lcd.print("   to go back   "); //Print message at LCD screen
        while (Finish_Control == 0) // While Finish_Control is 0 then
        {
          Manual_Step_Motor_Control(); //Call Manual_Step_Motor_Control
          delay(50); //Delay 50 msec
          if (Turn_Right == 1) //If Turn_Right flag is 1 then
          {
            for(i=0;i<=20;i++) //For 21 reps
            {
              Clockwise_DStep_Motor (); //Turn D step motor clockwise
            }
            Turn_Right = 0; //Reset Turn_Right flag
          } //if (Turn_Right == 1)...
          else if (Turn_Left == 1) //Else if Turn_Left flag is 1 then
          {
            for(i=0;i<=20;i++) //For 21 reps
            {
              Counterclockwise_DStep_Motor (); //Turn D step motor Counter Clockwise
            }
            Turn_Left = 0; //Reset Turn_Left flag
          } //else if (Turn_Left == 1)
        } //while (Finish_Control == 0)
      } //if (Manual_Select == 1)
    } //else if (manual_menu ==3)
    else if (manual_menu == 4) //If G string has been selected
    {
      lcd.setCursor(0,0); //Move the cursor to the 1st line and the 1st column
      lcd.print(" E"); //Print message to the lcd screen
```

```
lcd.setCursor(0,1); //Move the cursor to the 2nd line and the 1st column
lcd.print(" A"); //Print message to the lcd screen
lcd.setCursor(2,0); //Move the cursor to the 1st line and the 3rd column
lcd.print(" D"); //Print message to the lcd screen
lcd.setCursor(2,1); //Move the cursor to the 2nd line and the 3rd column
lcd.print(">G"); //Print message to the lcd screen
lcd.setCursor(4,0); //Move the cursor to the 1st line and the 5th column
lcd.print(" B"); //Print message to the lcd screen
lcd.setCursor(4,1); //Move the cursor to the 2nd line and the 5th column
lcd.print(" e"); //Print message to the lcd screen
lcd.setCursor(6,0); //Move the cursor to the 1st line and the 7th column
lcd.print(" Back"); //Print message to the lcd screen
if (Manual_Select == 1) //If Manual_Select flag is 1 then
{
 Manual_Select = 0; //Reset Manual_Select
 Finish_Control = 0; //Reset Finish_Control
 lcd.clear(); //Clear LCD screen
 lcd.setCursor(0,0); //Set cursor at 1st column and 1st line
 lcd.print(" Press Right or "); //Print message at LCD screen
 lcd.setCursor(0,1); //Set cursor at 1st column and 2nd line
 lcd.print("  Left Buttons  "); //Print message at LCD screen
 delay(3000); //Delay 3 sec
 lcd.clear(); //Clear LCD screen
 lcd.setCursor(0,0); //Set cursor at 1st column and 1st line
 lcd.print("  Press SELECT  "); //Print meassage at LCD screen
 lcd.setCursor(0,1); //Set cursor at 1st column and 2nd line
 lcd.print("   to go back   "); //Print message at LCD screen
 while (Finish_Control == 0) //While Finish_Control flag is 0 then
 {
  Manual_Step_Motor_Control(); //Call Manual_Step_Motor_Control function
  delay(50); //Delay 50 msec
  if (Turn_Right == 1) //If Turn_Right flag is 1 then
  {
   for(i=0;i<=20;i++) //For 21 reps
   {
    Clockwise_GStep_Motor (); //Turn G step motor clockwise
   }
   Turn_Right = 0; //Reset Turn_Right flag
  } //if (Turn_Right...
```

```
        else if (Turn_Left == 1) //Else if Turn_Left flag is 1 then
        {
         for(i=0;i<=20;i++) //For 21 reps
         {
          Counterclockwise_GStep_Motor (); //Turn G step motor counter clockwise
         }
         Turn_Left = 0; //Reset Turn_Left flag
        } //else if (Turn_Left...
       } //while (Finish_Control...
      } //if (Manual_Select...
     } //else if (manual_menu...
     else if (manual_menu == 5) //if the B string has been selected
     {
      lcd.setCursor(0,0); //Move the cursor to the 1st line and the 1st column
      lcd.print(" E"); //Print message to the lcd screen
      lcd.setCursor(0,1); //Move the cursor to the 2nd line and the 1st column
      lcd.print(" A"); //Print message to the lcd screen
      lcd.setCursor(2,0); //Move the cursor to the 1st line and the 3rd column
      lcd.print(" D"); //Print message to the lcd screen
      lcd.setCursor(2,1); //Move the cursor to the 2nd line and the 3rd column
      lcd.print(" G"); //Print message to the lcd screen
      lcd.setCursor(4,0); //Move the cursor to the 1st line and the 5th column
      lcd.print(">B"); //Print message to the lcd screen
      lcd.setCursor(4,1); //Move the cursor to the 2nd line and the 5th column
      lcd.print(" e"); //Print message to the lcd screen
      lcd.setCursor(6,0); //Move the cursor to the 1st line and the 7th column
      lcd.print(" Back"); //Print message to the lcd screen
      if (Manual_Select == 1) //if Manual_Select is 1 then
      {
       Manual_Select = 0; //Reset Manual_Select
       Finish_Control = 0; //Reset Finish_Control
       lcd.clear(); //Clear LCD screen
       lcd.setCursor(0,0); //Set cursor at 1st column and 1st line
       lcd.print(" Press Right or "); //Print message at LCD screen
       lcd.setCursor(0,1); //Set cursor at 1st column and 2nd line
       lcd.print("  Left Buttons  "); //Print message at LCD screen
       delay(3000); //Delay 3 sec
       lcd.clear(); //Clear LCD screen
       lcd.setCursor(0,0); //Set cursor at 1st column and 1st line
```

```
        lcd.print("  Press SELECT  "); //Print message at LCD screen
        lcd.setCursor(0,1); //Set cursor at 1st column and 2nd line
        lcd.print("   to go back   "); // Print message at LCD screen
        while (Finish_Control == 0) //While Finish_Control is 0 then
        {
         Manual_Step_Motor_Control(); //Call Manual_Step_Motor_Control function
         delay(50); //delay 50 msec
         if (Turn_Right == 1) //If Turn_Right flag is 1 then
         {
          for(i=0;i<=20;i++) //For 21 reps
          {
           Clockwise_BStep_Motor (); //Turn B step motor clockwise
          }
          Turn_Right = 0; //Reset Turn_Right flag
         } //if (Turn_Right...
         else if (Turn_Left == 1) //Else if Turn_Left is 1 then
         {
          for(i=0;i<=20;i++) //For 21 reps
          {
           Counterclockwise_BStep_Motor (); //Turn B step motor counter clockwise
          }
          Turn_Left = 0; // Reset Turn_Left flag
         } //else if (Turn_Left...
        } //while (Finish_Right...
       } //if (Manual_Select...
      } //else if (manual_menu...
      else if (manual_menu == 6) //If the e string has been selected
      {
       lcd.setCursor(0,0); //Move the cursor to the 1st line and the 1st column
       lcd.print(" E"); //Print message to the lcd screen
       lcd.setCursor(0,1); //Move the cursor to the 2nd line and the 1st column
       lcd.print(" A"); //Print message to the lcd screen
       lcd.setCursor(2,0); //Move the cursor to the 1st line and the 3rd column
       lcd.print(" D"); //Print message to the lcd screen
       lcd.setCursor(2,1); //Move the cursor to the 2nd line and the 3rd column
       lcd.print(" G"); //Print message to the lcd screen
       lcd.setCursor(4,0); //Move the cursor to the 1st line and the 5th column
       lcd.print(" B"); //Print message to the lcd screen
       lcd.setCursor(4,1); //Move the cursor to the 2nd line and the 5th column
```

```
        lcd.print(">e"); //Print message to the lcd screen
        lcd.setCursor(6,0); //Move the cursor to the 1st line and the 7th column
        lcd.print(" Back"); //Print message to the lcd screen
        if (Manual_Select == 1) //If Manual_Select is 1 then
        {
         Manual_Select = 0; //Reset Manual_Select
         Finish_Control = 0; //Reset Finish_Control
         lcd.clear(); //Clear LCD screen
         lcd.setCursor(0,0); //Set cursor at 1st column and 1st line
         lcd.print(" Press Right or "); //Print message at LCD screen
         lcd.setCursor(0,1); //Set cursor at 1st column and 2nd line
         lcd.print("  Left Buttons  "); //Print message at LCD screen
         delay(3000); //Delay 3 sec
         lcd.clear(); //Clear LCD screen
         lcd.setCursor(0,0); //Set cursor at 1st column and 1st line
         lcd.print("  Press SELECT  "); //Print , message at LCD screen
         lcd.setCursor(0,1); //Set cursor at 1st column and 2nd line
         lcd.print("   to go back   "); //Print message at LCD screen
         while (Finish_Control == 0) //While Finish_Control is 0 then
         {
          Manual_Step_Motor_Control(); //Call Manual_Step_Motor_Control function
          delay(50); //Delay 50 msec
          if (Turn_Right == 1) //If Turn_Right flag is 1 then
          {
           for(i=0;i<=20;i++) //For 21 reps
           {
            Clockwise_eStep_Motor (); //Turn e Step Motor Clockwise
           }
           Turn_Right = 0; //Reset Turn_Right flag
          } //if (Turn_Right...
          else if (Turn_Left == 1) //Else if Turn_Left flag is 1 then
          {
           for(i=0;i<=20;i++) //For 21 reps
           {
            Counterclockwise_eStep_Motor (); //Turn e Step_Motor Counterclockwise
           }
           Turn_Left = 0; //Reset Turn_Left flag
          } //else if (Turn_Left...
         } //while (Finish_Control...
```

```
        } //if (Manual_Select...
       } //else if (manual_menu...
       else if (manual_menu == 7) //If Back to the main menu has been selected
       {
        lcd.setCursor(0,0); //Move the cursor to the 1st line and the 1st column
        lcd.print(" E"); //Print message to the lcd screen
        lcd.setCursor(0,1); //Move the cursor to the 2nd line and the 1st column
        lcd.print(" A"); //Print message to the lcd screen
        lcd.setCursor(2,0); //Move the cursor to the 1st line and the 3rd column
        lcd.print(" D"); //Print message to the lcd screen
        lcd.setCursor(2,1); //Move the cursor to the 2nd line and the 3rd column
        lcd.print(" G"); //Print message to the lcd screen
        lcd.setCursor(4,0); //Move the cursor to the 1st line and the 5th column
        lcd.print(" B"); //Print message to the lcd screen
        lcd.setCursor(4,1); //Move the cursor to the 2nd line and the 5th column
        lcd.print(" e"); //Print message to the lcd screen
        lcd.setCursor(6,0); //Move the cursor to the 1st line and the 7th column
        lcd.print(">Back"); //Print message to the lcd screen
        if (Manual_Select == 1) //If Manual_Select is 1 then
        {
          Manual_Select = 0; //Reset Manual_Select flag
          Manual_Finish = 1; //Set Manual_Finish flag
        } //if (Manual_Select...
       } //else if (manual_menu...
       delay(50); //delay 50 msec
     } //while (Manual_Finish...
    } //else if (menu == 7...
    Menu(); //Call Menu() function
    delay(200); //Delay 200 msec
  } //else if (x<800...
}//void ButtonRead
```

3.1.5 **Ανάλυση των συναρτήσεων reset (), checkClipping(), FREQ_DETECTION() & FREQ_DETECTION_STOP() (Κώδικας προγραμματισμού Μέρος 9ο)**

Σε αυτή την ενότητα, πραγματοποιείται ανάλυση του κώδικα προγραμματισμού κάποιων συναρτήσεων που είναι απαραίτητες για την ορθή λειτουργία του συστήματος (Πίνακας 3.9) [127]. Πιο συγκεκριμένα, αναλύεται η συνάρτηση reset(), η οποία χρησιμοποιείτε στην ρουτίνα εξυπηρέτησης διακοπής για την οποία γίνεται αναφορά στο

επόμενο κεφάλαιο. Επιπροσθέτως, γίνεται ανάλυση της συνάρτησης checkClipping(), η οποία χρησιμοποιείτε από την συνάρτηση FREQ_DETECTION1() της οποίας η ανάλυση γίνεται σε επόμενη ενότητα. Επίσης, γίνεται ανάλυση των συναρτήσεων FREQ_DETECTION() και FREQ_DETECTION_STOP(), οι οποίες καλούνται στο μπλοκ εντολών της συνάρτησης ButtonRead() η οποία με την σειρά της, αναλύθηκε στην προηγούμενη ενότητα.

Αρχικά, η συνάρτηση reset() έχει ένα απλό μπλοκ εντολών και σκοπός της είναι να εκτελέσει κάποιες αρχικοποιήσεις. Πιο συγκεκριμένα, εκτελεί την εντολή (index = 0), η οποία έχει σκοπό τον μηδενισμό του δείκτη με όνομα (index). Έπειτα, εκτελείται η εντολή (noMatch = 0) η οποία αποσκοπεί στον μηδενισμό της μεταβλητής με όνομα (noMatch). Ύστερα, εκτελείται η τελευταία εντολή του μπλοκ εντολών της συνάρτησης reset(), η οποία είναι η (maxSlope=0) και σκοπός της είναι να μηδενίσει την τιμή της μεταβλητής με όνομα (maxSlope).

Εν συνεχεία, η συνάρτηση checkClipping(), αποτελείται και αυτή από ένα απλό μπλοκ εντολών. Πιο αναλυτικά, περιλαμβάνει μία εντολή ελέγχου if (clipping), η οποία ελέγχει αν η σημαία (flag) είναι θετική, ή αλλιώς αληθές (TRUE). Σε περίπτωση που ικανοποιείτε η συνθήκη, τότε εκτελείται το μπλοκ εντολών της συνθήκης (if). Πιο συγκεκριμένα, εντός του μπλοκ εντολών της συνθήκης (if) γίνεται μηδενισμός της σημαίας (flag) με όνομα (clipping).

Όσον αφορά την συνάρτηση FREQ_DETECTION(), το μπλοκ εντολών της παίζει ένα πολύ σημαντικό ρόλο στην ορθότητα του προγράμματος καθώς και στην λειτουργία του συστήματος. Ουσιαστικά, η συγκεκριμένη συνάρτηση, εκτελεί εντολές σε επίπεδο καταχωρητών, ώστε να γίνει καθορισμός των παραμέτρων και ενεργοποίηση του αναλογικό-ψηφιακού μετατροπέα καθώς και εκκίνηση της δειγματοληψίας του σήματος από το μικρόφωνο επαφής μέσω της αναλογικής θύρας δέκα πέντε. Πιο αναλυτικά, εντός του μπλοκ εντολών της συγκεκριμένης συνάρτησης, εκτελείται η εντολή Serial.println(“”), η οποία έχει σκοπό την εκτύπωση του μηνύματος (Into freq_detection function) στην σειριακή οθόνη, ώστε ο προγραμματιστής να εντοπίσει σε πιο σημείο βρίσκεται η εκτέλεση του προγράμματος. Έπειτα, εκτελείται η εντολή cli(), η οποία αποσκοπεί στην απενεργοποίηση των διακοπών (Interrupts). Ύστερα, γίνεται εκτέλεση των εντολών ADCSRA = 0, ADCSRB = 0 & ADMUX = 0, οι οποίες αποσκοπούν στον μηδενισμό όλων των bit των καταχωρητών ADCSRA, ADCSRB & ADMUX. Μετά τον μηδενισμό των bit των καταχωρητών, εκτελείται η εντολή ADMUX|=(1<<REFS0), η οποία θέτει με ένα το bit με όνομα (REFS0) του καταχωρητή (ADMUX). Με αυτόν τον τρόπο, καθορίζεται η τάση αναφοράς του αναλογικό-ψηφιακού μετατροπέα. Στην συνέχεια, εκτελείται η εντολή ADMUX|=(1<<ADLAR), η οποία θέτει με ένα το bit με όνομα (ADLAR) του καταχωρητή (ADMUX). Αυτό έχει ως αποτέλεσμα, τα δεδομένα που παράγονται από τον αναλογικό-ψηφιακό μετατροπέα, να αποθηκεύονται με αριστερή στοίχιση εντός του καταχωρητή (ADC). Έπειτα, εκτελούνται με την σειρά οι εντολές ADCSRB|=(1<<MUX5) και ADMUX|=(1<<MUX0)|(1<<MUX1)|(1<<MUX2) οι οποίες θέτουν με ένα το bit (MUX5) του καταχωρητή (ADCSRB) καθώς επίσης θέτουν με ένα τα bit (MUX0, MUX1 & MUX2) του καταχωρητή (ADMUX). Με αυτόν τον τρόπο ορίζεται η δειγματοληψία του σήματος να πραγματοποιείται από την αναλογική θύρα δεκαπέντε του Arduino Mega 2560. Ύστερα, εκτελείται η εντολή ADCSRA|=(1<<ADPS2)|(1<<ADPS0) η οποία θέτει με ένα τα bit (ADPS2 & ADPS0) του καταχωρητή (ADCSRA). Αυτό έχει ως αποτέλεσμα τον καθορισμό του προκλιμακωτή του ρολογιού

(prescaler) του αναλογικό-ψηφιακού μετατροπέα με την τιμή τριάντα δύο. Έτσι επιτυγχάνεται η συχνότητα δειγματοληψίας του αναλογικό-ψηφιακού μετατροπέα να έχει τιμή 16MHz/32 = 500kHz. Εν συνεχεία, πραγματοποιείται εκτέλεση της εντολής ADCSRA|=(1<<ADATE), η οποία θέτει με ένα το bit (ADATE) του καταχωρητή (ADCSRA). Με αυτό τον τρόπο, ενεργοποιείται η αυτόματη διέγερση του αναλογικό-ψηφιακού μετατροπέα όταν το σήμα διέγερσης έχει θετική κλίση. Έπειτα, εκτελείται η εντολή ADCSRA|=(1<<ADIE), η οποία θέτει την τιμή ένα στο bit (ADIE) του καταχωρητή (ADCSRA). Αυτό έχει ως αποτέλεσμα την ενεργοποίηση των διακοπών όταν ολοκληρωθεί η δειγματοληψία του σήματος από το μικρόφωνο επαφής. Ύστερα, γίνεται εκτέλεση της εντολής ADCSRA|=(1<<ADEN), η οποία θέτει με ένα το bit (ADEN) του καταχωρητή (ADCSRA). Με αυτόν τον τρόπο, πραγματοποιείτε ενεργοποίηση του αναλογικό-ψηφιακού μετατροπέα. Στην συνέχεια, εκτελείται η εντολή ADCSRA|= (1<<ADSC), η οποία θέτει με ένα το bit (ADSC) του καταχωρητή (ADCSRA). Αυτό έχει ως αποτέλεσμα την εκκίνηση της δειγματοληψίας από τον αναλογικό-ψηφιακό μετατροπέα. Το συγκεκριμένο bit παραμένει ένα κατά την διάρκεια της δειγματοληψίας και μεταβαίνει στην τιμή μηδέν όταν ολοκληρωθεί η μετατροπή. Το μπλοκ εντολών της συγκεκριμένης συνάρτησης, ολοκληρώνεται με την εκτέλεση της εντολής sei(). Ουσιαστικά, με την εκτέλεση της συγκεκριμένης εντολής πραγματοποιείται ενεργοποίηση των διακοπών (Interrupts) του συστήματος.

Τέλος, η συνάρτηση FREQ_DETECTION_STOP() αποτελείται και αυτή από ένα αρκετά απλό μπλοκ εντολών. Σκοπός της συγκεκριμένης συνάρτησης, είναι να σταματήσει την δειγματοληψία του σήματος από το μικρόφωνο επαφής απενεργοποιώντας τον αναλογικό ψηφιακό μετατροπέα. Ακόμα, με αυτόν τον τρόπο σταματούν και οι διακοπές (Interrupts) που πραγματοποιούνται κάθε φορά που ολοκληρώνεται η μετατροπή του σήματος από το μικρόφωνο επαφής. Πιο συγκεκριμένα, εντός του μπλοκ εντολών της συγκεκριμένης συνάρτησης, εκτελούνται με την σειρά οι εντολές ADCSRA = 0, ADCSRB = 0 & ADMUX = 0, οι οποίες αποσκοπούν στον μηδενισμό όλων των bit των καταχωρητών (ADCSRA, ADCSRB & ADMUX).

Πίνακας 3.9: Ο κώδικας προγραμματισμού των συναρτήσεων reset(), checkClipping(), FREQ_DETECTION() & FREQ_DETECTION_STOP() [35].

Κώδικας Προγραμματισμού (Μέρος 9)

void reset(){ // reset () function
index=0; //Reset value of index variable
noMatch=0; //Reset value of noMatch variable
maxSlope=0; //Reset value of maxSlope variable
}//void reset()
 	
void checkClipping(){ //checkClipping () function
if (clipping) //If clipping is positive, then
{
clipping = 0; //Reset value of clipping variable
}//if (clipping)

```
}//void checkClipping...

void FREQ_DETECTION() //FREQ_DETECTION Function
{
  Serial.println("Into freq_detection function"); //Print message at Serial Port
  cli(); //disable interrupts
  ADCSRA=0; //Clear ADCSRA register
  ADCSRB=0; //Clear ADCSRB register
  ADMUX=0; //Clear ADMUX register
  ADMUX|=(1<<REFS0); //Set REFS0 bit at ADMUX register. Set reference voltage
  ADMUX|=(1<<ADLAR); //Set ADLAR bit at ADMUX register. Left align the ADC value to read highest 8 bits from ADCH register only
  ADCSRB|=(1<<MUX5); //Set MUX5 bit at ADCSRB register. Analog port 15 for use
  ADMUX|=(1<<MUX0)|(1<<MUX1)|(1<<MUX2); //Set MUX0, MUX1 & MUX2 bit at ADMUX register. Analog port 15 for use
  ADCSRA|=(1<<ADPS2)|(1<<ADPS0); //Set ADPS2 & ADPS0 bit at ADCSRA register. Set ADC clock with 32 prescaler, 16MHz/32=500KHz
  ADCSRA|=(1<<ADATE); //Set ADATE bit at ADCSRA register. Enable Auto trigger
  ADCSRA|=(1<<ADIE); //Set ADIE bit at ADCSRA register. Enable interrupts when measurement complete
  ADCSRA|=(1<<ADEN); //Set ADEN bit at ADCSRA register. Enable ADC.
  ADCSRA|=(1<<ADSC); //Set ADSC bit at ADCSRA register. Start ADC measurement
  sei(); //Enable interrupts
}//void FREQ_DETECTION

void FREQ_DETECTION_STOP() //FREQ_DETECTION_STOP() function
{
  ADCSRA = 0; //Clear ADCSRA register
  ADCSRB = 0; //Clear ADCSRB register
  ADMUX = 0; //Clear ADMUX register
} //void FREQ_DETECTION_STOP()
```

3.1.6 **Ανάλυση της ρουτίνας εξυπηρέτησης διακοπής (ISR) (Κώδικας προγραμματισμού Μέρος 10ο)**

Σε αυτήν την ενότητα, πραγματοποιείται ανάλυση της ρουτίνας εξυπηρέτησης διακοπής (ISR), η οποία αποτελεί σημαντικό μέρος του κώδικα (Πίνακας 3.10) [127]. Πιο συγκεκριμένα, το μπλοκ εντολών της ρουτίνας εξυπηρέτησης διακοπής (ISR), εκτελεί τις απαραίτητες εντολές για τον υπολογισμό της περιόδου του σήματος που προέρχεται από το μικρόφωνο επαφής. Όπως αναφέρθηκε στο κεφάλαιο 2, το συγκεκριμένο κομμάτι κώδικα, προέρχεται από μία εργασία της Amanda Ghassaei για την αναγνώριση της

συχνότητας ταλάντωσης της χορδής. Πάραυτα, έχει πραγματοποιηθεί μια μικρή διασκευή του κώδικα κυρίως σε σημεία τα οποία δεν ήταν χρήσιμα για την εφαρμογή της συγκεκριμένης διπλωματικής εργασίας.

Αρχικά, η δήλωση της ρουτίνας εξυπηρέτησης διακοπής (ISR), πραγματοποιείτε με την εντολή ISR(), όπου εντός των παρενθέσεων γράφεται η εντολή (ADC_vect). Με αυτόν τον τρόπο, δηλώνεται ότι η συγκεκριμένη ρουτίνα εξυπηρέτησης διακοπής, εκτελείται μόνο όταν ολοκληρωθεί η μετατροπή του σήματος από τον αναλογικό-ψηφιακό μετατροπέα.

Στην συνέχεια, εντός του μπλοκ εντολών εκτελείται η εντολή prevData = newData, η οποία αποθηκεύει την τιμή της μεταβλητής newData στην μεταβλητή prevData. Έπειτα, εκτελείται η εντολή newData = ADCH, η οποία αποθηκεύει το αποτέλεσμα της μετατροπής από τον αναλογικό-ψηφιακό μετατροπέα στην μεταβλητή newData. Με αυτές τις δύο εντολές πραγματοποιείται αποθήκευση της καινούργιας τιμής του αναλογικό-ψηφιακού μετατροπέα, χωρίς την απόρριψη της προηγούμενης. Επίσης, ο καταχωρητής ADCH, περιλαμβάνει τα πιο σημαντικά bit του αποτελέσματος της μετατροπής. Αυτό προκύπτει εξαιτίας της αριστερής στοίχισης των δεδομένων που έχει γίνει στον καταχωρητή (ADC) σύμφωνα με τις αρχικοποιήσεις των bit εντός του μπλοκ εντολών της συνάρτησης FREQ_DETECTION().

Ύστερα, εκτελείται η εντολή ελέγχου if(prevData < 127 && newData>=127), η οποία ελέγχει αν η τιμή της μεταβλητής (prevData) είναι μικρότερη από εκατό είκοσι επτά και αν η τιμή της μεταβλητής (newData) είναι μεγαλύτερη ή ίση με εκατό είκοσι επτά. Αν ικανοποιείται αυτή η συνθήκη, τότε αυτό σημαίνει ότι το σήμα διέρχεται και ξεπερνά το σημείο τομής. Επομένως, σε αυτή την περίπτωση εκτελείται το μπλοκ εντολών της συνθήκης ελέγχου (if). Πιο αναλυτικά, εντός του μπλοκ εντολών της συνθήκης ελέγχου (if), πραγματοποιείται η εντολή newSlope = newData – prevData, η οποία αποθηκεύει την διαφορά μεταξύ των μεταβλητών (newData) και (prevData) στην μεταβλητή (newSlope). Με αυτόν τον τρόπο, υπολογίζεται η κλίση του σήματος. Έπειτα, εφαρμόζεται η τεχνική ελέγχου (if – else if – else) για τον έλεγχο της κλίσης του σήματος. Πιο συγκεκριμένα, πρώτα εκτελείται η εντολή ελέγχου if(abs(newSlope-maxSlope)<slopeTol), η οποία ελέγχει αν η απόλυτη τιμή της διαφοράς μεταξύ των μεταβλητών (newSlope) και (maxSlope) είναι μικρότερη από την τιμή της μεταβλητής (slopeTol). Με αυτόν τον τρόπο ελέγχεται αν η νέα κλίση του σήματος είναι ίση με την μέγιστη κλίση που μπορεί να έχει το σήμα. Αν ικανοποιείται η συγκεκριμένη συνθήκη, τότε εκτελείται το μπλοκ εντολών της συνθήκης ελέγχου (if). Πιο συγκεκριμένα, εκτελείται η εντολή slope[index] = newSlope, μέσω της οποίας γίνεται αποθήκευση της τιμής της μεταβλητής (newSlope) στον μονοδιάστατο πίνακα (slope), στην θέση που ορίζει ο δείκτης (index). Έπειτα, εκτελείται η εντολή timer[index] = time, μέσω της οποίας πραγματοποιείται αποθήκευση της τιμής της μεταβλητής (time) στον μονοδιάστατο πίνακα (timer) και πιο συγκεκριμένα στην θέση που ορίζει ο δείκτης (index). Ύστερα, εκτελείται η εντολή time = 0, η οποία αποσκοπεί στον μηδενισμό της μεταβλητής (time). Στην συνέχεια, εφαρμόζεται πάλι η τεχνική ελέγχου (if – else if – else) ώστε να διαπιστωθεί από το πρόγραμμα αν υπάρχουν τα απαραίτητα δεδομένα για τον υπολογισμό της περιόδου του σήματος. Πιο συγκεκριμένα, εκτελείται η εντολή if(index == 0), η οποία ελέγχει αν η τιμή του δείκτη (index) είναι ίση με το μηδέν. Αν ικανοποιείται αυτή η συνθήκη ελέγχου, τότε αυτό σημαίνει ότι έχει γίνει επανατοποθέτηση της νέας μέγιστης κλίσης. Συνεπώς, αν ικανοποιείται η συγκεκριμένη συνθήκη ελέγχου (if), τότε εκτελείται το αντίστοιχο μπλοκ

εντολών. Πιο αναλυτικά, το μπλοκ εντολών της συνθήκης ελέγχου (if), εκτελεί την εντολή noMatch = 0 η οποία αποσκοπεί στον μηδενισμό της τιμής της μεταβλητής (noMatch). Έπειτα, εκτελεί την εντολή index++ η οποία αυξάνει την τιμή του δείκτη (index) κατά μία μονάδα. Στην περίπτωση όμως που δεν ικανοποιείται η συνθήκη της εντολής if(index == 0), τότε εκτελείται η εντολή ελέγχου else if(abs(timer[0]-timer[index])<timerTol&&abs(slope[0]-newSlope)<slopeTol). Η συγκεκριμένη εντολή, ελέγχει αν η απόλυτη τιμή της διαφοράς μεταξύ των τιμών του πίνακα (timer[0]) και (timer[index]) είναι μικρότερη από την τιμή της μεταβλητής (timerTol) καθώς επίσης ελέγχει αν η απόλυτη τιμή της διαφορά μεταξύ των τιμών του πίνακα (slope[0]) και (slope[index]) είναι μικρότερη από την τιμή της μεταβλητής (slopeTol). Αν ικανοποιείται η συγκεκριμένη συνθήκη, τότε τα δεδομένα των δύο πινάκων ανταποκρίνονται στα όρια των ανοχών που έχον τεθεί. Επομένως, εντός του μπλοκ εντολών της συνθήκης (else if), εκτελείται η εντολή totalTimer = 0, η οποία αποσκοπεί στον μηδενισμό της μεταβλητής (totalTimer). Στην συνέχεια, εκτελείται ένας βρόγχος επανάληψης (for), ο οποίος στο εσωτερικό του μπλοκ εντολών του εκτελεί την εντολή totalTimer+=timer[k]. Σκοπός του συγκεκριμένου συνδυασμού εντολών είναι να αποθηκευτεί στην μεταβλητή (totalTimer) το άθροισμα όλων των τιμών του πίνακα (timer). Αφού ολοκληρωθεί η εκτέλεση του βρόγχου επανάληψης (for), τότε εκτελείται η εντολή period = totalTimer, οποία αποθηκεύει την τιμή της μεταβλητής (totalTimer) στην μεταβλητή (period). Με αυτό τον τρόπο λοιπόν υπολογίζεται η περίοδος του σήματος. Ύστερα, εκτελείται η εντολή timer[0] = timer[index], η οποία αποσκοπεί στην αποθήκευση της (timer[index]) στην θέση (timer[0]). Έπειτα, εκτελείται η εντολή slope[0] = slope[index] η οποία αποσκοπεί στην αποθήκευση της (slope[index]) στην θέση (slope[0]). Εν συνεχεία, πραγματοποιείται αρχικοποίηση του δείκτη (index) με την τιμή ένα, μέσω της εντολής index = 1. Επίσης, πραγματοποιείται αρχικοποίηση και της μεταβλητής (noMatch) με την τιμή μηδέν, μέσω της εντολής noMatch = 0. Αν όμως, δεν ικανοποιείται η συνθήκη της εντολής ελέγχου else if (abs(timer[0]-timer[index]) < timerTol && abs(slope[0]-newSlope) < slopeTol), τότε εκτελείται το μπλοκ εντολών της εντολής (else). Το γεγονός αυτό σημαίνει ότι το σήμα δεν έχει περάσει αρκετές φορές από το σημείο τομής. Επομένως, εντός του μπλοκ εντολών της συνθήκης ελέγχου (else), πραγματοποιείται αύξηση του δείκτη (index) κατά μία μονάδα μέσω της εντολής index++. Έπειτα εκτελείται η εντολή ελέγχου if (index>9) η οποία ελέγχει αν η τιμή του δείκτη (index) είναι μεγαλύτερη από εννέα. Αν ικανοποιείται αυτή η συνθήκη, τότε γίνεται κλήση της συνάρτησης reset(). Αυτό σημαίνει ότι έχουν αποθηκευτεί αρκετά δεδομένα τα οποία δεν είναι κατάλληλα για τον υπολογισμό της περιόδου του σήματος. Με αυτές τις εντολές που προαναφέρθηκαν, κλείνει το μπλοκ εντολών της συνθήκης if(abs(newSlope-maxSlope)<slopeTol). Στην περίπτωση όμως που δεν ικανοποιείται η εντολή ελέγχου if(abs(newSlope-maxSlope)<slopeTol), τότε εκτελείται η εντολή ελέγχου else if (newSlope>maxSlope) η οποία ελέγχει αν η μεταβλητή (newSlope) είναι μεγαλύτερη από την μεταβλητή (maxSlope). Αν ικανοποιείται αυτή η συνθήκη, τότε εκτελείται το μπλοκ εντολών της συνθήκης ελέγχου (else if) γεγονός που σημαίνει ότι η νέα κλίση που έχει εντοπιστεί είναι μεγαλύτερη από την μέγιστη κλίση που υπήρχε μέχρι τώρα. Επομένως, εντός του μπλοκ εντολών της συνθήκης (else if), εκτελείται η εντολή maxSlope = newSlope, η οποία αποθηκεύει την τιμή της μεταβλητής (newSlope) στην μεταβλητή (maxSlope). Έπειτα, εκτελούνται με την σειρά οι εντολές time = 0, noMatch = 0 και index = 0 οι οποίες μηδενίζουν τις τιμές των μεταβλητών (time) και (noMatch) καθώς και του δείκτη (index). Αν όμως δεν ικανοποιείται η εντολή ελέγχου else if (newSlope>maxSlope), τότε εκτελείται το μπλοκ

εντολών της εντολής (else). Αυτό σημαίνει ότι η κλίση του σήματος δεν είναι τόσο απότομη όσο θα έπρεπε. Επομένως, εντός του μπλοκ εντολών της εντολής (else), πραγματοποιείται αύξηση της μεταβλητής (noMatch) μέσω της εντολής noMatch++. Έπειτα, εκτελείται η εντολή ελέγχου if(noMatch>9), η οποία ελέγχει αν η τιμή της μεταβλητής (noMatch) είναι μεγαλύτερη από εννέα. Αν ικανοποιείται η συνθήκη ελέγχου (if), τότε γίνεται κλήση της συνάρτησης reset(). Σε αυτό το σημείο ολοκληρώνεται το μπλοκ εντολών της εντολής ελέγχου if(prevData < 127 && newData>=127).

Αφού ολοκληρωθεί η εκτέλεση του μπλοκ εντολών της συνθήκης (if), τότε εκτελείται η εντολή ελέγχου if(newData == 0 || newData == 1023), η οποία ελέγχει αν η τιμή της μεταβλητής (newData) είναι ίση με το μηδέν ή ίση με χίλια είκοσι τρία. Αν ικανοποιείται η συγκεκριμένη συνθήκη, τότε αυτό σημαίνει ότι το σήμα βγαίνει εκτός ορίων και κόβεται. Συνεπώς, σε αυτή την περίπτωση εκτελείτε το μπλοκ εντολών της συνθήκης (if), το οποίο θέτει την τιμή ένα στην σημαία (flag) με όνομα (clipping), μέσω της εντολής clipping = 1 και εμφανίζει το μήνυμα (clipping) στην σειριακή οθόνη με την χρήση της εντολής Serial.println(“ “). Μετά την ολοκλήρωση του μπλοκ εντολών (if), πραγματοποιείται αύξηση της μεταβλητής (time) κατά μία μονάδα με την χρήση της εντολής time++. Επίσης, επιτυγχάνεται η αύξηση της τιμής της μεταβλητής (ampTimer) κατά μία μονάδα με την εκτέλεση της εντολής ampTimer++.

Στην συνέχεια, εκτελείται μία ακόμα εντολή ελέγχου if(abs(127-ADCH) > maxAmp), η οποία ελέγχει αν η απόλυτη τιμή της διαφοράς μεταξύ του εκατόν είκοσι επτά και του καταχωρητή (ADCH) είναι μεγαλύτερη από την μεταβλητή (maxAmp). Στην περίπτωση που ικανοποιείται η συγκεκριμένη συνθήκη, τότε εκτελείται το μπλοκ εντολών της συνθήκης (if). Πιο συγκεκριμένα, εκτελείται η εντολή maxAmp=abs(127-ADCH), η οποία αποθηκεύει την απόλυτη τιμή της διαφοράς μεταξύ του εκατόν είκοσι επτά και της τιμής του καταχωρητή (ADCH) στην μεταβλητή (maxAmp).

Τέλος, το μπλοκ εντολών της ρουτίνας εξυπηρέτησης διακοπής (ISR), ολοκληρώνεται με την εκτέλεση της εντολής ελέγχου if (ampTimer == 1000), η οποία ελέγχει αν η μεταβλητή (ampTimer) είναι ίση με χίλια. Στην περίπτωση που ικανοποιείται η συγκεκριμένη συνθήκη ελέγχου, τότε πραγματοποιείται εκτέλεση του μπλοκ εντολών της συγκεκριμένης συνθήκης (if). Πιο συγκεκριμένα, πραγματοποιείται μηδενισμός της μεταβλητής (ampTimer), με την χρήση της εντολής ampTimer = 0. Επίσης, πραγματοποιείται αποθήκευση της τιμής της μεταβλητής (maxAmp) στην μεταβλητή (checkMaxAmp), με την εκτέλεση της εντολής checkMaxAmp = maxAmp. Ολοκληρώνοντας, πραγματοποιείται μηδενισμός της μεταβλητής (maxAmp) με την χρήση της εντολής maxAmp = 0.

Πίνακας 3.10: Ο κώδικας προγραμματισμού της ρουτίνας εξυπηρέτησης διακοπής (ISR) [36].

Κώδικας Προγραμματισμού (Μέρος 10ο)

ISR(ADC_vect){ //interrupt routine that runs when a new ADC value is ready
prevData = newData; //Store previous value
newData = ADCH; //Read value from ADCH register
if(prevData < 127 && newData>=127){ //If prevData is smaller than 127 and newData is bigger than or equal with 127, then (increasing and crossing midpoint)

```
  newSlope = newData - prevData; //Store the deference between newData minus prevData to newSlope variable (calculate slope)
  if(abs(newSlope-maxSlope)<slopeTol){ //If the absolute deference between newData minus maxSlope is smaller than slopeTol, then (if slopes are equal)
    slope[index] = newSlope; //Store newSlope value at slope Array in position that index show
    timer[index] = time; //Store time value at timer Array in position that index show
    time = 0; //Reset time
    if(index == 0){ //if index is equal with 0, then (new max slope just reset)
      noMatch = 0; //Store 0 value at noMatch variable
      index++; // Increase index by 1
    }//if(index ==0);
    else      if(abs(timer[0]-timer[index])<timerTol&&abs(slope[0]-newSlope)<slopeTol){ //Else if the absolute deference between timer[0] minus timer [index] is smaller than timerTol and the absolute diference between slope [0] minus new slope is smaller than slopeTol, then
      totalTimer = 0; //Reset totalTime variable
      for (byte k=0;k<index;k++){ //For k is equal with 0, k smaller than index and k increase by 1 every time, then
          totalTimer+=timer[k]; //Store at totalTimer variable the sum of timer array
      }//for
      period = totalTimer; //Store totalTimer value at period variable (set period)
      timer[0] = timer[index]; //Store timer[index] value at timer[0]
      slope[0] = slope[index]; //Store slope[index] value at slope[0]
      index = 1; //Store number 1 at index variable
      noMatch = 0; //Reset noMatch value
    }//else if (abs(timer[0]...
    else { //Else (Crossing midpoint but not match)
      index++; //Increase index by 1
      if (index>9){ //If index is bigger than 9, then
        reset(); //Call reset() function
      }//if
    }//else
  }//if(abs(newSlope...
  else if (newSlope>maxSlope){ //Else if newSlope variable is bigger than maxSlope, then (newSlope is much larger than maxSlope)
    maxSlope = newSlope; //Store newSlope value at maxSlope variable
    time = 0; //Reset time
    noMatch = 0; //Reset noMatch value
    index = 0; //Reset index value
```

```
   }//else if(newSlope...
   else{ //Else (slope no steep enough)
     noMatch++; //Increase noMatch by 1
     if(noMatch>9){ //If noMatch value is bigger than 9, then
       reset(); //Call reset() function
     }//if(noMatch>9)...
   }//else
  }//if(prevData...
  if(newData == 0 || newData == 1023){ //If newData is equal to 0 or newData is equal to 1023, then (If clipping)
   clipping = 1; //Store number 1 at clipping variable
   Serial.println("clipping"); //Print a message at serial port
  }//if(newData==0||newData==1023)...
  time++; //Increase time by 1
  ampTimer++; //Increase ampTimer by 1
  if(abs(127-ADCH) > maxAmp){ //If the absolute difference between 127 minus ADCH is bigger than maxAmp variable, then
   maxAmp=abs(127-ADCH); //Store at maxAmp variable the absolute difference between 127 minus ADCH
  }//if(abs(127...
  if (ampTimer == 1000){ //If ampTimer is equal to 1000, then
   ampTimer = 0; //Reset ampTimer
   checkMaxAmp = maxAmp; //Store maxAmp variable at checkMaxAmp variable
   maxAmp = 0; //Reset maxAmp variable
  }//if(ampTimer...
} //ISR(ADC_vect)
```

3.1.7 **Ανάλυση της συνάρτησης FREQ_DETECTION1() (Κώδικας προγραμματισμού Μέρος 11ο)**

Σε αυτήν την ενότητα αναλύεται η συνάρτηση FREQ_DETECTION1(), η οποία αποτελεί αναπόσπαστο μέρος της ορθής λειτουργίας του συστήματος (Πίνακας 3.11). Ουσιαστικός σκοπός της συγκεκριμένης συνάρτησης, είναι να υπολογίσει την συχνότητα ταλάντωσης της χορδής και να εντοπίσει την νότα με την οποία αντιστοιχεί. Αυτό επιτυγχάνεται μέσα από μία σειρά εντολών που βρίσκονται εντός του μπλοκ εντολών της συνάρτησης FREQ_DETECTION1(). Επίσης, η κλήση της συγκεκριμένης συνάρτησης πραγματοποιείται στο μπλοκ εντολών της συνάρτησης ButtonRead() και ιδιαίτερα στα σημεία όπου πρέπει να πραγματοποιηθεί ανίχνευση της θεμελιώδης συχνότητας ταλάντωσης της διεγερμένης χορδής.

Αρχικά, το μπλοκ εντολών της συνάρτησης FREQ_DETECTION1() ξεκινά με την κλήση της συνάρτησης checkClipping() η οποία αναλύθηκε σε προηγούμενη ενότητα και αποσκοπεί στον μηδενισμό της σημαίας (flag) με όνομα (clipping). Ύστερα εκτελείται η

εντολή ελέγχου if (checkMaxAmp > ampThreshold), η οποία ελέγχει αν η τιμή της μεταβλητής (checkMaxAmp) είναι μεγαλύτερη από την τιμή της μεταβλητής (ampThreshold). Στην περίπτωση που ικανοποιείται η συγκεκριμένη συνθήκη (if), τότε εκτελείται το αντίστοιχο μπλοκ εντολών της. Πιο συγκεκριμένα, εντός του μπλοκ εντολών της συνθήκης (if), πραγματοποιείται ο υπολογισμός της συχνότητας ταλάντωσης της διεγερμένης χορδής με την χρήση της εντολής frequency = 38462/float(period). Για την εύρεση της συχνότητας ταλάντωσης της διεγερμένης χορδής, θα πρέπει να υπολογιστεί το πηλίκο του ρυθμού της χρονοδιακοπής (38462) και της περιόδου του σήματος ταλάντωσης (period). Επειδή όμως, η μεταβλητή (period) είναι τύπου μη προσημασμένου ακεραίου (unsigned integer) και το πηλίκο της διαίρεσης θα πρέπει να αποθηκευτεί στην μεταβλητή (frequency) που είναι τύπου δεκαδικού αριθμού (float), θα πρέπει να γίνεται χρήση της τεχνικής της μάσκας (mask). Πιο συγκεκριμένα, η τεχνική της μάσκας πραγματοποιείτε στην συγκεκριμένη περίπτωση με την εντολή (float(period)), όπου η μάσκα είναι η εντολή float() και εντός των παρενθέσεων γράφεται η μεταβλητή (period). Με αυτόν τον τρόπο, η μεταβλητή (period), γίνεται για μία στιγμή και μόνο για αυτήν την πράξη μεταβλητή τύπου κινητής υποδιαστολής (float). Μετά την ολοκλήρωση του υπολογισμού της θεμελιώδης συχνότητας ταλάντωσης της διεγερμένης χορδής, εκτελούνται με την σειρά οι εντολές Serial.print(frequency) και Serial.println("Hz") ώστε να εμφανιστεί στην σειριακή οθόνη η τιμή της μεταβλητής (frequency).

Ύστερα, εφαρμόζεται η τεχνική (if – else if – else) ώστε να εντοπιστεί η νότα στην οποία αντιστοιχεί η συγκεκριμένη συχνότητα ταλάντωσης της χορδής. Πιο συγκεκριμένα, εκτελείται η εντολή if (frequency > 63.58 && frequency < 67.35), η οποία ελέγχει αν η τιμή της μεταβλητής (frequency) είναι μεγαλύτερη από 63.58Hz και μικρότερη από 67.35Hz. Αν ικανοποιείτε η συγκεκριμένη συνθήκη, τότε εκτελούνται με κατάλληλη σειρά οι εντολές lcd.setCursor() και lcd.print(), ώστε να εμφανιστεί στην LCD οθόνη το μήνυμα (C) το οποίο δηλώνει στον χρήστη την νότα στην οποία αντιστοιχεί η συγκεκριμένη συχνότητα.

Αν όμως δεν ικανοποιείται η συνθήκη ελέγχου (if), τότε εκτελείται η επόμενη εντολή ελέγχου else if(frequency > 67.36 && frequency < 71.36), η οποία ελέγχει αν η τιμή της μεταβλητής (frequency) είναι μεγαλύτερη από 67.36Hz και μικρότερη από 71.36Hz. Επομένως, στην περίπτωση που ικανοποιείται η συγκεκριμένη συνθήκη πραγματοποιείται εκτέλεση των εντολών lcd.setCursor() και lcd.print() με την κατάλληλη σειρά, με αποτέλεσμα να εμφανιστεί στην LCD οθόνη το μήνυμα (C#). Με αυτόν τον τρόπο, ο χρήστης αντιλαμβάνεται σε ποια νότα αντιστοιχεί η συγκεκριμένη συχνότητα ταλάντωσης της χορδής.

Στην αντίθετη περίπτωση, αν δεν ικανοποιείται ούτε η προηγούμενη συνθήκη ελέγχου (else if), τότε εκτελείται η εντολή else if (frequency > 71.37 && frequency < 75.60), η οποία ελέγχει αν η τιμή της μεταβλητής (frequency) είναι μεγαλύτερη από 71.37Hz και μικρότερη από 75.60Hz. Συνεπώς, αν ικανοποιείτε η συγκεκριμένη συνθήκη, τότε εκτελούνται με κατάλληλη σειρά οι εντολές lcd.setCursor() και lcd.print(), ώστε να εμφανιστεί στην LCD οθόνη το μήνυμα (D). Έτσι, δηλώνεται στον χρήστη η νότα στην οποία αντιστοιχεί η συγκεκριμένη συχνότητα ταλάντωσης της διεγερμένης χορδής.

Στην περίπτωση όμως που δεν ικανοποιείται ούτε αυτή η συνθήκη ελέγχου (else if), τότε εκτελείται η επόμενη εντολή ελέγχου else if (frequency > 75.61 && frequency < 80.09), η οποία ελέγχει αν η τιμή της μεταβλητής (frequency) είναι μεγαλύτερη από 75.61Hz και μικρότερη από 80.09Hz. Επομένως, εφόσον ικανοποιείται η συγκεκριμένη

συνθήκη, τότε πραγματοποιείται εκτέλεση των εντολών lcd.setCursor() και lcd.print() με την κατάλληλη σειρά, με αποτέλεσμα να εμφανιστεί στην LCD οθόνη το μήνυμα (D#). Με αυτόν τον τρόπο, ο χρήστης αντιλαμβάνεται σε ποια νότα αντιστοιχεί η συγκεκριμένη συχνότητα ταλάντωσης της χορδής.

Αν όμως, δεν ικανοποιείται ούτε η συνθήκη ελέγχου (else if) που αναφέρθηκε προηγουμένως, τότε πραγματοποιείται εκτέλεση της επόμενης εντολής ελέγχου else if (frequency > 80.10 && frequency < 84.86), η οποία ελέγχει αν η τιμή της μεταβλητής (frequency) είναι μεγαλύτερη από 80.10Hz και μικρότερη από 84.86Hz. Συνεπώς, αν ικανοποιείτε η συγκεκριμένη συνθήκη, τότε εκτελούνται με κατάλληλη σειρά οι εντολές lcd.setCursor() και lcd.print(), ώστε να εμφανιστεί στην LCD οθόνη το μήνυμα (E). Έτσι, δηλώνεται στον χρήστη η νότα στην οποία αντιστοιχεί η συγκεκριμένη συχνότητα ταλάντωσης της διεγερμένης χορδής.

Σε αντίθετη περίπτωση, αν δεν ικανοποιείται ούτε η προηγούμενη συνθήκη ελέγχου (else if), τότε εκτελείται η εντολή else if(frequency > 84.87 && frequency < 89.90), η οποία ελέγχει αν η τιμή της μεταβλητής (frequency) είναι μεγαλύτερη από 84.87Hz και μικρότερη από 89.90Hz. Συνεπώς, αν ικανοποιείτε η συγκεκριμένη συνθήκη, τότε εκτελούνται με κατάλληλη σειρά οι εντολές lcd.setCursor() και lcd.print(), ώστε να εμφανιστεί στην LCD οθόνη το μήνυμα (F). Με αυτόν τον τρόπο, δηλώνεται στον χρήστη η νότα στην οποία αντιστοιχεί η συγκεκριμένη συχνότητα ταλάντωσης της χορδής που έχει διεγερθεί.

Στην περίπτωση όμως που δεν ικανοποιείται ούτε αυτή η συνθήκη ελέγχου (else if), τότε εκτελείται η επόμενη εντολή ελέγχου else if (frequency > 89.91 && frequency < 95.25), η οποία ελέγχει αν η τιμή της μεταβλητής (frequency) είναι μεγαλύτερη από 89.91Hz και μικρότερη από 95.25Hz. Επομένως, εφόσον ικανοποιείται η συγκεκριμένη συνθήκη, τότε πραγματοποιείται εκτέλεση των εντολών lcd.setCursor() και lcd.print() με την κατάλληλη σειρά, με αποτέλεσμα να εμφανιστεί στην LCD οθόνη το μήνυμα (F#). Με αυτόν τον τρόπο, ο χρήστης αντιλαμβάνεται σε ποια νότα αντιστοιχεί η συγκεκριμένη συχνότητα ταλάντωσης της χορδής.

Αν όμως, δεν ικανοποιείται ούτε η συνθήκη ελέγχου (else if) που αναφέρθηκε προηγουμένως, τότε πραγματοποιείται εκτέλεση της επόμενης εντολής ελέγχου else if (frequency > 95.26 && frequency < 100.91), η οποία ελέγχει αν η τιμή της μεταβλητής (frequency) είναι μεγαλύτερη από 95.26Hz και μικρότερη από 100.91Hz. Συνεπώς, αν ικανοποιείτε η συγκεκριμένη συνθήκη, τότε εκτελούνται με κατάλληλη σειρά οι εντολές lcd.setCursor() και lcd.print(), ώστε να εμφανιστεί στην LCD οθόνη το μήνυμα (G). Έτσι, δηλώνεται στον χρήστη η νότα στην οποία αντιστοιχεί η συγκεκριμένη συχνότητα ταλάντωσης της διεγερμένης χορδής.

Σε αντίθετη περίπτωση, αν δεν ικανοποιείται ούτε η προηγούμενη συνθήκη ελέγχου (else if), τότε εκτελείται η εντολή else if (frequency > 100.92 && frequency < 106.91), η οποία ελέγχει αν η τιμή της μεταβλητής (frequency) είναι μεγαλύτερη από 100.92Hz και μικρότερη από 106.91Hz. Συνεπώς, αν ικανοποιείτε η συγκεκριμένη συνθήκη, τότε εκτελούνται με κατάλληλη σειρά οι εντολές lcd.setCursor() και lcd.print(), ώστε να εμφανιστεί στην LCD οθόνη το μήνυμα (G#). Με αυτόν τον τρόπο, δηλώνεται στον χρήστη η νότα στην οποία αντιστοιχεί η συγκεκριμένη συχνότητα ταλάντωσης της χορδής που έχει διεγερθεί.

Στην περίπτωση όμως που δεν ικανοποιείται ούτε αυτή η συνθήκη ελέγχου (else if), τότε εκτελείται η επόμενη εντολή ελέγχου else if (frequency > 106.92 && frequency <

113.27), η οποία ελέγχει αν η τιμή της μεταβλητής (frequency) είναι μεγαλύτερη από 106.92Hz και μικρότερη από 113.27Hz. Επομένως, εφόσον ικανοποιείται η συγκεκριμένη συνθήκη, τότε πραγματοποιείται εκτέλεση των εντολών lcd.setCursor() και lcd.print() με την κατάλληλη σειρά, με αποτέλεσμα να εμφανιστεί στην LCD οθόνη το μήνυμα (A). Με αυτόν τον τρόπο, ο χρήστης αντιλαμβάνεται σε ποια νότα αντιστοιχεί η συγκεκριμένη συχνότητα ταλάντωσης της χορδής.

Αν όμως, δεν ικανοποιείται ούτε η συνθήκη ελέγχου (else if) που αναφέρθηκε προηγουμένως, τότε πραγματοποιείται εκτέλεση της επόμενης εντολής ελέγχου else if (frequency > 113.28 && frequency < 120), η οποία ελέγχει αν η τιμή της μεταβλητής (frequency) είναι μεγαλύτερη από 113.28Hz και μικρότερη από 120Hz. Συνεπώς, αν ικανοποιείτε η συγκεκριμένη συνθήκη, τότε εκτελούνται με κατάλληλη σειρά οι εντολές lcd.setCursor() και lcd.print(), ώστε να εμφανιστεί στην LCD οθόνη το μήνυμα (A#). Έτσι, δηλώνεται στον χρήστη η νότα στην οποία αντιστοιχεί η συγκεκριμένη συχνότητα ταλάντωσης της διεγερμένης χορδής.

Σε αντίθετη περίπτωση, αν δεν ικανοποιείται ούτε η προηγούμενη συνθήκη ελέγχου (else if), τότε εκτελείται η εντολή else if (frequency > 120.01 && frequency < 127.14), η οποία ελέγχει αν η τιμή της μεταβλητής (frequency) είναι μεγαλύτερη από 120.01Hz και μικρότερη από 127.14Hz. Συνεπώς, αν ικανοποιείτε η συγκεκριμένη συνθήκη, τότε εκτελούνται με κατάλληλη σειρά οι εντολές lcd.setCursor() και lcd.print(), ώστε να εμφανιστεί στην LCD οθόνη το μήνυμα (B). Με αυτόν τον τρόπο, δηλώνεται στον χρήστη η νότα στην οποία αντιστοιχεί η συγκεκριμένη συχνότητα ταλάντωσης της χορδής που έχει διεγερθεί.

Στην περίπτωση όμως που δεν ικανοποιείται ούτε αυτή η συνθήκη ελέγχου (else if), τότε εκτελείται η επόμενη εντολή ελέγχου else if (frequency > 127.15 && frequency < 134.81), η οποία ελέγχει αν η τιμή της μεταβλητής (frequency) είναι μεγαλύτερη από 127.15Hz και μικρότερη από 134.81Hz. Επομένως, εφόσον ικανοποιείται η συγκεκριμένη συνθήκη, τότε πραγματοποιείται εκτέλεση των εντολών lcd.setCursor() και lcd.print() με την κατάλληλη σειρά, με αποτέλεσμα να εμφανιστεί στην LCD οθόνη το μήνυμα (C). Με αυτόν τον τρόπο, ο χρήστης αντιλαμβάνεται σε ποια νότα αντιστοιχεί η συγκεκριμένη συχνότητα ταλάντωσης της χορδής.

Αν όμως, δεν ικανοποιείται ούτε η συνθήκη ελέγχου (else if) που αναφέρθηκε προηγουμένως, τότε πραγματοποιείται εκτέλεση της επόμενης εντολής ελέγχου else if (frequency > 134.82 && frequency < 142.71), η οποία ελέγχει αν η τιμή της μεταβλητής (frequency) είναι μεγαλύτερη από 134.82Hz και μικρότερη από 142.71Hz. Συνεπώς, αν ικανοποιείτε η συγκεκριμένη συνθήκη, τότε εκτελούνται με κατάλληλη σειρά οι εντολές lcd.setCursor() και lcd.print(), ώστε να εμφανιστεί στην LCD οθόνη το μήνυμα (C#). Έτσι, δηλώνεται στον χρήστη η νότα στην οποία αντιστοιχεί η συγκεκριμένη συχνότητα ταλάντωσης της διεγερμένης χορδής.

Σε αντίθετη περίπτωση, αν δεν ικανοποιείται ούτε η προηγούμενη συνθήκη ελέγχου (else if), τότε εκτελείται η εντολή else if (frequency > 142.72 && frequency < 151.19), η οποία ελέγχει αν η τιμή της μεταβλητής (frequency) είναι μεγαλύτερη από 142.72Hz και μικρότερη από 151.19Hz. Συνεπώς, αν ικανοποιείτε η συγκεκριμένη συνθήκη, τότε εκτελούνται με κατάλληλη σειρά οι εντολές lcd.setCursor() και lcd.print(), ώστε να εμφανιστεί στην LCD οθόνη το μήνυμα (D). Με αυτόν τον τρόπο, δηλώνεται στον χρήστη η νότα στην οποία αντιστοιχεί η συγκεκριμένη συχνότητα ταλάντωσης της χορδής που έχει διεγερθεί.

Στην περίπτωση όμως που δεν ικανοποιείται ούτε αυτή η συνθήκη ελέγχου (else if), τότε εκτελείται η επόμενη εντολή ελέγχου else if (frequency > 151.20 && frequency < 160.18), η οποία ελέγχει αν η τιμή της μεταβλητής (frequency) είναι μεγαλύτερη από 151.20Hz και μικρότερη από 160.18Hz. Επομένως, εφόσον ικανοποιείται η συγκεκριμένη συνθήκη, τότε πραγματοποιείται εκτέλεση των εντολών lcd.setCursor() και lcd.print() με την κατάλληλη σειρά, με αποτέλεσμα να εμφανιστεί στην LCD οθόνη το μήνυμα (D#). Με αυτόν τον τρόπο, ο χρήστης αντιλαμβάνεται σε ποια νότα αντιστοιχεί η συγκεκριμένη συχνότητα ταλάντωσης της χορδής.

Αν όμως, δεν ικανοποιείται ούτε η συνθήκη ελέγχου (else if) που αναφέρθηκε προηγουμένως, τότε πραγματοποιείται εκτέλεση της επόμενης εντολής ελέγχου else if (frequency > 160.19 && frequency < 169.71), η οποία ελέγχει αν η τιμή της μεταβλητής (frequency) είναι μεγαλύτερη από 160.19Hz και μικρότερη από 169.71Hz. Συνεπώς, αν ικανοποιείτε η συγκεκριμένη συνθήκη, τότε εκτελούνται με κατάλληλη σειρά οι εντολές lcd.setCursor() και lcd.print(), ώστε να εμφανιστεί στην LCD οθόνη το μήνυμα (E). Έτσι, δηλώνεται στον χρήστη η νότα στην οποία αντιστοιχεί η συγκεκριμένη συχνότητα ταλάντωσης της διεγερμένης χορδής.

Σε αντίθετη περίπτωση, αν δεν ικανοποιείται ούτε η προηγούμενη συνθήκη ελέγχου (else if), τότε εκτελείται η εντολή else if (frequency > 169.72 && frequency < 179.80), η οποία ελέγχει αν η τιμή της μεταβλητής (frequency) είναι μεγαλύτερη από 169.72Hz και μικρότερη από 179.80Hz. Συνεπώς, αν ικανοποιείτε η συγκεκριμένη συνθήκη, τότε εκτελούνται με κατάλληλη σειρά οι εντολές lcd.setCursor() και lcd.print(), ώστε να εμφανιστεί στην LCD οθόνη το μήνυμα (F). Με αυτόν τον τρόπο, δηλώνεται στον χρήστη η νότα στην οποία αντιστοιχεί η συγκεκριμένη συχνότητα ταλάντωσης της χορδής που έχει διεγερθεί.

Στην περίπτωση όμως που δεν ικανοποιείται ούτε αυτή η συνθήκη ελέγχου (else if), τότε εκτελείται η επόμενη εντολή ελέγχου else if (frequency > 179.81 && frequency < 190.50), η οποία ελέγχει αν η τιμή της μεταβλητής (frequency) είναι μεγαλύτερη από 179.81Hz και μικρότερη από 190.50Hz. Επομένως, εφόσον ικανοποιείται η συγκεκριμένη συνθήκη, τότε πραγματοποιείται εκτέλεση των εντολών lcd.setCursor() και lcd.print() με την κατάλληλη σειρά, με αποτέλεσμα να εμφανιστεί στην LCD οθόνη το μήνυμα (F#). Με αυτόν τον τρόπο, ο χρήστης αντιλαμβάνεται σε ποια νότα αντιστοιχεί η συγκεκριμένη συχνότητα ταλάντωσης της χορδής.

Αν όμως, δεν ικανοποιείται ούτε η συνθήκη ελέγχου (else if) που αναφέρθηκε προηγουμένως, τότε πραγματοποιείται εκτέλεση της επόμενης εντολής ελέγχου else if (frequency > 190.51 && frequency < 201.82), η οποία ελέγχει αν η τιμή της μεταβλητής (frequency) είναι μεγαλύτερη από 190.51Hz και μικρότερη από 201.82Hz. Συνεπώς, αν ικανοποιείτε η συγκεκριμένη συνθήκη, τότε εκτελούνται με κατάλληλη σειρά οι εντολές lcd.setCursor() και lcd.print(), ώστε να εμφανιστεί στην LCD οθόνη το μήνυμα (G). Έτσι, δηλώνεται στον χρήστη η νότα στην οποία αντιστοιχεί η συγκεκριμένη συχνότητα ταλάντωσης της διεγερμένης χορδής.

Σε αντίθετη περίπτωση, αν δεν ικανοποιείται ούτε η προηγούμενη συνθήκη ελέγχου (else if), τότε εκτελείται η εντολή else if (frequency > 201.83 && frequency < 213.82), η οποία ελέγχει αν η τιμή της μεταβλητής (frequency) είναι μεγαλύτερη από 201.83Hz και μικρότερη από 213.82Hz. Συνεπώς, αν ικανοποιείτε η συγκεκριμένη συνθήκη, τότε εκτελούνται με κατάλληλη σειρά οι εντολές lcd.setCursor() και lcd.print(), ώστε να εμφανιστεί στην LCD οθόνη το μήνυμα (G#). Με αυτόν τον τρόπο, δηλώνεται στον

χρήστη η νότα στην οποία αντιστοιχεί η συγκεκριμένη συχνότητα ταλάντωσης της χορδής που έχει διεγερθεί.

Στην περίπτωση όμως που δεν ικανοποιείται ούτε αυτή η συνθήκη ελέγχου (else if), τότε εκτελείται η επόμενη εντολή ελέγχου else if (frequency > 213.83 && frequency < 226.54), η οποία ελέγχει αν η τιμή της μεταβλητής (frequency) είναι μεγαλύτερη από 213.83Hz και μικρότερη από 226.54Hz. Επομένως, εφόσον ικανοποιείται η συγκεκριμένη συνθήκη, τότε πραγματοποιείται εκτέλεση των εντολών lcd.setCursor() και lcd.print() με την κατάλληλη σειρά, με αποτέλεσμα να εμφανιστεί στην LCD οθόνη το μήνυμα (A). Με αυτόν τον τρόπο, ο χρήστης αντιλαμβάνεται σε ποια νότα αντιστοιχεί η συγκεκριμένη συχνότητα ταλάντωσης της χορδής.

Αν όμως, δεν ικανοποιείται ούτε η συνθήκη ελέγχου (else if) που αναφέρθηκε προηγουμένως, τότε πραγματοποιείται εκτέλεση της επόμενης εντολής ελέγχου else if (frequency > 226.55 && frequency < 240.01), η οποία ελέγχει αν η τιμή της μεταβλητής (frequency) είναι μεγαλύτερη από 226.55Hz και μικρότερη από 240.01Hz. Συνεπώς, αν ικανοποιείτε η συγκεκριμένη συνθήκη, τότε εκτελούνται με κατάλληλη σειρά οι εντολές lcd.setCursor() και lcd.print(), ώστε να εμφανιστεί στην LCD οθόνη το μήνυμα (A#). Έτσι, δηλώνεται στον χρήστη η νότα στην οποία αντιστοιχεί η συγκεκριμένη συχνότητα ταλάντωσης της διεγερμένης χορδής.

Σε αντίθετη περίπτωση, αν δεν ικανοποιείται ούτε η προηγούμενη συνθήκη ελέγχου (else if), τότε εκτελείται η εντολή else if (frequency > 240.02 && frequency < 254.28), η οποία ελέγχει αν η τιμή της μεταβλητής (frequency) είναι μεγαλύτερη από 240.02Hz και μικρότερη από 254.28Hz. Συνεπώς, αν ικανοποιείτε η συγκεκριμένη συνθήκη, τότε εκτελούνται με κατάλληλη σειρά οι εντολές lcd.setCursor() και lcd.print(), ώστε να εμφανιστεί στην LCD οθόνη το μήνυμα (B). Με αυτόν τον τρόπο, δηλώνεται στον χρήστη η νότα στην οποία αντιστοιχεί η συγκεκριμένη συχνότητα ταλάντωσης της χορδής που έχει διεγερθεί.

Στην περίπτωση όμως που δεν ικανοποιείται ούτε αυτή η συνθήκη ελέγχου (else if), τότε εκτελείται η επόμενη εντολή ελέγχου else if (frequency > 254.29 && frequency < 269.40), η οποία ελέγχει αν η τιμή της μεταβλητής (frequency) είναι μεγαλύτερη από 254.29Hz και μικρότερη από 269.40Hz. Επομένως, εφόσον ικανοποιείται η συγκεκριμένη συνθήκη, τότε πραγματοποιείται εκτέλεση των εντολών lcd.setCursor() και lcd.print() με την κατάλληλη σειρά, με αποτέλεσμα να εμφανιστεί στην LCD οθόνη το μήνυμα (C). Με αυτόν τον τρόπο, ο χρήστης αντιλαμβάνεται σε ποια νότα αντιστοιχεί η συγκεκριμένη συχνότητα ταλάντωσης της χορδής.

Αν όμως, δεν ικανοποιείται ούτε η συνθήκη ελέγχου (else if) που αναφέρθηκε προηγουμένως, τότε πραγματοποιείται εκτέλεση της επόμενης εντολής ελέγχου else if (frequency > 269.41 && frequency < 285.42), η οποία ελέγχει αν η τιμή της μεταβλητής (frequency) είναι μεγαλύτερη από 269.41Hz και μικρότερη από 285.42Hz. Συνεπώς, αν ικανοποιείτε η συγκεκριμένη συνθήκη, τότε εκτελούνται με κατάλληλη σειρά οι εντολές lcd.setCursor() και lcd.print(), ώστε να εμφανιστεί στην LCD οθόνη το μήνυμα (C#). Έτσι, δηλώνεται στον χρήστη η νότα στην οποία αντιστοιχεί η συγκεκριμένη συχνότητα ταλάντωσης της διεγερμένης χορδής.

Σε αντίθετη περίπτωση, αν δεν ικανοποιείται ούτε η προηγούμενη συνθήκη ελέγχου (else if), τότε εκτελείται η εντολή else if (frequency > 285.43 && frequency < 302.39), η οποία ελέγχει αν η τιμή της μεταβλητής (frequency) είναι μεγαλύτερη από 285.43Hz και μικρότερη από 302.39Hz. Συνεπώς, αν ικανοποιείτε η συγκεκριμένη συνθήκη, τότε

εκτελούνται με κατάλληλη σειρά οι εντολές lcd.setCursor() και lcd.print(), ώστε να εμφανιστεί στην LCD οθόνη το μήνυμα (D). Με αυτόν τον τρόπο, δηλώνεται στον χρήστη η νότα στην οποία αντιστοιχεί η συγκεκριμένη συχνότητα ταλάντωσης της χορδής που έχει διεγερθεί.

Στην περίπτωση όμως που δεν ικανοποιείται ούτε αυτή η συνθήκη ελέγχου (else if), τότε εκτελείται η επόμενη εντολή ελέγχου else if (frequency > 302.40 && frequency < 320.38), η οποία ελέγχει αν η τιμή της μεταβλητής (frequency) είναι μεγαλύτερη από 302.40Hz και μικρότερη από 320.38Hz. Επομένως, εφόσον ικανοποιείται η συγκεκριμένη συνθήκη, τότε πραγματοποιείται εκτέλεση των εντολών lcd.setCursor() και lcd.print() με την κατάλληλη σειρά, με αποτέλεσμα να εμφανιστεί στην LCD οθόνη το μήνυμα (D#). Με αυτόν τον τρόπο, ο χρήστης αντιλαμβάνεται σε ποια νότα αντιστοιχεί η συγκεκριμένη συχνότητα ταλάντωσης της χορδής.

Αν όμως, δεν ικανοποιείται ούτε η συνθήκη ελέγχου (else if) που αναφέρθηκε προηγουμένως, τότε πραγματοποιείται εκτέλεση της επόμενης εντολής ελέγχου else if (frequency > 320.39 && frequency < 339.43), η οποία ελέγχει αν η τιμή της μεταβλητής (frequency) είναι μεγαλύτερη από 320.39Hz και μικρότερη από 339.43Hz. Συνεπώς, αν ικανοποιείτε η συγκεκριμένη συνθήκη, τότε εκτελούνται με κατάλληλη σειρά οι εντολές lcd.setCursor() και lcd.print(), ώστε να εμφανιστεί στην LCD οθόνη το μήνυμα (E). Έτσι, δηλώνεται στον χρήστη η νότα στην οποία αντιστοιχεί η συγκεκριμένη συχνότητα ταλάντωσης της διεγερμένης χορδής.

Σε αντίθετη περίπτωση, αν δεν ικανοποιείται ούτε η προηγούμενη συνθήκη ελέγχου (else if), τότε εκτελείται η εντολή else if (frequency > 339.44 && frequency < 359.61), η οποία ελέγχει αν η τιμή της μεταβλητής (frequency) είναι μεγαλύτερη από 339.44Hz και μικρότερη από 359.61Hz. Συνεπώς, αν ικανοποιείτε η συγκεκριμένη συνθήκη, τότε εκτελούνται με κατάλληλη σειρά οι εντολές lcd.setCursor() και lcd.print(), ώστε να εμφανιστεί στην LCD οθόνη το μήνυμα (F). Με αυτόν τον τρόπο, δηλώνεται στον χρήστη η νότα στην οποία αντιστοιχεί η συγκεκριμένη συχνότητα ταλάντωσης της χορδής που έχει διεγερθεί.

Στην περίπτωση όμως που δεν ικανοποιείται ούτε αυτή η συνθήκη ελέγχου (else if), τότε εκτελείται η επόμενη εντολή ελέγχου else if (frequency > 359.62 && frequency < 380.99), η οποία ελέγχει αν η τιμή της μεταβλητής (frequency) είναι μεγαλύτερη από 359.62Hz και μικρότερη από 380.99Hz. Επομένως, εφόσον ικανοποιείται η συγκεκριμένη συνθήκη, τότε πραγματοποιείται εκτέλεση των εντολών lcd.setCursor() και lcd.print() με την κατάλληλη σειρά, με αποτέλεσμα να εμφανιστεί στην LCD οθόνη το μήνυμα (F#). Με αυτόν τον τρόπο, ο χρήστης αντιλαμβάνεται σε ποια νότα αντιστοιχεί η συγκεκριμένη συχνότητα ταλάντωσης της χορδής.

Αν όμως, δεν ικανοποιείται ούτε η συνθήκη ελέγχου (else if) που αναφέρθηκε προηγουμένως, τότε πραγματοποιείται εκτέλεση της επόμενης εντολής ελέγχου else if (frequency > 381 && frequency < 403.65), η οποία ελέγχει αν η τιμή της μεταβλητής (frequency) είναι μεγαλύτερη από 381Hz και μικρότερη από 403.65Hz. Συνεπώς, αν ικανοποιείτε η συγκεκριμένη συνθήκη, τότε εκτελούνται με κατάλληλη σειρά οι εντολές lcd.setCursor() και lcd.print(), ώστε να εμφανιστεί στην LCD οθόνη το μήνυμα (G). Έτσι, δηλώνεται στον χρήστη η νότα στην οποία αντιστοιχεί η συγκεκριμένη συχνότητα ταλάντωσης της διεγερμένης χορδής.

Σε αντίθετη περίπτωση, αν δεν ικανοποιείται ούτε η προηγούμενη συνθήκη ελέγχου (else if), τότε εκτελείται η εντολή else if (frequency > 403.66 && frequency < 427.65), η

οποία ελέγχει αν η τιμή της μεταβλητής (frequency) είναι μεγαλύτερη από 403.66Hz και μικρότερη από 427.65Hz. Συνεπώς, αν ικανοποιείτε η συγκεκριμένη συνθήκη, τότε εκτελούνται με κατάλληλη σειρά οι εντολές lcd.setCursor() και lcd.print(), ώστε να εμφανιστεί στην LCD οθόνη το μήνυμα (G#). Με αυτόν τον τρόπο, δηλώνεται στον χρήστη η νότα στην οποία αντιστοιχεί η συγκεκριμένη συχνότητα ταλάντωσης της χορδής που έχει διεγερθεί.

Στην περίπτωση όμως που δεν ικανοποιείται ούτε αυτή η συνθήκη ελέγχου (else if), τότε εκτελείται η επόμενη εντολή ελέγχου else if (frequency > 427.66 && frequency < 453.08), η οποία ελέγχει αν η τιμή της μεταβλητής (frequency) είναι μεγαλύτερη από 427.66Hz και μικρότερη από 453.08Hz. Επομένως, εφόσον ικανοποιείται η συγκεκριμένη συνθήκη, τότε πραγματοποιείται εκτέλεση των εντολών lcd.setCursor() και lcd.print() με την κατάλληλη σειρά, με αποτέλεσμα να εμφανιστεί στην LCD οθόνη το μήνυμα (A). Με αυτόν τον τρόπο, ο χρήστης αντιλαμβάνεται σε ποια νότα αντιστοιχεί η συγκεκριμένη συχνότητα ταλάντωσης της χορδής.

Αν όμως, δεν ικανοποιείται ούτε η συνθήκη ελέγχου (else if) που αναφέρθηκε προηγουμένως, τότε πραγματοποιείται εκτέλεση της επόμενης εντολής ελέγχου else if (frequency > 453.09 && frequency < 480.02), η οποία ελέγχει αν η τιμή της μεταβλητής (frequency) είναι μεγαλύτερη από 453.09Hz και μικρότερη από 480.02Hz. Συνεπώς, αν ικανοποιείτε η συγκεκριμένη συνθήκη, τότε εκτελούνται με κατάλληλη σειρά οι εντολές lcd.setCursor() και lcd.print(), ώστε να εμφανιστεί στην LCD οθόνη το μήνυμα (A#). Έτσι, δηλώνεται στον χρήστη η νότα στην οποία αντιστοιχεί η συγκεκριμένη συχνότητα ταλάντωσης της διεγερμένης χορδής.

Σε αντίθετη περίπτωση, αν δεν ικανοποιείται ούτε η προηγούμενη συνθήκη ελέγχου (else if), τότε εκτελείται η εντολή else if (frequency > 480.03 && frequency < 508.56), η οποία ελέγχει αν η τιμή της μεταβλητής (frequency) είναι μεγαλύτερη από 480.03Hz και μικρότερη από 508.56Hz. Συνεπώς, αν ικανοποιείτε η συγκεκριμένη συνθήκη, τότε εκτελούνται με κατάλληλη σειρά οι εντολές lcd.setCursor() και lcd.print(), ώστε να εμφανιστεί στην LCD οθόνη το μήνυμα (B). Με αυτόν τον τρόπο, δηλώνεται στον χρήστη η νότα στην οποία αντιστοιχεί η συγκεκριμένη συχνότητα ταλάντωσης της χορδής που έχει διεγερθεί.

Στην περίπτωση όμως που δεν ικανοποιείται ούτε αυτή η συνθήκη ελέγχου (else if), τότε εκτελείται η επόμενη εντολή ελέγχου else if (frequency > 508.57 && frequency < 538.81), η οποία ελέγχει αν η τιμή της μεταβλητής (frequency) είναι μεγαλύτερη από 508.57Hz και μικρότερη από 538.81Hz. Επομένως, εφόσον ικανοποιείται η συγκεκριμένη συνθήκη, τότε πραγματοποιείται εκτέλεση των εντολών lcd.setCursor() και lcd.print() με την κατάλληλη σειρά, με αποτέλεσμα να εμφανιστεί στην LCD οθόνη το μήνυμα (C). Με αυτόν τον τρόπο, ο χρήστης αντιλαμβάνεται σε ποια νότα αντιστοιχεί η συγκεκριμένη συχνότητα ταλάντωσης της χορδής.

Τέλος, σε περίπτωση που δεν ικανοποιείται καμία συνθήκη ελέγχου (if – else if), τότε εκτελείται το μπλοκ εντολών της συνθήκης (else). Πιο συγκεκριμένα, πραγματοποιείται μηδενισμός της μεταβλητής (frequency), εξαιτίας της εντολής frequency = 0.00. Ύστερα, εκτελούνται με κατάλληλη σειρά η εντολές lcd.setCursor() και lcd.print(), ώστε να σβηστεί η νότα που εμφανιζόταν μέχρι στιγμής στην LCD οθόνη. Αυτό το μπλοκ εντολών εκτελείται όταν ο χρήστης δεν έχει διεγείρει καμία χορδή ή όταν η συχνότητα ταλάντωσης που έχει υπολογιστεί είναι εκτός ορίων, εξαιτίας της διέγερσης περισσότερων

των χορδών. Έτσι λοιπόν, ολοκληρώνεται το μπλοκ εντολών της συνάρτησης FREQ_DETECTION1().

Πίνακας 3.11: Ο κώδικας προγραμματισμού συνάρτησης FREQ_DETECTION1() [37].

Κώδικας Προγραμματισμού (Μέρος 11ο)

```
void FREQ_DETECTION1(){ //FREQ_DETECTION1() function
 checkClipping(); //Call checkClipping function
 if (checkMaxAmp>ampThreshold) //If checkMaxAmp is bigger than ampThreshold, then
 {
   frequency = 38462/float(period); //Calculate frequency
   Serial.print(frequency); //Print message at serial port
   Serial.println("Hz"); //Print message at serial port
  if (frequency > 63.58 && frequency < 67.35) //If frequency value is bigger than 63.58Hz and smaller than 67.35Hz, then
  {
    lcd.setCursor(10,1); //Set cursor at eleventh column and second line
    lcd.print("C"); //Print message at LCD screen
    lcd.setCursor(11,1); //Set cursor at twelfth column and second line
    lcd.print(" "); //Print message at LCD screen
  }
  else if(frequency > 67.36 && frequency < 71.36) //If frequency value is bigger than 67.36Hz and smaller than 71.36Hz, then
  {
    lcd.setCursor(10,1); //Set cursor at eleventh column and second line
    lcd.print("C#"); //Print message at LCD screen
  }
  else if(frequency > 71.37 && frequency < 75.60) //If frequency value is bigger than 71.37Hz and smaller than 75.60Hz, then
  {
    lcd.setCursor(10,1); //Set cursor at eleventh column and second line
    lcd.print("D"); //Print message at LCD screen
    lcd.setCursor(11,1); //Set cursor at twelfth column and second line
    lcd.print(" "); //Print message at LCD screen
  }
  else if(frequency > 75.61 && frequency < 80.09) //If frequency value is bigger than 75.61Hz and smaller than 80.09Hz, then
  {
    lcd.setCursor(10,1); //Set cursor at eleventh column and second line
    lcd.print("D#"); //Print message at LCD screen
  }
```

```
  else if(frequency > 80.10 && frequency < 84.86) //If frequency value is bigger than 80.10Hz and smaller than 84.86Hz, then
  {
   lcd.setCursor(10,1); //Set cursor at eleventh column and second line
   lcd.print(" "); //Print message at LCD screen
   lcd.setCursor(11,1); //Set cursor at twelfth column and second line
   lcd.print("E"); //Print message at LCD screen
  }
  else if(frequency > 84.87 && frequency < 89.90) //If frequency value is bigger than 84.87Hz and smaller than 89.90Hz, then
  {
   lcd.setCursor(10,1); //Set cursor at eleventh column and second line
   lcd.print("F"); //Print message at LCD screen
   lcd.setCursor(11,1); //Set cursor at twelfth column and second line
   lcd.print(" "); //Print message at LCD screen
  }
  else if (frequency > 89.91 && frequency < 95.25) //If frequency value is bigger than 89.91Hz and smaller than 95.25Hz, then
  {
   lcd.setCursor(10,1); //Set cursor at eleventh column and second line
   lcd.print("F#"); //Print message at LCD screen
  }
  else if (frequency > 95.26 && frequency < 100.91) //If frequency value is bigger than 95.26Hz and smaller than 100.91Hz, then
  {
   lcd.setCursor(10,1); //Set cursor at eleventh column and second line
   lcd.print("G"); //Print message at LCD screen
   lcd.setCursor(11,1); //Set cursor at twelfth column and second line
   lcd.print(" "); //Print message at LCD screen
  }
  else if (frequency > 100.92 && frequency < 106.91) //If frequency value is bigger than 100.92Hz and smaller than 106.90Hz, then
  {
   lcd.setCursor(10,1); //Set cursor at eleventh column and second line
   lcd.print("G#"); //Print message at LCD screen
  }
  else if (frequency > 106.92 && frequency < 113.27) //If frequency value is bigger than 106.92Hz and smaller than 113.27Hz, then
  {
   lcd.setCursor(10,1); //Set cursor at eleventh column and second line
```

```
    lcd.print("A"); //Print message at LCD screen
    lcd.setCursor(11,1); //Set cursor at twelfth column and second line
    lcd.print(" "); //Print message at LCD screen
  }
  else if (frequency > 113.28 && frequency < 120) //If frequency value is bigger than 113.28Hz and smaller than 120Hz, then
  {
    lcd.setCursor(10,1); //Set cursor at eleventh column and second line
    lcd.print("A#"); //Print message at LCD screen
  }
  else if (frequency > 120.01 && frequency < 127.14) //If frequency value is bigger than 120.01Hz and smaller than 127.14Hz, then
  {
    lcd.setCursor(10,1); //Set cursor at eleventh column and second line
    lcd.print("B"); //Print message at LCD screen
    lcd.setCursor(10,1); //Set cursor at twelfth column and second line
    lcd.print(" "); //Print message at LCD screen
  }
  else if (frequency > 127.15 && frequency < 134.81) //If frequency value is bigger than 127.15Hz and smaller than 134.81Hz, then
  {
    lcd.setCursor(10,1); //Set cursor at eleventh column and second line
    lcd.print("C"); //Print message at LCD screen
    lcd.setCursor(11,1); //Set cursor at twelfth column and second line
    lcd.print(" "); //Print message at LCD screen
  }
  else if (frequency > 134.82 && frequency < 142.71)//If frequency value is bigger than 134.82Hz and smaller than 142.71Hz, then
  {
    lcd.setCursor(10,1); //Set cursor at eleventh column and second line
    lcd.print("C#"); //Print message at LCD screen
  }
  else if (frequency > 142.72 && frequency < 151.19) //If frequency value is bigger than 142.72Hz and smaller than 151.19Hz, then
  {
    lcd.setCursor(10,1); //Set cursor at eleventh column and second line
    lcd.print("D"); //Print message at LCD screen
    lcd.setCursor(11,1); //Set cursor at twelfth column and second line
    lcd.print(" "); //Print message at LCD screen
  }
```

```
  else if (frequency > 151.20 && frequency < 160.18) //If frequency value is bigger than 151.207Hz and smaller than 160.18Hz, then
  {
   lcd.setCursor(10,1); //Set cursor at eleventh column and second line
   lcd.print("D#"); //Print message at LCD screen
  }
  else if (frequency > 160.19 && frequency < 169.71) //If frequency value is bigger than 160.19Hz and smaller than 169.71Hz, then
  {
   lcd.setCursor(10,1); //Set cursor at eleventh column and second line
   lcd.print(" "); //Print message at LCD screen
   lcd.setCursor(11,1); //Set cursor at twelfth column and second line
   lcd.print("E"); //Print message at LCD screen
  }
  else if (frequency > 169.72 && frequency < 179.80) //If frequency value is bigger than 169.72Hz and smaller than 179.80Hz, then
  {
   lcd.setCursor(10,1); //Set cursor at eleventh column and second line
   lcd.print("F"); //Print message at LCD screen
   lcd.setCursor(11,1); //Set cursor at twelfth column and second line
   lcd.print(" "); //Print message at LCD screen
  }
  else if (frequency > 179.81 && frequency < 190.50) //If frequency value is bigger than 179.81Hz and smaller than 190.50Hz, then
  {
   lcd.setCursor(10,1); //Set cursor at eleventh column and second line
   lcd.print("F#"); //Print message at LCD screen
  }
  else if (frequency > 190.51 && frequency < 201.82) //If frequency value is bigger than 190.51Hz and smaller than 201.82Hz, then
  {
   lcd.setCursor(10,1); //Set cursor at eleventh column and second line
   lcd.print("G"); //Print message at LCD screen
   lcd.setCursor(11,1); //Set cursor at twelfth column and second line
   lcd.print(" "); //Print message at LCD screen
  }
  else if (frequency > 201.83 && frequency < 213.82) //If frequency value is bigger than 201.83Hz and smaller than 213.82Hz, then
  {
   lcd.setCursor(10,1); //Set cursor at eleventh column and second line
```

```
    lcd.print("G#"); //Print message at LCD screen
   }
   else if (frequency > 213.83 && frequency < 226.54) //If frequency value is bigger than 213.83Hz and smaller than 226.54Hz, then
   {
    lcd.setCursor(10,1); //Set cursor at eleventh column and second line
    lcd.print("A"); //Print message at LCD screen
    lcd.setCursor(11,1); //Set cursor at twelfth column and second line
    lcd.print(" "); //Print message at LCD screen
   }
   else if (frequency > 226.55 && frequency < 240.01) //If frequency value is bigger than 226.55Hz and smaller than 240.01Hz, then
   {
    lcd.setCursor(10,1); //Set cursor at eleventh column and second line
    lcd.print("A#"); //Print message at LCD screen
   }
   else if (frequency > 240.02 && frequency < 254.28) //If frequency value is bigger than 240.02Hz and smaller than 254.28Hz, then
   {
    lcd.setCursor(10,1); //Set cursor at eleventh column and second line
    lcd.print("B"); //Print message at LCD screen
    lcd.setCursor(11,1); //Set cursor at twelfth column and second line
    lcd.print(" "); //Print message at LCD screen
   }
   else if (frequency > 254.29 && frequency < 269.40) //If frequency value is bigger than 254.29Hz and smaller than 269.40Hz, then
   {
    lcd.setCursor(10,1); //Set cursor at eleventh column and second line
    lcd.print("C"); //Print message at LCD screen
    lcd.setCursor(11,1); //Set cursor at twelfth column and second line
    lcd.print(" "); //Print message at LCD screen
   }
   else if (frequency > 269.41 && frequency < 285.42) //If frequency value is bigger than 269.41Hz and smaller than 285.42Hz, then
   {
    lcd.setCursor(10,1); //Set cursor at eleventh column and second line
    lcd.print("C#"); //Print message at LCD screen
   }
   else if (frequency > 285.43 && frequency < 302.39) //If frequency value is bigger than 285.43Hz and smaller than 302.39Hz, then
```

```
  {
   lcd.setCursor(10,1); //Set cursor at eleventh column and second line
   lcd.print("D"); //Print message at LCD screen
   lcd.setCursor(11,1); //Set cursor at twelfth column and second line
   lcd.print(" "); //Print message at LCD screen
  }
  else if (frequency > 302.40 && frequency < 320.38) //If frequency value is bigger than 302.40Hz and smaller than 320.38Hz, then
  {
   lcd.setCursor(10,1); //Set cursor at eleventh column and second line
   lcd.print("D#"); //Print message at LCD screen
  }
  else if (frequency > 320.39 && frequency < 339.43) //If frequency value is bigger than 320.39Hz and smaller than 339.43Hz, then
  {
   lcd.setCursor(10,1); //Set cursor at eleventh column and second line
   lcd.print("E"); //Print message at LCD screen
   lcd.setCursor(11,1); //Set cursor at twelfth column and second line
   lcd.print(" "); //Print message at LCD screen
  }
  else if (frequency > 339.44 && frequency < 359.61) //If frequency value is bigger than 339.44Hz and smaller than 359.61Hz, then
  {
   lcd.setCursor(10,1); //Set cursor at eleventh column and second line
   lcd.print("F"); //Print message at LCD screen
   lcd.setCursor(11,1); //Set cursor at twelfth column and second line
   lcd.print(" "); //Print message at LCD screen
  }
  else if (frequency > 359.62 && frequency < 380.99) //If frequency value is bigger than 359.62Hz and smaller than 380.99Hz, then
  {
   lcd.setCursor(10,1); //Set cursor at eleventh column and second line
   lcd.print("F#"); //Print message at LCD screen
  }
  else if (frequency > 381 && frequency < 403.65) //If frequency value is bigger than 381Hz and smaller than 403.65Hz, then
  {
   lcd.setCursor(10,1); //Set cursor at eleventh column and second line
   lcd.print("G"); //Print message at LCD screen
   lcd.setCursor(11,1); //Set cursor at twelfth column and second line
```

```
    lcd.print(" "); //Print message at LCD screen
   }
   else if (frequency > 403.66 && frequency < 427.65) //If frequency value is bigger than 403.66Hz and smaller than 427.65Hz, then
   {
    lcd.setCursor(10,1); //Set cursor at eleventh column and second line
    lcd.print("G#"); //Print message at LCD screen
   }
   else if (frequency > 427.66 && frequency < 453.08) //If frequency value is bigger than 427.66Hz and smaller than 453.08Hz, then
   {
    lcd.setCursor(10,1); //Set cursor at eleventh column and second line
    lcd.print("A"); //Print message at LCD screen
    lcd.setCursor(11,1); //Set cursor at twelfth column and second line
    lcd.print(" "); //Print message at LCD screen
   }
   else if (frequency > 453.09 && frequency < 480.02) //If frequency value is bigger than 453.09Hz and smaller than 480.02Hz, then
   {
    lcd.setCursor(10,1); //Set cursor at eleventh column and second line
    lcd.print("A#"); //Print message at LCD screen
   }
   else if (frequency > 480.03 && frequency < 508.56) //If frequency value is bigger than 480.03Hz and smaller than 508.56Hz, then
   {
    lcd.setCursor(10,1); //Set cursor at eleventh column and second line
    lcd.print("B"); //Print message at LCD screen
    lcd.setCursor(11,1); //Set cursor at twelfth column and second line
    lcd.print(" "); //Print message at LCD screen
   }
   else if (frequency > 508.57 && frequency < 538.81) //If frequency value is bigger than 508.57Hz and smaller than 538.81Hz, then
   {
    lcd.setCursor(10,1); //Set cursor at eleventh column and second line
    lcd.print("C"); //Print message at LCD screen
    lcd.setCursor(11,1); //Set cursor at twelfth column and second line
    lcd.print(" "); //Print message at LCD screen
   }
   else //Else
   {
```

```
    frequency = 0.00; //Reset frequency value
    lcd.setCursor(10,1); //Set cursor at eleventh column and second line
    lcd.print(" "); //Print message at LCD screen
    lcd.setCursor(11,1); //Set cursor at twelfth column and second line
    lcd.print(" "); //Print message at LCD screen
   }
  }//if (checkMaxAmp...
  //Serial.println(note);
}//void FREQ_DETECTION
```

3.1.8 **Ανάλυση των συναρτήσεων Clockwise_eStep_Motor () & Counterclockwise_eStep_Motor () (Κώδικας προγραμματισμού Μέρος 12ο)**

Σε αυτή την ενότητα πραγματοποιείται ανάλυση των συναρτήσεων που είναι υπεύθυνες για τον έλεγχο της περιστροφή του βηματικού κινητήρα της χορδής Μι καντίνι (e). Πιο συγκεκριμένα, πραγματοποιείται ανάλυση της συνάρτησης Clockwise_eStep_Motor (), η οποία είναι υπεύθυνη για την ωρολογιακή περιστροφή του βηματικού κινητήρα της χορδής Μι καντίνι (e) καθώς και της συνάρτησης Counterclockwise_eStep_Motor (), η οποία είναι υπεύθυνη για την αντί-ωρολογιακή περιστροφή του κινητήρα της χορδής Μι καντίνι (e) (Πίνακας 3.12). Ουσιαστικά, οι συναρτήσεις αυτές καλούνται εντός του μπλοκ εντολών της συνάρτησης ButtonRead() με σκοπό τον έλεγχο του κλειδιού της χορδής Μι καντίνι (e) είτε κατά χειροκίνητο έλεγχο είτε κατά το κούρδισμα της συγκεκριμένης χορδής.

Αρχικά, εντός του μπλοκ εντολών της συνάρτησης Clockwise_eStep_Motor () εκτελούνται εντολές οι οποίες αποσκοπούν στον έλεγχο των ψηφιακών εξόδων του Arduino MEGA 2560. Αυτό επιτυγχάνεται με την χρήση της εντολής digitalWrite(), η οποία θα θέσει την αντίστοιχη ψηφιακή θύρα είτε σε υψηλό δυναμικό είτε σε χαμηλό δυναμικό. Επίσης, όπως έχει αναφερθεί στο Κεφάλαιο 2, ο βηματικός κινητήρας, αποτελείται από τέσσερις μαγνήτες όπου ο κάθε ένας ελέγχεται από μία ψηφιακή θύρα του Arduino, μέσω της πλακέτας οδήγησης του βηματικού κινητήρα. Επομένως, δημιουργούνται τετράδες εντολών digitalWrite(), ώστε να επιτευχθεί ο έλεγχος και των τεσσάρων μαγνητών του βηματικού κινητήρα. Πιο αναλυτικά, στην πρώτη τετράδα εντολών digitalWrite(), οι ψηφιακοί ακροδέκτες του Arduino (eStepper_Pin1) και (eStepper_Pin2) θέτονται κατάσταση (LOW) ή αλλιώς σε χαμηλό δυναμικό, ενώ οι ψηφιακοί ακροδέκτες (eStepper_Pin3) και (eStepper_Pin4) θέτονται σε κατάσταση (HIGH) ή αλλιώς σε υψηλό δυναμικό. Έπειτα, εκτελείτε μια μικρή καθυστέρηση των 2msec εξαιτίας της εντολής καθυστέρησης delay(2). Σκοπός της μικρής αυτής καθυστέρησης είναι να προλάβει να ολοκληρώσει ο βηματικός κινητήρας την περιστροφή του άξονά του. Στην δεύτερη τετράδα εντολών digitalWrite(), οι ψηφιακοί ακροδέκτες του Arduino (eStepper_Pin2) και (eStepper_Pin3) θέτονται κατάσταση (LOW) ή αλλιώς σε χαμηλό δυναμικό, ενώ οι ψηφιακοί ακροδέκτες (eStepper_Pin1) και (eStepper_Pin4) θέτονται σε κατάσταση (HIGH) ή αλλιώς σε υψηλό δυναμικό. Ύστερα, εκτελείτε ξανά μια μικρή καθυστέρηση των 2msec εξαιτίας της εντολής καθυστέρησης delay(2) ώστε να

προλάβει ο βηματικός κινητήρας να ολοκληρώσει την περιστροφή του άξονά του. Στην συνέχεια, εκτελείται η τρίτη τετράδα εντολών digitalWrite(), οι οποίες θέτουν του ψηφιακούς ακροδέκτες του Arduino (eStepper_Pin3) και (eStepper_Pin4) σε κατάσταση (LOW) ή αλλιώς σε χαμηλό δυναμικό, ενώ οι ψηφιακοί ακροδέκτες (eStepper_Pin1) και (eStepper_Pin2) θέτονται σε κατάσταση (HIGH) ή αλλιώς σε υψηλό δυναμικό. Έπειτα, εκτελείται ξανά η εντολή καθυστέρησης delay(2) προκαλώντας μια καθυστέρηση των 2msec με σκοπό την ολοκλήρωση της περιστροφής του άξονα του βηματικού κινητήρα. Ύστερα, εκτελείται η τέταρτη τετράδα εντολών digitalWrite(), οι οποίες θέτουν του ψηφιακούς ακροδέκτες του Arduino (eStepper_Pin1) και (eStepper_Pin4) σε κατάσταση (LOW) ή αλλιώς σε χαμηλό δυναμικό, ενώ οι ψηφιακοί ακροδέκτες (eStepper_Pin2) και (eStepper_Pin3) θέτονται σε κατάσταση (HIGH) ή αλλιώς σε υψηλό δυναμικό. Εν συνεχεία, πραγματοποιείται ξανά μια καθυστέρηση των 2msec εξαιτίας της εντολής delay(2) με την οποία ολοκληρώνεται το μπλοκ εντολών της συνάρτησης Clockwise_eStep_Motor ().

Όσον αφορά το μπλοκ εντολών της συνάρτησης Counterclockwise_eStep_Motor (), ακολουθείτε μια παρόμοια λογική ενεργοποίησης των ψηφιακών ακροδεκτών του Arduino με την διαφορά ότι ενεργοποιούνται ανάποδα σε σχέση με την προηγούμενη συνάρτηση Clockwise_eStep_Motor (). Πιο αναλυτικά, στην πρώτη τετράδα εντολών digitalWrite(), οι ψηφιακοί ακροδέκτες του Arduino (eStepper_Pin3) και (eStepper_Pin4) θέτονται κατάσταση (LOW) ή αλλιώς σε χαμηλό δυναμικό, ενώ οι ψηφιακοί ακροδέκτες (eStepper_Pin1) και (eStepper_Pin2) θέτονται σε κατάσταση (HIGH) ή αλλιώς σε υψηλό δυναμικό. Έπειτα, εκτελείτε μια μικρή καθυστέρηση των 2msec εξαιτίας της εντολής καθυστέρησης delay(2). Σκοπός της μικρής αυτής καθυστέρησης είναι να προλάβει να ολοκληρώσει ο βηματικός κινητήρας την περιστροφή του άξονά του. Στην δεύτερη τετράδα εντολών digitalWrite(), οι ψηφιακοί ακροδέκτες του Arduino (eStepper_Pin2) και (eStepper_Pin3) θέτονται κατάσταση (LOW) ή αλλιώς σε χαμηλό δυναμικό, ενώ οι ψηφιακοί ακροδέκτες (eStepper_Pin1) και (eStepper_Pin4) θέτονται σε κατάσταση (HIGH) ή αλλιώς σε υψηλό δυναμικό. Ύστερα, εκτελείτε ξανά μια μικρή καθυστέρηση των 2msec εξαιτίας της εντολής καθυστέρησης delay(2) ώστε να προλάβει ο βηματικός κινητήρας να ολοκληρώσει την περιστροφή του άξονά του. Στην συνέχεια, εκτελείται η τρίτη τετράδα εντολών digitalWrite(), οι οποίες θέτουν του ψηφιακούς ακροδέκτες του Arduino (eStepper_Pin1) και (eStepper_Pin2) σε κατάσταση (LOW) ή αλλιώς σε χαμηλό δυναμικό, ενώ οι ψηφιακοί ακροδέκτες (eStepper_Pin3) και (eStepper_Pin4) θέτονται σε κατάσταση (HIGH) ή αλλιώς σε υψηλό δυναμικό. Έπειτα, εκτελείται ξανά η εντολή καθυστέρησης delay(2) προκαλώντας μια καθυστέρηση των 2msec με σκοπό την ολοκλήρωση της περιστροφής του άξονα του βηματικού κινητήρα. Ύστερα, εκτελείται η τέταρτη τετράδα εντολών digitalWrite(),οι οποίες θέτουν του ψηφιακούς ακροδέκτες του Arduino (eStepper_Pin1) και (eStepper_Pin4) σε κατάσταση (LOW) ή αλλιώς σε χαμηλό δυναμικό, ενώ οι ψηφιακοί ακροδέκτες (eStepper_Pin2) και (eStepper_Pin3) θέτονται σε κατάσταση (HIGH) ή αλλιώς σε υψηλό δυναμικό. Τέλος, το μπλοκ εντολών της συνάρτησης Counterclockwise_eStep_Motor () ολοκληρώνεται με τη εκτέλεση της εντολής καθυστέρησης delay(2), η οποία δημιουργεί μια καθυστέρηση των 2msec.

Πίνακας 3.12: Ο κώδικας προγραμματισμού των συναρτήσεων Clockwise_eStep_Motor () & Counterclockwise_eStep_Motor () [38].

Κώδικας προγραμματισμού (Μέρος 12ο)

```
void Clockwise_eStep_Motor () { // Function for clockwise movement of eStep_Motor
  digitalWrite(eStepper_Pin1,LOW); //Set eStepper_Pin1 LOW = 0
  digitalWrite(eStepper_Pin2,LOW); //Set eStepper_Pin2 LOW = 0
  digitalWrite(eStepper_Pin3,HIGH); //Set eStepper_Pin3 HIGH = 1
  digitalWrite(eStepper_Pin4, HIGH); //Set eStepper_Pin4 HIGH = 1
  delay(2); //Time delay 2 msec

  digitalWrite(eStepper_Pin1,HIGH); //Set eStepper_Pin1 HIGH = 1
  digitalWrite(eStepper_Pin2,LOW); //Set eStepper_Pin2 LOW = 0
  digitalWrite(eStepper_Pin3,LOW); //Set eStepper_Pin3 LOW = 0
  digitalWrite(eStepper_Pin4, HIGH); //Set eStepper_Pin4 HIGH = 1
  delay(2); //Time delay 2 msec

  digitalWrite(eStepper_Pin1,HIGH); //Set eStepper_Pin1 HIGH = 1
  digitalWrite(eStepper_Pin2,HIGH); //Set eStepper_Pin2 HIGH = 1
  digitalWrite(eStepper_Pin3,LOW); //Set eStepper_Pin3 LOW = 0
  digitalWrite(eStepper_Pin4, LOW); //Set eStepper_Pin4 LOW = 0
  delay(2); //Time delay 2 msec

  digitalWrite(eStepper_Pin1,LOW); //Set eStepper_Pin1 LOW = 0
  digitalWrite(eStepper_Pin2,HIGH); //Set eStepper_Pin2 HIGH = 1
  digitalWrite(eStepper_Pin3,HIGH); //Set eStepper_Pin3 HIGH = 1
  digitalWrite(eStepper_Pin4, LOW); //Set eStepper_Pin4 LOW = 0
  delay(2); //Time delay 2 msec
}
void Counterclockwise_eStep_Motor () { // Function for Counterclockwise movement of
eStep_Motor
  digitalWrite(eStepper_Pin1, HIGH); //Set eStepper_Pin1 HIGH = 1
  digitalWrite(eStepper_Pin2, HIGH); //Set eStepper_Pin2 HIGH = 1
  digitalWrite(eStepper_Pin3, LOW); //Set eStepper_Pin1 LOW = 0
  digitalWrite(eStepper_Pin4, LOW); //Set eStepper_Pin1 LOW = 0
  delay(2); //Time delay 2 msec

  digitalWrite(eStepper_Pin1, HIGH); //Set eStepper_Pin1 HIGH = 1
  digitalWrite(eStepper_Pin2, LOW); //Set eStepper_Pin2 LOW = 0
  digitalWrite(eStepper_Pin3, LOW); //Set eStepper_Pin1 LOW = 0
  digitalWrite(eStepper_Pin4, HIGH); //Set eStepper_Pin1 HIGH = 1
  delay(2); //Time delay 2 msec
```

```
  digitalWrite(eStepper_Pin1, LOW); //Set eStepper_Pin1 LOW = 0
  digitalWrite(eStepper_Pin2, LOW); //Set eStepper_Pin2 LOW = 0
  digitalWrite(eStepper_Pin3, HIGH); //Set eStepper_Pin1 HIGH = 1
  digitalWrite(eStepper_Pin4, HIGH); //Set eStepper_Pin1 HIGH = 1
  delay(2); //Time delay 2 msec

  digitalWrite(eStepper_Pin1, LOW); //Set eStepper_Pin1 LOW = 0
  digitalWrite(eStepper_Pin2, HIGH); //Set eStepper_Pin2 HIGH = 1
  digitalWrite(eStepper_Pin3, HIGH); //Set eStepper_Pin1 HIGH = 1
  digitalWrite(eStepper_Pin4, LOW); //Set eStepper_Pin1 LOW = 0
  delay(2); //Time delay 2 msec
}
```

3.1.9 **Ανάλυση των συναρτήσεων Clockwise_BStep_Motor () & Counterclockwise_BStep_Motor () (Κώδικας προγραμματισμού Μέρος 13ο)**

Σε αυτή την ενότητα πραγματοποιείται ανάλυση των συναρτήσεων που είναι υπεύθυνες για τον έλεγχο της περιστροφή του βηματικού κινητήρα της χορδής Σι (B). Ουσιαστικά, πραγματοποιείται ανάλυση της συνάρτησης Clockwise_BStep_Motor (), η οποία είναι υπεύθυνη για την ωρολογιακή περιστροφή του βηματικού κινητήρα της χορδής Σι (B) καθώς και της συνάρτησης Counterclockwise_BStep_Motor (), η οποία είναι υπεύθυνη για την αντί-ωρολογιακή περιστροφή του κινητήρα της χορδής Σι (B) (Πίνακας 3.13). Πιο συγκεκριμένα, οι συναρτήσεις αυτές καλούνται εντός του μπλοκ εντολών της συνάρτησης ButtonRead() με σκοπό τον έλεγχο του κλειδιού της χορδής Σι (B) είτε κατά τον χειροκίνητο έλεγχο είτε κατά το κούρδισμα της συγκεκριμένης χορδής.

Αρχικά, εντός του μπλοκ εντολών της συνάρτησης Clockwise_BStep_Motor () εκτελούνται κατάλληλα οι εντολές digitalWrite() οι οποίες αποσκοπούν στον έλεγχο των ψηφιακών εξόδων του Arduino MEGA 2560. Ακόμα, όπως έχει αναφερθεί στο Κεφάλαιο 2, κάθε ένας από τους τέσσερις μαγνήτες του βηματικού κινητήρα, ελέγχεται από μία ψηφιακή θύρα του Arduino, μέσω της πλακέτας οδήγησης του βηματικού κινητήρα. Επομένως, για την επίτευξη του ελέγχου των τεσσάρων μαγνητών του βηματικού κινητήρα, δημιουργούνται τετράδες εντολών digitalWrite(). Πιο συγκεκριμένα, στην πρώτη τετράδα εντολών digitalWrite(), οι ψηφιακοί ακροδέκτες του Arduino (BStepper_Pin1) και (BStepper_Pin2) θέτονται κατάσταση (LOW) ή αλλιώς σε χαμηλό δυναμικό, ενώ οι ψηφιακοί ακροδέκτες (BStepper_Pin3) και (BStepper_Pin4) θέτονται σε κατάσταση (HIGH) ή αλλιώς σε υψηλό δυναμικό. Στην συνέχεια, πραγματοποιείται μια καθυστέρηση των 2msec εξαιτίας της εντολής καθυστέρησης delay(2). Η μικρή αυτή καθυστέρηση χρειάζεται για να ολοκληρώσει ο βηματικός κινητήρας την περιστροφή του άξονά του. Στην δεύτερη τετράδα εντολών digitalWrite(), οι ψηφιακοί ακροδέκτες του Arduino (BStepper_Pin2) και (BStepper_Pin3) θέτονται κατάσταση (LOW) ή αλλιώς σε χαμηλό δυναμικό, ενώ οι ψηφιακοί ακροδέκτες (BStepper_Pin1) και (BStepper_Pin4) θέτονται σε κατάσταση (HIGH) ή αλλιώς σε υψηλό δυναμικό. Έπειτα, εκτελείτε ξανά μια

μικρή καθυστέρηση των 2msec εξαιτίας της εντολής καθυστέρησης delay(2) ώστε να προλάβει ο βηματικός κινητήρας να ολοκληρώσει την περιστροφή του άξονά του. Στην συνέχεια, εκτελείται η τρίτη τετράδα εντολών digitalWrite(), οι οποίες θέτουν του ψηφιακούς ακροδέκτες του Arduino (BStepper_Pin3) και (BStepper_Pin4) σε κατάσταση (LOW) ή αλλιώς σε χαμηλό δυναμικό, ενώ οι ψηφιακοί ακροδέκτες (BStepper_Pin1) και (BStepper_Pin2) θέτονται σε κατάσταση (HIGH) ή αλλιώς σε υψηλό δυναμικό. Ύστερα, εκτελείται ξανά η εντολή καθυστέρησης delay(2) προκαλώντας μια καθυστέρηση των 2msec με σκοπό την ολοκλήρωση της περιστροφής του άξονα του βηματικού κινητήρα. Έπειτα, εκτελείται η τέταρτη τετράδα εντολών digitalWrite(), οι οποίες θέτουν του ψηφιακούς ακροδέκτες του Arduino (BStepper_Pin1) και (BStepper_Pin4) σε κατάσταση (LOW) ή αλλιώς σε χαμηλό δυναμικό, ενώ οι ψηφιακοί ακροδέκτες (BStepper_Pin2) και (BStepper_Pin3) θέτονται σε κατάσταση (HIGH) ή αλλιώς σε υψηλό δυναμικό. Αμέσως μετά, πραγματοποιείται ξανά μια καθυστέρηση των 2msec εξαιτίας της εντολής delay(2) με την οποία ολοκληρώνεται το μπλοκ εντολών της συνάρτησης Clockwise_BStep_Motor ().

Από την άλλη πλευρά όμως, το μπλοκ εντολών της συνάρτησης Counterclockwise_BStep_Motor (), ακολουθεί μια ανάποδη σειρά ενεργοποίησης των ψηφιακών ακροδεκτών του Arduino, σε σχέση με την προηγούμενη συνάρτηση Clockwise_BStep_Motor (). Πιο συγκεκριμένα, στην πρώτη τετράδα εντολών digitalWrite(), οι ψηφιακοί ακροδέκτες του Arduino (BStepper_Pin3) και (BStepper_Pin4) θέτονται κατάσταση (LOW) ή αλλιώς σε χαμηλό δυναμικό, ενώ οι ψηφιακοί ακροδέκτες (BStepper_Pin1) και (BStepper_Pin2) θέτονται σε κατάσταση (HIGH) ή αλλιώς σε υψηλό δυναμικό. Εν συνεχεία, εκτελείτε μια μικρή καθυστέρηση των 2msec εξαιτίας της εντολής καθυστέρησης delay(2), ώστε ο βηματικός κινητήρας να ολοκληρώσει την περιστροφή του άξονά του. Στην δεύτερη τετράδα εντολών digitalWrite(), οι ψηφιακοί ακροδέκτες του Arduino (BStepper_Pin2) και (BStepper_Pin3) θέτονται κατάσταση (LOW) ή αλλιώς σε χαμηλό δυναμικό, ενώ οι ψηφιακοί ακροδέκτες (BStepper_Pin1) και (BStepper_Pin4) θέτονται σε κατάσταση (HIGH) ή αλλιώς σε υψηλό δυναμικό. Έπειτα, εκτελείτε ξανά μια μικρή καθυστέρηση των 2msec εξαιτίας της εντολής καθυστέρησης delay(2) ώστε να προλάβει ο βηματικός κινητήρας να ολοκληρώσει την περιστροφή του άξονά του. Ύστερα, εκτελείται η τρίτη τετράδα εντολών digitalWrite(), οι οποίες θέτουν του ψηφιακούς ακροδέκτες του Arduino (BStepper_Pin1) και (BStepper_Pin2) σε κατάσταση (LOW) ή αλλιώς σε χαμηλό δυναμικό, ενώ οι ψηφιακοί ακροδέκτες (BStepper_Pin3) και (BStepper_Pin4) θέτονται σε κατάσταση (HIGH) ή αλλιώς σε υψηλό δυναμικό. Έπειτα, εκτελείται ξανά η εντολή καθυστέρησης delay(2) προκαλώντας μια καθυστέρηση των 2msec με σκοπό την ολοκλήρωση της περιστροφής του άξονα του βηματικού κινητήρα. Στην συνέχεια, εκτελείται η τέταρτη τετράδα εντολών digitalWrite(),οι οποίες θέτουν του ψηφιακούς ακροδέκτες του Arduino (BStepper_Pin1) και (BStepper_Pin4) σε κατάσταση (LOW) ή αλλιώς σε χαμηλό δυναμικό, ενώ οι ψηφιακοί ακροδέκτες (BStepper_Pin2) και (BStepper_Pin3) θέτονται σε κατάσταση (HIGH) ή αλλιώς σε υψηλό δυναμικό. Ολοκληρώνοντας την ανάλυση του μπλοκ εντολών της συνάρτησης Counterclockwise_BStep_Motor (), πραγματοποιείται εκτέλεση της εντολής καθυστέρησης delay(2), η οποία δημιουργεί μια καθυστέρηση των 2msec η οποία είναι αναγκαία για να ολοκληρώσει ο βηματικός κινητήρας την περιστροφή του άξονά του.

Πίνακας 3.13: Ο κώδικας προγραμματισμού των συναρτήσεων Clockwise_BStep_Motor () & Counterclockwise_BStep_Motor () [39].

Κώδικας προγραμματισμού (Μέρος 13ο)

```
void Clockwise_BStep_Motor () { // Function for clockwise movement of BStep_Motor
  digitalWrite(BStepper_Pin1,LOW); //Set BStepper_Pin1 LOW = 0
  digitalWrite(BStepper_Pin2,LOW); //Set BStepper_Pin2 LOW = 0
  digitalWrite(BStepper_Pin3,HIGH); //Set BStepper_Pin3 HIGH = 1
  digitalWrite(BStepper_Pin4, HIGH); //Set BStepper_Pin4 HIGH = 1
  delay(2); //Time delay 2 msec

  digitalWrite(BStepper_Pin1,HIGH); //Set BStepper_Pin1 HIGH = 1
  digitalWrite(BStepper_Pin2,LOW); //Set BStepper_Pin2 LOW = 0
  digitalWrite(BStepper_Pin3,LOW); //Set BStepper_Pin3 LOW = 0
  digitalWrite(BStepper_Pin4, HIGH); //Set BStepper_Pin4 HIGH = 1
  delay(2); //Time delay 2 msec

  digitalWrite(BStepper_Pin1,HIGH); //Set BStepper_Pin1 HIGH = 1
  digitalWrite(BStepper_Pin2,HIGH); //Set BStepper_Pin2 HIGH = 1
  digitalWrite(BStepper_Pin3,LOW); //Set BStepper_Pin3 LOW = 0
  digitalWrite(BStepper_Pin4, LOW); //Set BStepper_Pin4 LOW = 0
  delay(2); //Time delay 2 msec

  digitalWrite(BStepper_Pin1,LOW); //Set BStepper_Pin1 LOW = 0
  digitalWrite(BStepper_Pin2,HIGH); //Set BStepper_Pin2 HIGH = 1
  digitalWrite(BStepper_Pin3,HIGH); //Set BStepper_Pin3 HIGH = 1
  digitalWrite(BStepper_Pin4, LOW); //Set BStepper_Pin4 LOW = 0
  delay(2); //Time delay 2 msec
}

void Counterclockwise_BStep_Motor () { // Function for Counterclockwise movement of BStep_Motor
  digitalWrite(BStepper_Pin1, HIGH); //Set BStepper_Pin1 HIGH = 1
  digitalWrite(BStepper_Pin2, HIGH); //Set BStepper_Pin2 HIGH = 1
  digitalWrite(BStepper_Pin3, LOW); //Set BStepper_Pin1 LOW = 0
  digitalWrite(BStepper_Pin4, LOW); //Set BStepper_Pin1 LOW = 0
  delay(2); //Time delay 2 msec

  digitalWrite(BStepper_Pin1, HIGH); //Set BStepper_Pin1 HIGH = 1
  digitalWrite(BStepper_Pin2, LOW); //Set BStepper_Pin2 LOW = 0
  digitalWrite(BStepper_Pin3, LOW); //Set BStepper_Pin1 LOW = 0
```

```
  digitalWrite(BStepper_Pin4, HIGH); //Set BStepper_Pin1 HIGH = 1
  delay(2); //Time delay 2 msec

  digitalWrite(BStepper_Pin1, LOW); //Set BStepper_Pin1 LOW = 0
  digitalWrite(BStepper_Pin2, LOW); //Set BStepper_Pin2 LOW = 0
  digitalWrite(BStepper_Pin3, HIGH); //Set BStepper_Pin1 HIGH = 1
  digitalWrite(BStepper_Pin4, HIGH); //Set BStepper_Pin1 HIGH = 1
  delay(2); //Time delay 2 msec

  digitalWrite(BStepper_Pin1, LOW); //Set BStepper_Pin1 LOW = 0
  digitalWrite(BStepper_Pin2, HIGH); //Set BStepper_Pin2 HIGH = 1
  digitalWrite(BStepper_Pin3, HIGH); //Set BStepper_Pin1 HIGH = 1
  digitalWrite(BStepper_Pin4, LOW); //Set BStepper_Pin1 LOW = 0
  delay(2); //Time delay 2 msec
}
```

3.1.10 Ανάλυση των συναρτήσεων Clockwise_GStep_Motor () & Counterclockwise_GStep_Motor () (Κώδικας προγραμματισμού Μέρος 14ο)

Σε αυτή την ενότητα πραγματοποιείται ανάλυση των συναρτήσεων που είναι υπεύθυνες για τον έλεγχο της περιστροφή του βηματικού κινητήρα της χορδής Σολ (G). Πιο συγκεκριμένα, πραγματοποιείται ανάλυση της συνάρτησης Clockwise_GStep_Motor (), η οποία είναι υπεύθυνη για την ωρολογιακή περιστροφή του βηματικού κινητήρα της χορδής Σολ (G) καθώς και της συνάρτησης Counterclockwise_GStep_Motor (), η οποία είναι υπεύθυνη για την αντί-ωρολογιακή περιστροφή του κινητήρα της χορδής Σολ (G) (Πίνακας 3.14). Ουσιαστικά, οι συναρτήσεις αυτές καλούνται εντός του μπλοκ εντολών της συνάρτησης ButtonRead() με σκοπό τον έλεγχο του κλειδιού της χορδής Σολ (G) είτε κατά χειροκίνητο έλεγχο είτε κατά το κούρδισμα της συγκεκριμένης χορδής.

Αρχικά, εντός του μπλοκ εντολών της συνάρτησης Clockwise_GStep_Motor () εκτελούνται εντολές οι οποίες αποσκοπούν στον έλεγχο των ψηφιακών εξόδων του Arduino MEGA 2560. Αυτό επιτυγχάνεται με την χρήση της εντολής digitalWrite(), η οποία θα θέσει την αντίστοιχη ψηφιακή θύρα είτε σε υψηλό δυναμικό είτε σε χαμηλό δυναμικό. Επίσης, όπως έχει αναφερθεί στο Κεφάλαιο 2, ο βηματικός κινητήρας, αποτελείται από τέσσερις μαγνήτες όπου ο κάθε ένας ελέγχεται από μία ψηφιακή θύρα του Arduino, μέσω της πλακέτας οδήγησης του βηματικού κινητήρα. Επομένως, δημιουργούνται τετράδες εντολών digitalWrite(), ώστε να επιτευχθεί ο έλεγχος και των τεσσάρων μαγνητών του βηματικού κινητήρα. Πιο αναλυτικά, στην πρώτη τετράδα εντολών digitalWrite(), οι ψηφιακοί ακροδέκτες του Arduino (GStepper_Pin1) και (GStepper_Pin2) θέτονται κατάσταση (LOW) ή αλλιώς σε χαμηλό δυναμικό, ενώ οι ψηφιακοί ακροδέκτες (GStepper_Pin3) και (GStepper_Pin4) θέτονται σε κατάσταση (HIGH) ή αλλιώς σε υψηλό δυναμικό. Έπειτα, εκτελείτε μια μικρή καθυστέρηση των 2msec εξαιτίας της εντολής καθυστέρησης delay(2). Σκοπός της μικρής αυτής

καθυστέρησης είναι να προλάβει να ολοκληρώσει ο βηματικός κινητήρας την περιστροφή του άξονά του. Στην δεύτερη τετράδα εντολών digitalWrite(), οι ψηφιακοί ακροδέκτες του Arduino (GStepper_Pin2) και (GStepper_Pin3) θέτονται κατάσταση (LOW) ή αλλιώς σε χαμηλό δυναμικό, ενώ οι ψηφιακοί ακροδέκτες (GStepper_Pin1) και (GStepper_Pin4) θέτονται σε κατάσταση (HIGH) ή αλλιώς σε υψηλό δυναμικό. Ύστερα, εκτελείτε ξανά μια μικρή καθυστέρηση των 2msec εξαιτίας της εντολής καθυστέρησης delay(2) ώστε να προλάβει ο βηματικός κινητήρας να ολοκληρώσει την περιστροφή του άξονά του. Στην συνέχεια, εκτελείται η τρίτη τετράδα εντολών digitalWrite(), οι οποίες θέτουν του ψηφιακούς ακροδέκτες του Arduino (GStepper_Pin3) και (GStepper_Pin4) σε κατάσταση (LOW) ή αλλιώς σε χαμηλό δυναμικό, ενώ οι ψηφιακοί ακροδέκτες (GStepper_Pin1) και (GStepper_Pin2) θέτονται σε κατάσταση (HIGH) ή αλλιώς σε υψηλό δυναμικό. Έπειτα, εκτελείται ξανά η εντολή καθυστέρησης delay(2) προκαλώντας μια καθυστέρηση των 2msec με σκοπό την ολοκλήρωση της περιστροφής του άξονα του βηματικού κινητήρα. Ύστερα, εκτελείται η τέταρτη τετράδα εντολών digitalWrite(), οι οποίες θέτουν του ψηφιακούς ακροδέκτες του Arduino (GStepper_Pin1) και (GStepper_Pin4) σε κατάσταση (LOW) ή αλλιώς σε χαμηλό δυναμικό, ενώ οι ψηφιακοί ακροδέκτες (GStepper_Pin2) και (GStepper_Pin3) θέτονται σε κατάσταση (HIGH) ή αλλιώς σε υψηλό δυναμικό. Εν συνεχεία, πραγματοποιείται ξανά μια καθυστέρηση των 2msec εξαιτίας της εντολής delay(2) με την οποία ολοκληρώνεται το μπλοκ εντολών της συνάρτησης Clockwise_GStep_Motor ().

Όσον αφορά το μπλοκ εντολών της συνάρτησης Counterclockwise_GStep_Motor (), ακολουθείτε μια παρόμοια λογική ενεργοποίησης των ψηφιακών ακροδεκτών του Arduino με την διαφορά ότι ενεργοποιούνται ανάποδα σε σχέση με την προηγούμενη συνάρτηση Clockwise_GStep_Motor (). Πιο αναλυτικά, στην πρώτη τετράδα εντολών digitalWrite(), οι ψηφιακοί ακροδέκτες του Arduino (GStepper_Pin3) και (GStepper_Pin4) θέτονται κατάσταση (LOW) ή αλλιώς σε χαμηλό δυναμικό, ενώ οι ψηφιακοί ακροδέκτες (GStepper_Pin1) και (GStepper_Pin2) θέτονται σε κατάσταση (HIGH) ή αλλιώς σε υψηλό δυναμικό. Έπειτα, εκτελείτε μια μικρή καθυστέρηση των 2msec εξαιτίας της εντολής καθυστέρησης delay(2). Σκοπός της μικρής αυτής καθυστέρησης είναι να προλάβει να ολοκληρώσει ο βηματικός κινητήρας την περιστροφή του άξονά του. Στην δεύτερη τετράδα εντολών digitalWrite(), οι ψηφιακοί ακροδέκτες του Arduino (GStepper_Pin2) και (GStepper_Pin3) θέτονται κατάσταση (LOW) ή αλλιώς σε χαμηλό δυναμικό, ενώ οι ψηφιακοί ακροδέκτες (GStepper_Pin1) και (GStepper_Pin4) θέτονται σε κατάσταση (HIGH) ή αλλιώς σε υψηλό δυναμικό. Ύστερα, εκτελείτε ξανά μια μικρή καθυστέρηση των 2msec εξαιτίας της εντολής καθυστέρησης delay(2) ώστε να προλάβει ο βηματικός κινητήρας να ολοκληρώσει την περιστροφή του άξονά του. Στην συνέχεια, εκτελείται η τρίτη τετράδα εντολών digitalWrite(), οι οποίες θέτουν του ψηφιακούς ακροδέκτες του Arduino (GStepper_Pin1) και (GStepper_Pin2) σε κατάσταση (LOW) ή αλλιώς σε χαμηλό δυναμικό, ενώ οι ψηφιακοί ακροδέκτες (GStepper_Pin3) και (GStepper_Pin4) θέτονται σε κατάσταση (HIGH) ή αλλιώς σε υψηλό δυναμικό. Έπειτα, εκτελείται ξανά η εντολή καθυστέρησης delay(2) προκαλώντας μια καθυστέρηση των 2msec με σκοπό την ολοκλήρωση της περιστροφής του άξονα του βηματικού κινητήρα. Ύστερα, εκτελείται η τέταρτη τετράδα εντολών digitalWrite(),οι οποίες θέτουν του ψηφιακούς ακροδέκτες του Arduino (GStepper_Pin1) και (GStepper_Pin4) σε κατάσταση (LOW) ή αλλιώς σε χαμηλό δυναμικό, ενώ οι ψηφιακοί ακροδέκτες (GStepper_Pin2) και (GStepper_Pin3) θέτονται σε κατάσταση (HIGH) ή αλλιώς σε υψηλό δυναμικό. Τέλος, το μπλοκ εντολών της συνάρτησης

Counterclockwise_GStep_Motor () ολοκληρώνεται με τη εκτέλεση της εντολής καθυστέρησης delay(2), η οποία δημιουργεί μια καθυστέρηση των 2msec.

Πίνακας 3.14: Ο κώδικας προγραμματισμού των συναρτήσεων Clockwise_GStep_Motor () & Counterclockwise_GStep_Motor () [40].

Κώδικας προγραμματισμού (Μέρος 14ο)

```
void Clockwise_GStep_Motor () { // Function for clockwise movement of GStep_Motor
  digitalWrite(GStepper_Pin1,LOW); //Set GStepper_Pin1 LOW = 0
  digitalWrite(GStepper_Pin2,LOW); //Set GStepper_Pin2 LOW = 0
  digitalWrite(GStepper_Pin3,HIGH); //Set GStepper_Pin3 HIGH = 1
  digitalWrite(GStepper_Pin4, HIGH); //Set GStepper_Pin4 HIGH = 1
  delay(2); //Time delay 2 msec

  digitalWrite(GStepper_Pin1,HIGH); //Set GStepper_Pin1 HIGH = 1
  digitalWrite(GStepper_Pin2,LOW); //Set GStepper_Pin2 LOW = 0
  digitalWrite(GStepper_Pin3,LOW); //Set GStepper_Pin3 LOW = 0
  digitalWrite(GStepper_Pin4, HIGH); //Set GStepper_Pin4 HIGH = 1
  delay(2); //Time delay 2 msec

  digitalWrite(GStepper_Pin1,HIGH); //Set GStepper_Pin1 HIGH = 1
  digitalWrite(GStepper_Pin2,HIGH); //Set GStepper_Pin2 HIGH = 1
  digitalWrite(GStepper_Pin3,LOW); //Set GStepper_Pin3 LOW = 0
  digitalWrite(GStepper_Pin4, LOW); //Set GStepper_Pin4 LOW = 0
  delay(2); //Time delay 2 msec

  digitalWrite(GStepper_Pin1,LOW); //Set GStepper_Pin1 LOW = 0
  digitalWrite(GStepper_Pin2,HIGH); //Set GStepper_Pin2 HIGH = 1
  digitalWrite(GStepper_Pin3,HIGH); //Set GStepper_Pin3 HIGH = 1
  digitalWrite(GStepper_Pin4, LOW); //Set GStepper_Pin4 LOW = 0
  delay(2); //Time delay 2 msec
}

void Counterclockwise_GStep_Motor() { // Function for Counterclockwise movement of GStep_Motor
  digitalWrite(GStepper_Pin1, HIGH); //Set GStepper_Pin1 HIGH = 1
  digitalWrite(GStepper_Pin2, HIGH); //Set GStepper_Pin2 HIGH = 1
  digitalWrite(GStepper_Pin3, LOW); //Set GStepper_Pin1 LOW = 0
  digitalWrite(GStepper_Pin4, LOW); //Set GStepper_Pin1 LOW = 0
  delay(2); //Time delay 2 msec
```

```
  digitalWrite(GStepper_Pin1, HIGH); //Set GStepper_Pin1 HIGH = 1
  digitalWrite(GStepper_Pin2, LOW); //Set GStepper_Pin2 LOW = 0
  digitalWrite(GStepper_Pin3, LOW); //Set GStepper_Pin1 LOW = 0
  digitalWrite(GStepper_Pin4, HIGH); //Set GStepper_Pin1 HIGH = 1
  delay(2); //Time delay 2 msec

  digitalWrite(GStepper_Pin1, LOW); //Set GStepper_Pin1 LOW = 0
  digitalWrite(GStepper_Pin2, LOW); //Set GStepper_Pin2 LOW = 0
  digitalWrite(GStepper_Pin3, HIGH); //Set GStepper_Pin1 HIGH = 1
  digitalWrite(GStepper_Pin4, HIGH); //Set GStepper_Pin1 HIGH = 1
  delay(2); //Time delay 2 msec

  digitalWrite(GStepper_Pin1, LOW); //Set GStepper_Pin1 LOW = 0
  digitalWrite(GStepper_Pin2, HIGH); //Set GStepper_Pin2 HIGH = 1
  digitalWrite(GStepper_Pin3, HIGH); //Set GStepper_Pin1 HIGH = 1
  digitalWrite(GStepper_Pin4, LOW); //Set GStepper_Pin1 LOW = 0
  delay(2); //Time delay 2 msec
}
```

3.1.11 **Ανάλυση των συναρτήσεων Clockwise_DStep_Motor () & Counterclockwise_DStep_Motor () (Κώδικας προγραμματισμού Μέρος 15ο)**

Σε αυτή την ενότητα πραγματοποιείται ανάλυση των συναρτήσεων που είναι υπεύθυνες για τον έλεγχο της περιστροφή του βηματικού κινητήρα της χορδής Ρε (D). Ουσιαστικά, πραγματοποιείται ανάλυση της συνάρτησης Counterclockwise_DStep_Motor (), η οποία είναι υπεύθυνη για την αντί-ωρολογιακή περιστροφή του βηματικού κινητήρα της χορδής Ρε (D) καθώς και της συνάρτησης Clockwise_DStep_Motor (), η οποία είναι υπεύθυνη για την ωρολογιακή περιστροφή του κινητήρα της χορδής Ρε (D) (Πίνακας 3.15). Πιο συγκεκριμένα, οι συναρτήσεις αυτές καλούνται εντός του μπλοκ εντολών της συνάρτησης ButtonRead() με σκοπό τον έλεγχο του κλειδιού της χορδής Ρε (D) είτε κατά τον χειροκίνητο έλεγχο είτε κατά το κούρδισμα της συγκεκριμένης χορδής.

Αρχικά, εντός του μπλοκ εντολών της συνάρτησης Counterclockwise_DStep_Motor εκτελούνται κατάλληλα οι εντολές digitalWrite() οι οποίες αποσκοπούν στον έλεγχο των ψηφιακών εξόδων του Arduino MEGA 2560. Ακόμα, όπως έχει αναφερθεί στο Κεφάλαιο 2, κάθε ένας από τους τέσσερις μαγνήτες του βηματικού κινητήρα, ελέγχεται από μία ψηφιακή θύρα του Arduino, μέσω της πλακέτας οδήγησης του βηματικού κινητήρα. Επομένως, για την επίτευξη του ελέγχου των τεσσάρων μαγνητών του βηματικού κινητήρα, δημιουργούνται τετράδες εντολών digitalWrite(). Πιο συγκεκριμένα, στην πρώτη τετράδα εντολών digitalWrite(), οι ψηφιακοί ακροδέκτες του Arduino (DStepper_Pin1) και (DStepper_Pin2) θέτονται κατάσταση (LOW) ή αλλιώς σε χαμηλό δυναμικό, ενώ οι ψηφιακοί ακροδέκτες (DStepper_Pin3) και (DStepper_Pin4) θέτονται σε κατάσταση (HIGH) ή αλλιώς σε υψηλό δυναμικό. Στην συνέχεια, πραγματοποιείται μια

καθυστέρηση των 2msec εξαιτίας της εντολής καθυστέρησης delay(2). Η μικρή αυτή καθυστέρηση χρειάζεται για να ολοκληρώσει ο βηματικός κινητήρας την περιστροφή του άξονά του. Στην δεύτερη τετράδα εντολών digitalWrite(), οι ψηφιακοί ακροδέκτες του Arduino (DStepper_Pin2) και (DStepper_Pin3) θέτονται κατάσταση (LOW) ή αλλιώς σε χαμηλό δυναμικό, ενώ οι ψηφιακοί ακροδέκτες (DStepper_Pin1) και (DStepper_Pin4) θέτονται σε κατάσταση (HIGH) ή αλλιώς σε υψηλό δυναμικό. Έπειτα, εκτελείτε ξανά μια μικρή καθυστέρηση των 2msec εξαιτίας της εντολής καθυστέρησης delay(2) ώστε να προλάβει ο βηματικός κινητήρας να ολοκληρώσει την περιστροφή του άξονά του. Στην συνέχεια, εκτελείται η τρίτη τετράδα εντολών digitalWrite(), οι οποίες θέτουν του ψηφιακούς ακροδέκτες του Arduino (DStepper_Pin3) και (DStepper_Pin4) σε κατάσταση (LOW) ή αλλιώς σε χαμηλό δυναμικό, ενώ οι ψηφιακοί ακροδέκτες (DStepper_Pin1) και (DStepper_Pin2) θέτονται σε κατάσταση (HIGH) ή αλλιώς σε υψηλό δυναμικό. Ύστερα, εκτελείται ξανά η εντολή καθυστέρησης delay(2) προκαλώντας μια καθυστέρηση των 2msec με σκοπό την ολοκλήρωση της περιστροφής του άξονα του βηματικού κινητήρα. Έπειτα, εκτελείται η τέταρτη τετράδα εντολών digitalWrite(), οι οποίες θέτουν του ψηφιακούς ακροδέκτες του Arduino (DStepper_Pin1) και (DStepper_Pin4) σε κατάσταση (LOW) ή αλλιώς σε χαμηλό δυναμικό, ενώ οι ψηφιακοί ακροδέκτες (DStepper_Pin2) και (DStepper_Pin3) θέτονται σε κατάσταση (HIGH) ή αλλιώς σε υψηλό δυναμικό. Αμέσως μετά, πραγματοποιείται ξανά μια καθυστέρηση των 2msec εξαιτίας της εντολής delay(2) με την οποία ολοκληρώνεται το μπλοκ εντολών της συνάρτησης Counterclockwise_DStep_Motor ().

Από την άλλη πλευρά όμως, το μπλοκ εντολών της συνάρτησης Clockwise_DStep_Motor (), ακολουθεί μια ανάποδη σειρά ενεργοποίησης των ψηφιακών ακροδεκτών του Arduino, σε σχέση με την προηγούμενη συνάρτηση Counterclockwise_DStep_Motor (). Πιο συγκεκριμένα, στην πρώτη τετράδα εντολών digitalWrite(), οι ψηφιακοί ακροδέκτες του Arduino (DStepper_Pin3) και (DStepper_Pin4) θέτονται κατάσταση (LOW) ή αλλιώς σε χαμηλό δυναμικό, ενώ οι ψηφιακοί ακροδέκτες (DStepper_Pin1) και (DStepper_Pin2) θέτονται σε κατάσταση (HIGH) ή αλλιώς σε υψηλό δυναμικό. Εν συνεχεία, εκτελείτε μια μικρή καθυστέρηση των 2msec εξαιτίας της εντολής καθυστέρησης delay(2), ώστε ο βηματικός κινητήρας να ολοκληρώσει την περιστροφή του άξονά του. Στην δεύτερη τετράδα εντολών digitalWrite(), οι ψηφιακοί ακροδέκτες του Arduino (DStepper_Pin2) και (DStepper_Pin3) θέτονται κατάσταση (LOW) ή αλλιώς σε χαμηλό δυναμικό, ενώ οι ψηφιακοί ακροδέκτες (DStepper_Pin1) και (DStepper_Pin4) θέτονται σε κατάσταση (HIGH) ή αλλιώς σε υψηλό δυναμικό. Έπειτα, εκτελείτε ξανά μια μικρή καθυστέρηση των 2msec εξαιτίας της εντολής καθυστέρησης delay(2) ώστε να προλάβει ο βηματικός κινητήρας να ολοκληρώσει την περιστροφή του άξονά του. Ύστερα, εκτελείται η τρίτη τετράδα εντολών digitalWrite(), οι οποίες θέτουν του ψηφιακούς ακροδέκτες του Arduino (DStepper_Pin1) και (DStepper_Pin2) σε κατάσταση (LOW) ή αλλιώς σε χαμηλό δυναμικό, ενώ οι ψηφιακοί ακροδέκτες (DStepper_Pin3) και (DStepper_Pin4) θέτονται σε κατάσταση (HIGH) ή αλλιώς σε υψηλό δυναμικό. Έπειτα, εκτελείται ξανά η εντολή καθυστέρησης delay(2) προκαλώντας μια καθυστέρηση των 2msec με σκοπό την ολοκλήρωση της περιστροφής του άξονα του βηματικού κινητήρα. Στην συνέχεια, εκτελείται η τέταρτη τετράδα εντολών digitalWrite(), οι οποίες θέτουν του ψηφιακούς ακροδέκτες του Arduino (DStepper_Pin1) και (DStepper_Pin4) σε κατάσταση (LOW) ή αλλιώς σε χαμηλό δυναμικό, ενώ οι ψηφιακοί ακροδέκτες (DStepper_Pin2) και (DStepper_Pin3) θέτονται σε κατάσταση (HIGH) ή αλλιώς σε υψηλό δυναμικό.

Ολοκληρώνοντας την ανάλυση του μπλοκ εντολών της συνάρτησης Clockwise_DStep_Motor (), πραγματοποιείται εκτέλεση της εντολής καθυστέρησης delay(2), η οποία δημιουργεί μια καθυστέρηση των 2msec η οποία είναι αναγκαία για να ολοκληρώσει ο βηματικός κινητήρας την περιστροφή του άξονά του.

Πίνακας 3.15: Ο κώδικας προγραμματισμού των συναρτήσεων Clockwise_DStep_Motor () & Counterclockwise_DStep_Motor () [41].

Κώδικας προγραμματισμού (Μέρος 15ο)

```
void Counterclockwise_DStep_Motor () { // Function for clockwise movement of DStep_Motor
  digitalWrite(DStepper_Pin1,LOW); //Set DStepper_Pin1 LOW = 0
  digitalWrite(DStepper_Pin2,LOW); //Set DStepper_Pin2 LOW = 0
  digitalWrite(DStepper_Pin3,HIGH); //Set DStepper_Pin3 HIGH = 1
  digitalWrite(DStepper_Pin4, HIGH); //Set DStepper_Pin4 HIGH = 1
  delay(2); //Time delay 2 msec

  digitalWrite(DStepper_Pin1,HIGH); //Set DStepper_Pin1 HIGH = 1
  digitalWrite(DStepper_Pin2,LOW); //Set DStepper_Pin2 LOW = 0
  digitalWrite(DStepper_Pin3,LOW); //Set DStepper_Pin3 LOW = 0
  digitalWrite(DStepper_Pin4, HIGH); //Set DStepper_Pin4 HIGH = 1
  delay(2); //Time delay 2 msec

  digitalWrite(DStepper_Pin1,HIGH); //Set DStepper_Pin1 HIGH = 1
  digitalWrite(DStepper_Pin2,HIGH); //Set DStepper_Pin2 HIGH = 1
  digitalWrite(DStepper_Pin3,LOW); //Set DStepper_Pin3 LOW = 0
  digitalWrite(DStepper_Pin4, LOW); //Set DStepper_Pin4 LOW = 0
  delay(2); //Time delay 2 msec

  digitalWrite(DStepper_Pin1,LOW); //Set DStepper_Pin1 LOW = 0
  digitalWrite(DStepper_Pin2,HIGH); //Set DStepper_Pin2 HIGH = 1
  digitalWrite(DStepper_Pin3,HIGH); //Set DStepper_Pin3 HIGH = 1
  digitalWrite(DStepper_Pin4, LOW); //Set DStepper_Pin4 LOW = 0
  delay(2); //Time delay 2 msec
}

void Clockwise_DStep_Motor () { // Function for Counterclockwise movement of DStep_Motor
  digitalWrite(DStepper_Pin1, HIGH); //Set DStepper_Pin1 HIGH = 1
  digitalWrite(DStepper_Pin2, HIGH); //Set DStepper_Pin2 HIGH = 1
  digitalWrite(DStepper_Pin3, LOW); //Set DStepper_Pin1 LOW = 0
```

```
  digitalWrite(DStepper_Pin4, LOW); //Set DStepper_Pin1 LOW = 0
  delay(2); //Time delay 2 msec

  digitalWrite(DStepper_Pin1, HIGH); //Set DStepper_Pin1 HIGH = 1
  digitalWrite(DStepper_Pin2, LOW); //Set DStepper_Pin2 LOW = 0
  digitalWrite(DStepper_Pin3, LOW); //Set DStepper_Pin1 LOW = 0
  digitalWrite(DStepper_Pin4, HIGH); //Set DStepper_Pin1 HIGH = 1
  delay(2); //Time delay 2 msec

  digitalWrite(DStepper_Pin1, LOW); //Set DStepper_Pin1 LOW = 0
  digitalWrite(DStepper_Pin2, LOW); //Set DStepper_Pin2 LOW = 0
  digitalWrite(DStepper_Pin3, HIGH); //Set DStepper_Pin1 HIGH = 1
  digitalWrite(DStepper_Pin4, HIGH); //Set DStepper_Pin1 HIGH = 1
  delay(2); //Time delay 2 msec

  digitalWrite(DStepper_Pin1, LOW); //Set DStepper_Pin1 LOW = 0
  digitalWrite(DStepper_Pin2, HIGH); //Set DStepper_Pin2 HIGH = 1
  digitalWrite(DStepper_Pin3, HIGH); //Set DStepper_Pin1 HIGH = 1
  digitalWrite(DStepper_Pin4, LOW); //Set DStepper_Pin1 LOW = 0
  delay(2); //Time delay 2 msec
}
```

3.1.12 Ανάλυση των συναρτήσεων Clockwise_AStep_Motor () & Counterclockwise_AStep_Motor () (Κώδικας προγραμματισμού Μέρος 16ο)

Σε αυτή την ενότητα πραγματοποιείται ανάλυση των συναρτήσεων που είναι υπεύθυνες για τον έλεγχο της περιστροφή του βηματικού κινητήρα της χορδής Λα (Α). Πιο συγκεκριμένα, πραγματοποιείται ανάλυση της συνάρτησης Counterclockwise_AStep_Motor (), η οποία είναι υπεύθυνη για την αντί-ωρολογιακή περιστροφή του βηματικού κινητήρα της χορδής Λα (Α) καθώς και της συνάρτησης Clockwise_AStep_Motor (), η οποία είναι υπεύθυνη για την ωρολογιακή περιστροφή του κινητήρα της χορδής Λα (Α) (Πίνακας 3.16). Ουσιαστικά, οι συναρτήσεις αυτές καλούνται εντός του μπλοκ εντολών της συνάρτησης ButtonRead() με σκοπό τον έλεγχο του κλειδιού της χορδής Λα (Α) είτε κατά χειροκίνητο έλεγχο είτε κατά το κούρδισμα της συγκεκριμένης χορδής.

Αρχικά, εντός του μπλοκ εντολών της συνάρτησης Counterclockwise_AStep_Motor () εκτελούνται εντολές οι οποίες αποσκοπούν στον έλεγχο των ψηφιακών εξόδων του Arduino MEGA 2560. Αυτό επιτυγχάνεται με την χρήση της εντολής digitalWrite(), η οποία θα θέσει την αντίστοιχη ψηφιακή θύρα είτε σε υψηλό δυναμικό είτε σε χαμηλό δυναμικό. Επίσης, όπως έχει αναφερθεί στο Κεφάλαιο 2, ο βηματικός κινητήρας,

αποτελείται από τέσσερις μαγνήτες όπου ο κάθε ένας ελέγχεται από μία ψηφιακή θύρα του Arduino, μέσω της πλακέτας οδήγησης του βηματικού κινητήρα. Επομένως, δημιουργούνται τετράδες εντολών digitalWrite(), ώστε να επιτευχθεί ο έλεγχος και των τεσσάρων μαγνητών του βηματικού κινητήρα. Πιο αναλυτικά, στην πρώτη τετράδα εντολών digitalWrite(), οι ψηφιακοί ακροδέκτες του Arduino (AStepper_Pin1) και (AStepper_Pin2) θέτονται κατάσταση (LOW) ή αλλιώς σε χαμηλό δυναμικό, ενώ οι ψηφιακοί ακροδέκτες (AStepper_Pin3) και (AStepper_Pin4) θέτονται σε κατάσταση (HIGH) ή αλλιώς σε υψηλό δυναμικό. Έπειτα, εκτελείτε μια μικρή καθυστέρηση των 2msec εξαιτίας της εντολής καθυστέρησης delay(2). Σκοπός της μικρής αυτής καθυστέρησης είναι να προλάβει να ολοκληρώσει ο βηματικός κινητήρας την περιστροφή του άξονά του. Στην δεύτερη τετράδα εντολών digitalWrite(), οι ψηφιακοί ακροδέκτες του Arduino (AStepper_Pin2) και (AStepper_Pin3) θέτονται κατάσταση (LOW) ή αλλιώς σε χαμηλό δυναμικό, ενώ οι ψηφιακοί ακροδέκτες (AStepper_Pin1) και (AStepper_Pin4) θέτονται σε κατάσταση (HIGH) ή αλλιώς σε υψηλό δυναμικό. Ύστερα, εκτελείτε ξανά μια μικρή καθυστέρηση των 2msec εξαιτίας της εντολής καθυστέρησης delay(2) ώστε να προλάβει ο βηματικός κινητήρας να ολοκληρώσει την περιστροφή του άξονά του. Στην συνέχεια, εκτελείται η τρίτη τετράδα εντολών digitalWrite(), οι οποίες θέτουν του ψηφιακούς ακροδέκτες του Arduino (AStepper_Pin3) και (AStepper_Pin4) σε κατάσταση (LOW) ή αλλιώς σε χαμηλό δυναμικό, ενώ οι ψηφιακοί ακροδέκτες (AStepper_Pin1) και (AStepper_Pin2) θέτονται σε κατάσταση (HIGH) ή αλλιώς σε υψηλό δυναμικό. Έπειτα, εκτελείται ξανά η εντολή καθυστέρησης delay(2) προκαλώντας μια καθυστέρηση των 2msec με σκοπό την ολοκλήρωση της περιστροφής του άξονα του βηματικού κινητήρα. Ύστερα, εκτελείται η τέταρτη τετράδα εντολών digitalWrite(), οι οποίες θέτουν του ψηφιακούς ακροδέκτες του Arduino (AStepper_Pin1) και (AStepper_Pin4) σε κατάσταση (LOW) ή αλλιώς σε χαμηλό δυναμικό, ενώ οι ψηφιακοί ακροδέκτες (AStepper_Pin2) και (AStepper_Pin3) θέτονται σε κατάσταση (HIGH) ή αλλιώς σε υψηλό δυναμικό. Εν συνεχεία, πραγματοποιείται ξανά μια καθυστέρηση των 2msec εξαιτίας της εντολής delay(2) με την οποία ολοκληρώνεται το μπλοκ εντολών της συνάρτησης Counterclockwise_AStep_Motor ().

Όσον αφορά το μπλοκ εντολών της συνάρτησης Clockwise_AStep_Motor (), ακολουθείτε μια παρόμοια λογική ενεργοποίησης των ψηφιακών ακροδεκτών του Arduino με την διαφορά ότι ενεργοποιούνται ανάποδα σε σχέση με την προηγούμενη συνάρτηση Counterclockwise_AStep_Motor (). Πιο αναλυτικά, στην πρώτη τετράδα εντολών digitalWrite(), οι ψηφιακοί ακροδέκτες του Arduino (AStepper_Pin3) και (AStepper_Pin4) θέτονται κατάσταση (LOW) ή αλλιώς σε χαμηλό δυναμικό, ενώ οι ψηφιακοί ακροδέκτες (AStepper_Pin1) και (AStepper_Pin2) θέτονται σε κατάσταση (HIGH) ή αλλιώς σε υψηλό δυναμικό. Έπειτα, εκτελείτε μια μικρή καθυστέρηση των 2msec εξαιτίας της εντολής καθυστέρησης delay(2). Σκοπός της μικρής αυτής καθυστέρησης είναι να προλάβει να ολοκληρώσει ο βηματικός κινητήρας την περιστροφή του άξονά του. Στην δεύτερη τετράδα εντολών digitalWrite(), οι ψηφιακοί ακροδέκτες του Arduino (AStepper_Pin2) και (AStepper_Pin3) θέτονται κατάσταση (LOW) ή αλλιώς σε χαμηλό δυναμικό, ενώ οι ψηφιακοί ακροδέκτες (AStepper_Pin1) και (AStepper_Pin4) θέτονται σε κατάσταση (HIGH) ή αλλιώς σε υψηλό δυναμικό. Ύστερα, εκτελείτε ξανά μια μικρή καθυστέρηση των 2msec εξαιτίας της εντολής καθυστέρησης delay(2) ώστε να προλάβει ο βηματικός κινητήρας να ολοκληρώσει την περιστροφή του άξονά του. Στην συνέχεια, εκτελείται η τρίτη τετράδα εντολών digitalWrite(), οι οποίες θέτουν του ψηφιακούς ακροδέκτες του Arduino (AStepper_Pin1) και (AStepper_Pin2) σε κατάσταση

(LOW) ή αλλιώς σε χαμηλό δυναμικό, ενώ οι ψηφιακοί ακροδέκτες (AStepper_Pin3) και (AStepper_Pin4) θέτονται σε κατάσταση (HIGH) ή αλλιώς σε υψηλό δυναμικό. Έπειτα, εκτελείται ξανά η εντολή καθυστέρησης delay(2) προκαλώντας μια καθυστέρηση των 2msec με σκοπό την ολοκλήρωση της περιστροφής του άξονα του βηματικού κινητήρα. Ύστερα, εκτελείται η τέταρτη τετράδα εντολών digitalWrite(),οι οποίες θέτουν του ψηφιακούς ακροδέκτες του Arduino (AStepper_Pin1) και (AStepper_Pin4) σε κατάσταση (LOW) ή αλλιώς σε χαμηλό δυναμικό, ενώ οι ψηφιακοί ακροδέκτες (AStepper_Pin2) και (AStepper_Pin3) θέτονται σε κατάσταση (HIGH) ή αλλιώς σε υψηλό δυναμικό. Τέλος, το μπλοκ εντολών της συνάρτησης Clockwise_AStep_Motor () ολοκληρώνεται με τη εκτέλεση της εντολής καθυστέρησης delay(2), η οποία δημιουργεί μια καθυστέρηση των 2msec.

Πίνακας 3.16: Ο κώδικας προγραμματισμού των συναρτήσεων Clockwise_AStep_Motor () & Counterclockwise_AStep_Motor () [42].

Κώδικας προγραμματισμού (Μέρος 16ο)

```
void Counterclockwise_AStep_Motor () { // Function for clockwise movement of AStep_Motor
  digitalWrite(AStepper_Pin1,LOW); //Set AStepper_Pin1 LOW = 0
  digitalWrite(AStepper_Pin2,LOW); //Set AStepper_Pin2 LOW = 0
  digitalWrite(AStepper_Pin3,HIGH); //Set AStepper_Pin3 HIGH = 1
  digitalWrite(AStepper_Pin4, HIGH); //Set AStepper_Pin4 HIGH = 1
  delay(2); //Time delay 2 msec

  digitalWrite(AStepper_Pin1,HIGH); //Set AStepper_Pin1 HIGH = 1
  digitalWrite(AStepper_Pin2,LOW); //Set AStepper_Pin2 LOW = 0
  digitalWrite(AStepper_Pin3,LOW); //Set AStepper_Pin3 LOW = 0
  digitalWrite(AStepper_Pin4, HIGH); //Set AStepper_Pin4 HIGH = 1
  delay(2); //Time delay 2 msec

  digitalWrite(AStepper_Pin1,HIGH); //Set AStepper_Pin1 HIGH = 1
  digitalWrite(AStepper_Pin2,HIGH); //Set AStepper_Pin2 HIGH = 1
  digitalWrite(AStepper_Pin3,LOW); //Set AStepper_Pin3 LOW = 0
  digitalWrite(AStepper_Pin4, LOW); //Set AStepper_Pin4 LOW = 0
  delay(2); //Time delay 2 msec

  digitalWrite(AStepper_Pin1,LOW); //Set AStepper_Pin1 LOW = 0
  digitalWrite(AStepper_Pin2,HIGH); //Set AStepper_Pin2 HIGH = 1
  digitalWrite(AStepper_Pin3,HIGH); //Set AStepper_Pin3 HIGH = 1
  digitalWrite(AStepper_Pin4, LOW); //Set AStepper_Pin4 LOW = 0
  delay(2); //Time delay 2 msec
}
```

```
void Clockwise_AStep_Motor () { // Function for Counterclockwise movement of AStep_Motor
  digitalWrite(AStepper_Pin1, HIGH); //Set AStepper_Pin1 HIGH = 1
  digitalWrite(AStepper_Pin2, HIGH); //Set AStepper_Pin2 HIGH = 1
  digitalWrite(AStepper_Pin3, LOW); //Set AStepper_Pin1 LOW = 0
  digitalWrite(AStepper_Pin4, LOW); //Set AStepper_Pin1 LOW = 0
  delay(2); //Time delay 2 msec

  digitalWrite(AStepper_Pin1, HIGH); //Set AStepper_Pin1 HIGH = 1
  digitalWrite(AStepper_Pin2, LOW); //Set AStepper_Pin2 LOW = 0
  digitalWrite(AStepper_Pin3, LOW); //Set AStepper_Pin1 LOW = 0
  digitalWrite(AStepper_Pin4, HIGH); //Set AStepper_Pin1 HIGH = 1
  delay(2); //Time delay 2 msec

  digitalWrite(AStepper_Pin1, LOW); //Set AStepper_Pin1 LOW = 0
  digitalWrite(AStepper_Pin2, LOW); //Set AStepper_Pin2 LOW = 0
  digitalWrite(AStepper_Pin3, HIGH); //Set AStepper_Pin1 HIGH = 1
  digitalWrite(AStepper_Pin4, HIGH); //Set AStepper_Pin1 HIGH = 1
  delay(2); //Time delay 2 msec

  digitalWrite(AStepper_Pin1, LOW); //Set AStepper_Pin1 LOW = 0
  digitalWrite(AStepper_Pin2, HIGH); //Set AStepper_Pin2 HIGH = 1
  digitalWrite(AStepper_Pin3, HIGH); //Set AStepper_Pin1 HIGH = 1
  digitalWrite(AStepper_Pin4, LOW); //Set AStepper_Pin1 LOW = 0
  delay(2); //Time delay 2 msec
}
```

3.1.13 **Ανάλυση των συναρτήσεων Clockwise_EStep_Motor () & Counterclockwise_EStep_Motor () (Κώδικας προγραμματισμού Μέρος 17ο)**

Σε αυτή την ενότητα πραγματοποιείται ανάλυση των συναρτήσεων που είναι υπεύθυνες για τον έλεγχο της περιστροφή του βηματικού κινητήρα της χορδής Μι μπάσο (E). Ουσιαστικά, πραγματοποιείται ανάλυση της συνάρτησης Counterclockwise_EStep_Motor (), η οποία είναι υπεύθυνη για την αντί-ωρολογιακή περιστροφή του βηματικού κινητήρα της χορδής Μι μπάσο (E) καθώς και της συνάρτησης Clockwise_EStep_Motor (), η οποία είναι υπεύθυνη για την ωρολογιακή περιστροφή του κινητήρα της χορδής Μι μπάσο (E) (Πίνακας 3.17). Πιο συγκεκριμένα, οι συναρτήσεις αυτές καλούνται εντός του μπλοκ εντολών της συνάρτησης ButtonRead() με σκοπό τον

έλεγχο του κλειδιού της χορδής Μι μπάσο (Ε) είτε κατά τον χειροκίνητο έλεγχο είτε κατά το κούρδισμα της συγκεκριμένης χορδής.

Αρχικά, εντός του μπλοκ εντολών της συνάρτησης Counterclockwise_EStep_Motor εκτελούνται κατάλληλα οι εντολές digitalWrite() οι οποίες αποσκοπούν στον έλεγχο των ψηφιακών εξόδων του Arduino MEGA 2560. Ακόμα, όπως έχει αναφερθεί στο Κεφάλαιο 2, κάθε ένας από τους τέσσερις μαγνήτες του βηματικού κινητήρα, ελέγχεται από μία ψηφιακή θύρα του Arduino, μέσω της πλακέτας οδήγησης του βηματικού κινητήρα. Επομένως, για την επίτευξη του ελέγχου των τεσσάρων μαγνητών του βηματικού κινητήρα, δημιουργούνται τετράδες εντολών digitalWrite(). Πιο συγκεκριμένα, στην πρώτη τετράδα εντολών digitalWrite(), οι ψηφιακοί ακροδέκτες του Arduino (EStepper_Pin1) και (EStepper_Pin2) θέτονται κατάσταση (LOW) ή αλλιώς σε χαμηλό δυναμικό, ενώ οι ψηφιακοί ακροδέκτες (EStepper_Pin3) και (EStepper_Pin4) θέτονται σε κατάσταση (HIGH) ή αλλιώς σε υψηλό δυναμικό. Στην συνέχεια, πραγματοποιείται μια καθυστέρηση των 2msec εξαιτίας της εντολής καθυστέρησης delay(2). Η μικρή αυτή καθυστέρηση χρειάζεται για να ολοκληρώσει ο βηματικός κινητήρας την περιστροφή του άξονά του. Στην δεύτερη τετράδα εντολών digitalWrite(), οι ψηφιακοί ακροδέκτες του Arduino (EStepper_Pin2) και (EStepper_Pin3) θέτονται κατάσταση (LOW) ή αλλιώς σε χαμηλό δυναμικό, ενώ οι ψηφιακοί ακροδέκτες (EStepper_Pin1) και (EStepper_Pin4) θέτονται σε κατάσταση (HIGH) ή αλλιώς σε υψηλό δυναμικό. Έπειτα, εκτελείτε ξανά μια μικρή καθυστέρηση των 2msec εξαιτίας της εντολής καθυστέρησης delay(2) ώστε να προλάβει ο βηματικός κινητήρας να ολοκληρώσει την περιστροφή του άξονά του. Στην συνέχεια, εκτελείται η τρίτη τετράδα εντολών digitalWrite(), οι οποίες θέτουν του ψηφιακούς ακροδέκτες του Arduino (EStepper_Pin3) και (EStepper_Pin4) σε κατάσταση (LOW) ή αλλιώς σε χαμηλό δυναμικό, ενώ οι ψηφιακοί ακροδέκτες (EStepper_Pin1) και (EStepper_Pin2) θέτονται σε κατάσταση (HIGH) ή αλλιώς σε υψηλό δυναμικό. Ύστερα, εκτελείται ξανά η εντολή καθυστέρησης delay(2) προκαλώντας μια καθυστέρηση των 2msec με σκοπό την ολοκλήρωση της περιστροφής του άξονα του βηματικού κινητήρα. Έπειτα, εκτελείται η τέταρτη τετράδα εντολών digitalWrite(), οι οποίες θέτουν του ψηφιακούς ακροδέκτες του Arduino (EStepper_Pin1) και (EStepper_Pin4) σε κατάσταση (LOW) ή αλλιώς σε χαμηλό δυναμικό, ενώ οι ψηφιακοί ακροδέκτες (EStepper_Pin2) και (EStepper_Pin3) θέτονται σε κατάσταση (HIGH) ή αλλιώς σε υψηλό δυναμικό. Αμέσως μετά, πραγματοποιείται ξανά μια καθυστέρηση των 2msec εξαιτίας της εντολής delay(2) με την οποία ολοκληρώνεται το μπλοκ εντολών της συνάρτησης Counterclockwise_EStep_Motor ().

Από την άλλη πλευρά όμως, το μπλοκ εντολών της συνάρτησης Clockwise_EStep_Motor (), ακολουθεί μια ανάποδη σειρά ενεργοποίησης των ψηφιακών ακροδεκτών του Arduino, σε σχέση με την προηγούμενη συνάρτηση Counterclockwise_EStep_Motor (). Πιο συγκεκριμένα, στην πρώτη τετράδα εντολών digitalWrite(), οι ψηφιακοί ακροδέκτες του Arduino (EStepper_Pin3) και (EStepper_Pin4) θέτονται κατάσταση (LOW) ή αλλιώς σε χαμηλό δυναμικό, ενώ οι ψηφιακοί ακροδέκτες (EStepper_Pin1) και (EStepper_Pin2) θέτονται σε κατάσταση (HIGH) ή αλλιώς σε υψηλό δυναμικό. Εν συνεχεία, εκτελείτε μια μικρή καθυστέρηση των 2msec εξαιτίας της εντολής καθυστέρησης delay(2), ώστε ο βηματικός κινητήρας να ολοκληρώσει την περιστροφή του άξονά του. Στην δεύτερη τετράδα εντολών digitalWrite(), οι ψηφιακοί ακροδέκτες του Arduino (EStepper_Pin2) και (EStepper_Pin3) θέτονται κατάσταση (LOW) ή αλλιώς σε χαμηλό δυναμικό, ενώ οι ψηφιακοί ακροδέκτες (EStepper_Pin1) και (EStepper_Pin4)

θέτονται σε κατάσταση (HIGH) ή αλλιώς σε υψηλό δυναμικό. Έπειτα, εκτελείτε ξανά μια μικρή καθυστέρηση των 2msec εξαιτίας της εντολής καθυστέρησης delay(2) ώστε να προλάβει ο βηματικός κινητήρας να ολοκληρώσει την περιστροφή του άξονά του. Ύστερα, εκτελείται η τρίτη τετράδα εντολών digitalWrite(), οι οποίες θέτουν του ψηφιακούς ακροδέκτες του Arduino (EStepper_Pin1) και (EStepper_Pin2) σε κατάσταση (LOW) ή αλλιώς σε χαμηλό δυναμικό, ενώ οι ψηφιακοί ακροδέκτες (EStepper_Pin3) και (EStepper_Pin4) θέτονται σε κατάσταση (HIGH) ή αλλιώς σε υψηλό δυναμικό. Έπειτα, εκτελείται ξανά η εντολή καθυστέρησης delay(2) προκαλώντας μια καθυστέρηση των 2msec με σκοπό την ολοκλήρωση της περιστροφής του άξονα του βηματικού κινητήρα. Στην συνέχεια, εκτελείται η τέταρτη τετράδα εντολών digitalWrite(), οι οποίες θέτουν του ψηφιακούς ακροδέκτες του Arduino (EStepper_Pin1) και (EStepper_Pin4) σε κατάσταση (LOW) ή αλλιώς σε χαμηλό δυναμικό, ενώ οι ψηφιακοί ακροδέκτες (EStepper_Pin2) και (EStepper_Pin3) θέτονται σε κατάσταση (HIGH) ή αλλιώς σε υψηλό δυναμικό. Ολοκληρώνοντας την ανάλυση του μπλοκ εντολών της συνάρτησης Clockwise_EStep_Motor (), πραγματοποιείται εκτέλεση της εντολής καθυστέρησης delay(2), η οποία δημιουργεί μια καθυστέρηση των 2msec η οποία είναι αναγκαία για να ολοκληρώσει ο βηματικός κινητήρας την περιστροφή του άξονά του.

Πίνακας 3.17: Ο κώδικας προγραμματισμού των συναρτήσεων Clockwise_EStep_Motor () & Counterclockwise_EStep_Motor () [43].

Κώδικας προγραμματισμού (Μέρος 17ο)

```
void Counterclockwise_EStep_Motor () { // Function for clockwise movement of EStep_Motor
  digitalWrite(EStepper_Pin1,LOW); //Set EStepper_Pin1 LOW = 0
  digitalWrite(EStepper_Pin2,LOW); //Set EStepper_Pin2 LOW = 0
  digitalWrite(EStepper_Pin3,HIGH); //Set EStepper_Pin3 HIGH = 1
  digitalWrite(EStepper_Pin4, HIGH); //Set EStepper_Pin4 HIGH = 1
  delay(2); //Time delay 2 msec

  digitalWrite(EStepper_Pin1,HIGH); //Set EStepper_Pin1 HIGH = 1
  digitalWrite(EStepper_Pin2,LOW); //Set EStepper_Pin2 LOW = 0
  digitalWrite(EStepper_Pin3,LOW); //Set EStepper_Pin3 LOW = 0
  digitalWrite(EStepper_Pin4, HIGH); //Set EStepper_Pin4 HIGH = 1
  delay(2); //Time delay 2 msec

  digitalWrite(EStepper_Pin1,HIGH); //Set EStepper_Pin1 HIGH = 1
  digitalWrite(EStepper_Pin2,HIGH); //Set EStepper_Pin2 HIGH = 1
  digitalWrite(EStepper_Pin3,LOW); //Set EStepper_Pin3 LOW = 0
  digitalWrite(EStepper_Pin4, LOW); //Set EStepper_Pin4 LOW = 0
  delay(2); //Time delay 2 msec

  digitalWrite(EStepper_Pin1,LOW); //Set EStepper_Pin1 LOW = 0
```

```
  digitalWrite(EStepper_Pin2,HIGH); //Set EStepper_Pin2 HIGH = 1
  digitalWrite(EStepper_Pin3,HIGH); //Set EStepper_Pin3 HIGH = 1
  digitalWrite(EStepper_Pin4, LOW); //Set EStepper_Pin4 LOW = 0
  delay(2); //Time delay 2 msec
}

void Clockwise_EStep_Motor (){ // Function for Counterclockwise movement of EStep_Motor
  digitalWrite(EStepper_Pin1, HIGH); //Set EStepper_Pin1 HIGH = 1
  digitalWrite(EStepper_Pin2, HIGH); //Set EStepper_Pin2 HIGH = 1
  digitalWrite(EStepper_Pin3, LOW); //Set EStepper_Pin1 LOW = 0
  digitalWrite(EStepper_Pin4, LOW); //Set EStepper_Pin1 LOW = 0
  delay(2); //Time delay 2 msec

  digitalWrite(EStepper_Pin1, HIGH); //Set EStepper_Pin1 HIGH = 1
  digitalWrite(EStepper_Pin2, LOW); //Set EStepper_Pin2 LOW = 0
  digitalWrite(EStepper_Pin3, LOW); //Set EStepper_Pin1 LOW = 0
  digitalWrite(EStepper_Pin4, HIGH); //Set EStepper_Pin1 HIGH = 1
  delay(2); //Time delay 2 msec

  digitalWrite(EStepper_Pin1, LOW); //Set EStepper_Pin1 LOW = 0
  digitalWrite(EStepper_Pin2, LOW); //Set EStepper_Pin2 LOW = 0
  digitalWrite(EStepper_Pin3, HIGH); //Set EStepper_Pin1 HIGH = 1
  digitalWrite(EStepper_Pin4, HIGH); //Set EStepper_Pin1 HIGH = 1
  delay(2); //Time delay 2 msec

  digitalWrite(EStepper_Pin1, LOW); //Set EStepper_Pin1 LOW = 0
  digitalWrite(EStepper_Pin2, HIGH); //Set EStepper_Pin2 HIGH = 1
  digitalWrite(EStepper_Pin3, HIGH); //Set EStepper_Pin1 HIGH = 1
  digitalWrite(EStepper_Pin4, LOW); //Set EStepper_Pin1 LOW = 0
  delay(2); //Time delay 2 msec
}
```

3.1.14 **Ανάλυση των συναρτήσεων eStep_Motor_Stop (), BStep_Motor_Stop (), GStep_Motor_Stop (), DStep_Motor_Stop (), AStep_Motor_Stop () & EStep_Motor_Stop () (Κώδικας προγραμματισμού Μέρος 18ο)**

Σε αυτήν την ενότητα, πραγματοποιείται ανάλυση έξι συναρτήσεων απλών μεν αλλά σημαντικών δε για το σύστημα αυτομάτου ελέγχου για τον συντονισμό των χορδών της κιθάρας. Ουσιαστικά, αυτές οι συναρτήσεις αφορούν την απενεργοποίηση των βηματικών κινητήρων ώστε να εξοικονομηθεί η ενέργεια των μπαταριών. Πιο συγκεκριμένα, όταν ο

άξονας ολοκληρώσει την περιστροφή του, τότε οι μαγνήτες του δέχονται ακόμα κάποια τάση εξαιτίας των ψηφιακών ακροδεκτών που βρίσκονται σε υψηλό δυναμικό. Αυτό έχει ως αποτέλεσμα να σπαταλείται η ενέργεια των μπαταριών χωρίς ουσιαστικά να πραγματοποιείται κάποια περιστροφή του άξονα. Το πρόβλημα αυτό, αντιμετωπίζεται θέτοντας όλους τους ψηφιακούς ακροδέκτες εξόδου του Arduino που σχετίζονται με τον έλεγχο των βηματικών κινητήρων σε χαμηλό δυναμικό (Πίνακας 3.18).

Αρχικά, η συνάρτηση eStep_Motor_Stop () αναλαμβάνει την απενεργοποίηση του βηματικού κινητήρα που σχετίζεται με τον έλεγχο της περιστροφής του κλειδιού της χορδής Μι καντίνι (e). Πιο συγκεκριμένα, εκτελείτε μία τετράδα εντολών digitalWrite(), οι οποίες θέτουν τους ακροδέκτες (eStepper_Pin1, eStepper_Pin2, eStepper_Pin3 & eStepper_Pin4) σε κατάσταση (LOW) ή αλλιώς σε χαμηλό δυναμικό. Με αυτόν τον τρόπο, σταματά η πλακέτα οδήγησης του βηματικού κινητήρα της χορδής Μι καντίνι (e) να παρέχει τροφοδοσία στους μαγνήτες. Έπειτα, το μπλοκ εντολών της συνάρτησης eStep_Motor_Stop (), ολοκληρώνεται με την εκτέλεση της εντολής delay(2), η οποία πραγματοποιεί μια καθυστέρηση των 2msec.

Εν συνεχεία, η συνάρτηση BStep_Motor_Stop () αναλαμβάνει την απενεργοποίηση του βηματικού κινητήρα που σχετίζεται με τον έλεγχο της περιστροφής του κλειδιού της χορδής Σι (B). Πιο αναλυτικά, όπως στην προηγούμενη συνάρτηση, έτσι και εδώ, εκτελείτε μία τετράδα εντολών digitalWrite(), οι οποίες θέτουν τους ακροδέκτες (BStepper_Pin1, BStepper_Pin2, BStepper_Pin3 & BStepper_Pin4) σε κατάσταση (LOW) ή αλλιώς σε χαμηλό δυναμικό. Έτσι, παύει η πλακέτα οδήγησης του βηματικού κινητήρα της χορδής Σι (B) να παρέχει τροφοδοσία στους μαγνήτες του. Επίσης, το μπλοκ εντολών της συνάρτησης BStep_Motor_Stop () ολοκληρώνεται με την εκτέλεση της εντολής καθυστέρησης delay(2), η οποία δημιουργεί μια καθυστέρηση των 2msec.

Στην περίπτωση της συνάρτησης GStep_Motor_Stop (), πραγματοποιεί κάτι αντίστοιχο με τις προηγούμενες συναρτήσεις. Ουσιαστικά, η συγκεκριμένη συνάρτηση αναλαμβάνει την απενεργοποίηση του βηματικού κινητήρα που σχετίζεται με τον έλεγχο της περιστροφής του κλειδιού της χορδής Σολ (G). Πιο συγκεκριμένα, εκτελείτε μία τετράδα εντολών digitalWrite(), οι οποίες θέτουν τους ακροδέκτες (GStepper_Pin1, GStepper_Pin2, GStepper_Pin3 & GStepper_Pin4) σε κατάσταση (LOW) ή αλλιώς σε χαμηλό δυναμικό. Με αυτόν τον τρόπο, σταματά η πλακέτα οδήγησης του βηματικού κινητήρα της χορδής Σολ (G) να παρέχει τροφοδοσία στους μαγνήτες. Έπειτα, το μπλοκ εντολών της συνάρτησης GStep_Motor_Stop (), ολοκληρώνεται με την εκτέλεση της εντολής delay(2), η οποία πραγματοποιεί μια καθυστέρηση των 2msec.

Όσον αφορά την συνάρτηση DStep_Motor_Stop (), απώτερος σκοπός της είναι η απενεργοποίηση του βηματικού κινητήρα που σχετίζεται με τον έλεγχο της περιστροφής του κλειδιού της χορδής Ρε (D). Πιο αναλυτικά, όπως στην προηγούμενη συνάρτηση, έτσι και εδώ, εκτελείτε μία τετράδα εντολών digitalWrite(), οι οποίες θέτουν τους ακροδέκτες (DStepper_Pin1, DStepper_Pin2, DStepper_Pin3 & DStepper_Pin4) σε κατάσταση (LOW) ή αλλιώς σε χαμηλό δυναμικό. Έτσι, παύει η πλακέτα οδήγησης του βηματικού κινητήρα της χορδής Ρε (D) να παρέχει τροφοδοσία στους μαγνήτες του. Επίσης, το μπλοκ εντολών της συνάρτησης DStep_Motor_Stop () ολοκληρώνεται με την εκτέλεση της εντολής καθυστέρησης delay(2), η οποία δημιουργεί μια καθυστέρηση των 2msec.

Εν συνεχεία, η συνάρτηση AStep_Motor_Stop () αναλαμβάνει την απενεργοποίηση του βηματικού κινητήρα που σχετίζεται με τον έλεγχο της περιστροφής του κλειδιού της χορδής Λα (A). Πιο συγκεκριμένα, εκτελείτε μία τετράδα εντολών digitalWrite(), οι

οποίες θέτουν τους ακροδέκτες (AStepper_Pin1, AStepper_Pin2, AStepper_Pin3 & AStepper_Pin4) σε κατάσταση (LOW) ή αλλιώς σε χαμηλό δυναμικό. Με αυτόν τον τρόπο, σταματά η πλακέτα οδήγησης του βηματικού κινητήρα της χορδής Λα (A) να παρέχει τροφοδοσία στους μαγνήτες. Έπειτα, το μπλοκ εντολών της συνάρτησης AStep_Motor_Stop (), ολοκληρώνεται με την εκτέλεση της εντολής delay(2), η οποία πραγματοποιεί μια καθυστέρηση των 2msec.

Τέλος, η συνάρτηση EStep_Motor_Stop () πραγματοποιεί και αυτή κάτι αντίστοιχο με τις προηγούμενες συναρτήσεις. Ουσιαστικά, η συγκεκριμένη συνάρτηση αναλαμβάνει την απενεργοποίηση του βηματικού κινητήρα που σχετίζεται με τον έλεγχο της περιστροφής του κλειδιού της χορδής Μι μπάσο (E). Πιο συγκεκριμένα, εκτελείτε μία τετράδα εντολών digitalWrite(), οι οποίες θέτουν τους ακροδέκτες (EStepper_Pin1, EStepper_Pin2, EStepper_Pin3 & EStepper_Pin4) σε κατάσταση (LOW) ή αλλιώς σε χαμηλό δυναμικό. Με αυτόν τον τρόπο, σταματά η πλακέτα οδήγησης του βηματικού κινητήρα της χορδής Μι (E) να παρέχει τροφοδοσία στους μαγνήτες. Έπειτα, το μπλοκ εντολών της συνάρτησης EStep_Motor_Stop (), ολοκληρώνεται με την εκτέλεση της εντολής delay(2), η οποία πραγματοποιεί μια καθυστέρηση των 2msec.

Πίνακας 3.18: Ο κώδικας προγραμματισμού των συναρτήσεων eStep_Motor_Stop (), BStep_Motor_Stop (), GStep_Motor_Stop (), DStep_Motor_Stop (), AStep_Motor_Stop () & EStep_Motor_Stop () [44].

Κώδικας προγραμματισμού (Μέρος 18ο)

```
void eStep_Motor_Stop () { // Function to deactivate movement of eStep_Motor
  digitalWrite(eStepper_Pin1,LOW); //Set eStepper_Pin1 LOW = 0
  digitalWrite(eStepper_Pin2,LOW); //Set eStepper_Pin2 LOW = 0
  digitalWrite(eStepper_Pin3,LOW); //Set eStepper_Pin3 LOW = 0
  digitalWrite(eStepper_Pin4, LOW); //Set eStepper_Pin4 LOW = 0
  delay(2); //Time delay 2 msec
}

void BStep_Motor_Stop () { // Function to deactivate movement of BStep_Motor
  digitalWrite(BStepper_Pin1,LOW); //Set BStepper_Pin1 LOW = 0
  digitalWrite(BStepper_Pin2,LOW); //Set BStepper_Pin2 LOW = 0
  digitalWrite(BStepper_Pin3,LOW); //Set BStepper_Pin3 LOW = 0
  digitalWrite(BStepper_Pin4, LOW); //Set BStepper_Pin4 LOW = 0
  delay(2); //Time delay 2 msec
}

void GStep_Motor_Stop () { // Function to deactivate movement of GStep_Motor
  digitalWrite(GStepper_Pin1, LOW); //Set GStepper_Pin1 LOW = 0
  digitalWrite(GStepper_Pin2, LOW); //Set GStepper_Pin2 LOW = 0
  digitalWrite(GStepper_Pin3, LOW); //Set GStepper_Pin1 LOW = 0
  digitalWrite(GStepper_Pin4, LOW); //Set GStepper_Pin1 LOW = 0
```

```
  delay(2); //Time delay 2 msec
}

void DStep_Motor_Stop () { // Function to deactivate movement of DStep_Motor
  digitalWrite(DStepper_Pin1, LOW); //Set DStepper_Pin1 LOW = 0
  digitalWrite(DStepper_Pin2, LOW); //Set DStepper_Pin2 LOW = 0
  digitalWrite(DStepper_Pin3, LOW); //Set DStepper_Pin1 LOW = 0
  digitalWrite(DStepper_Pin4, LOW); //Set DStepper_Pin1 LOW = 0
  delay(2); //Time delay 2 msec
}

void AStep_Motor_Stop () { // Function to deactivate movement of AStep_Motor
  digitalWrite(AStepper_Pin1, LOW); //Set AStepper_Pin1 LOW = 0
  digitalWrite(AStepper_Pin2, LOW); //Set AStepper_Pin2 LOW = 0
  digitalWrite(AStepper_Pin3, LOW); //Set AStepper_Pin1 LOW = 0
  digitalWrite(AStepper_Pin4, LOW); //Set AStepper_Pin1 LOW = 0
  delay(2); //Time delay 2 msec
}

void EStep_Motor_Stop () {// Function to deactivate movement of EStep_Motor
  digitalWrite(EStepper_Pin1, LOW); //Set EStepper_Pin1 LOW = 0
  digitalWrite(EStepper_Pin2, LOW); //Set EStepper_Pin2 LOW = 0
  digitalWrite(EStepper_Pin3, LOW); //Set EStepper_Pin1 LOW = 0
  digitalWrite(EStepper_Pin4, LOW); //Set EStepper_Pin1 LOW = 0
  delay(2);
}
```

3.1.15 **Ανάλυση της συνάρτησης Manual_Button_Read() (Κώδικας προγραμματισμού μέρος 19ο)**

Σε αυτήν την ενότητα, πραγματοποιείται ανάλυση της συνάρτησης Manual_Button_Read(), η οποία αποσκοπεί στον εντοπισμών των πλήκτρων (UP, DOWN, LEFT, RIGHT & SELECT) καθώς και στην εύρεση της επιλογής του χρήστη στο χειροκίνητο μενού που του εμφανίζεται στην LCD οθόνη (Πίνακας 3.19). Ουσιαστικά, η συνάρτηση αυτή δίνει την ευχέρεια στον χρήστη να επιλέξει την χορδή που επιθυμεί να ελέγξει χειροκίνητα. Επίσης, αυτή η συνάρτηση αυτή καλείτε του μπλοκ εντολών της συνάρτησης ButtonRead() εφόσον ο χρήστης έχει επιλέξει το χειροκίνητο έλεγχο του συστήματος.

Αρχικά, για να πραγματοποιηθεί αναγνώριση των πλήκτρων (UP, DOWN, LEFT, RIGHT & SELECT), θα πρέπει να ενεργοποιηθεί ο αναλογικό-ψηφιακός μετατροπές και να πραγματοποιηθεί δειγματοληψία από την αναλογική θύρα (A0) του Arduino Mega

2560. Αυτό επιτυγχάνεται με τις πρώτες εντολές που εκτελούνται εντός του μπλοκ εντολών, οι οποίες καθορίζουν τις παραμέτρους του αναλογικό-ψηφιακού μετατροπέα. Πιο αναλυτικά, η εντολές ADCSRA = 0, ADCSRB = 0 και ADMUX = 0 αποσκοπούν στην εκκαθάριση όλων των bit των καταχωρητών (ADCSRA, ADCSRB & ADMUX). Ύστερα, εκτελείται η εντολή ADMUX |= (1<<REFS0)|(0<<REFS1), η οποία εναποθέτει (1) στο bit (REFS0) και (0) στο bit (REFS1) του καταχωρητή (ADMUX). Με αυτόν τον τρόπο, καθορίζεται η τάση αναφοράς του αναλογικό ψηφιακού μετατροπέα. Στην συνέχεια, εκτελείται η εντολή ADCSRB |= (1<<ACME), η οποία εναποθέτει (1) στο bit (ACME) του καταχωρητή (ADCSRB). Η εκτέλεση της εντολής αυτής, οδηγεί στην ενεργοποίηση του πολυπλέκτη σύγκρισης του αναλογικό-ψηφιακού μετατροπέα. Έπειτα, εκτελείται η εντολή ADCSRA |=(1<<ADEN), η οποία εναποθέτει (1) στο bit (ADEN) του καταχωρητή (ADCSRA). Με αυτόν τον τρόπο, πραγματοποιείται ενεργοποίηση του αναλογικό-ψηφιακού μετατροπέα. Αμέσως μετά, εκτελείται η εντολή ADCSRA |= (1<<ADSC), η οποία εναποθέτει (1) στο bit (ADSC) του καταχωρητή (ADCSRA). Σκοπός της συγκεκριμένης εντολής, είναι να ξεκινήσει την δειγματοληψία του σήματος ώστε να επακολουθήσει η αναγνώριση των πλήκτρων (UP, DOWN, LEFT, RIGHT & SELECT). Επίσης, πρέπει να σημειωθεί ότι όσο διαρκεί η δειγματοληψία, το bit (ADSC) παραμένει (1), ενώ όταν ολοκληρωθεί η μετατροπή τότε το bit (ADSC) μεταβαίνει σε κατάσταση (0). Το μόνο που δεν αναφέρθηκε μέχρι στιγμής είναι η αναλογική θύρα από την οποία θα πραγματοποιηθεί η δειγματοληψία του σήματος. Στην περίπτωση αυτή, το σύστημα ορίζει αυτόματα την αναλογική θύρα (A0) για την επίτευξη της δειγματοληψίας. Αφού λοιπόν ολοκληρωθεί η μετατροπή του σήματος από τον αναλογικό ψηφιακό μετατροπέα, τότε το αποτέλεσμα αποθηκεύεται στον καταχωρητή (ADC). Συνεπώς, με την εκτέλεση της εντολής x = ADC, πραγματοποιείται αποθήκευση της τιμής του καταχωρητή (ADC) στην μεταβλητή (x). Ύστερα, εκτελείται η εντολή delay(100), η οποία πραγματοποιεί μια καθυστέρηση των 100msec. Έπειτα, με την εντολή Serial.println(x), εμφανίζεται η τιμή της μεταβλητής (x) στην σειριακή οθόνη.

Εν συνεχεία, εφαρμόζεται η τεχνική ελέγχου (if – else if), ώστε να εντοπιστεί το πλήκτρο που πίεσε ο χρήστης. Πιο συγκεκριμένα, πρώτα εκτελείται η εντολή ελέγχου if (x<60), η οποία ελέγχει αν η μεταβλητή (x) είναι μικρότερη από την τιμή εξήντα. Εφόσον ικανοποιείται η συγκεκριμένη συνθήκη, τότε εκτελείται το μπλοκ εντολών της συνθήκης ελέγχου (if), γεγονός που σημαίνει ότι ο χρήστης πίεσε το κουμπί (RIGHT). Πιο αναλυτικά, εντός του μπλοκ εντολών της συνθήκης ελέγχου (if) εκτελείται η εντολή manual_menu+=2, η οποία αποσκοπεί στην αύξηση της τιμής της μεταβλητής (manual_menu) κατά δύο. Ύστερα, εκτελείται η φωλιασμένη εντολή ελέγχου if (manual_menu > 7), η οποία ελέγχει αν η τιμή της μεταβλητής (manual_menu) είναι μεγαλύτερη από την τιμή επτά. Στην περίπτωση που ικανοποιείται η συγκεκριμένη συνθήκη ελέγχου (if), τότε εκτελείται η μοναδική εντολή εντός του μπλοκ εντολών manual_menu = 7, η οποία αποσκοπεί στην αποθήκευση της τιμής επτά στην μεταβλητή (manual_menu). Βγαίνοντας από το μπλοκ εντολών της συνθήκης ελέγχου if (manual_menu > 7), προκαλείτε μια καθυστέρηση των 50msec μέσω της εκτέλεσης της εντολής delay(50), με την οποία και ολοκληρώνεται το μπλοκ εντολών της εντολής ελέγχου if (x<60).

Από την άλλη πλευρά, αν δεν ικανοποιείται η προηγούμενη εντολή ελέγχου (if), τότε εκτελείται η εντολή ελέγχου else if (x<200), η οποία ελέγχει αν η μεταβλητή (x) είναι μικρότερη από την τιμή διακόσια. Στην περίπτωση που ικανοποιείται η συγκεκριμένη

συνθήκη, αυτό σημαίνει ότι ο χρήστης έχει πιέσει το κουμπί (UP) και τότε εκτελείται το μπλοκ εντολών της συνθήκης ελέγχου (else if). Πιο συγκεκριμένα, εντός του μπλοκ εντολών, εκτελείται η εντολή manual_menu--, η οποία μειώνει την τιμή της μεταβλητής (manual_menu) κατά μία μονάδα. Έπειτα, εκτελείται η φωλιασμένη εντολή ελέγχου if (manual_menu == 0), η οποία ελέγχει αν η τιμή της μεταβλητής (manual_menu) είναι ίση με μηδέν. Αν ικανοποιείται η συγκεκριμένη συνθήκη ελέγχου, τότε εκτελείται το μπλοκ εντολών της φωλιασμένης συνθήκης ελέγχου (if). Το συγκεκριμένο μπλοκ εντολών εκτελεί την εντολή manual_menu = 1, η οποία αποθηκεύει στην μεταβλητή (manual_menu) την τιμή ένα. Βγαίνοντας από το μπλοκ εντολών της συνθήκης ελέγχου if (manual_menu == 0), προκαλείτε μια καθυστέρηση των 50msec μέσω της εκτέλεσης της εντολής delay(50), με την οποία και ολοκληρώνεται το μπλοκ εντολών της εντολής ελέγχου else if (x<200).

Στην περίπτωση όμως που δεν ικανοποιείται η προηγούμενη συνθήκη (else if), τότε εκτελείται η επόμενη εντολή ελέγχου else if(x<400), η οποία ελέγχει αν η μεταβλητή (x) είναι μικρότερη από την τιμή τετρακόσια. Για να ικανοποιηθεί η συγκεκριμένη συνθήκη, θα πρέπει ο χρήστης να πιέσει το πλήκτρο (DOWN) και τότε θα ξεκινήσει η εκτέλεση του μπλοκ εντολών της συνθήκης ελέγχου (else if). Πιο αναλυτικά, εντός του συγκεκριμένου μπλοκ εντολών, εκτελείται η εντολή manual_menu++, η οποία αυξάνει την τιμή της μεταβλητής (manual_menu) κατά μία μονάδα. Ύστερα, εκτελείται η φωλιασμένη εντολή ελέγχου if (manual_menu > 7), η οποία ελέγχει αν η τιμή της μεταβλητής (manual_menu) είναι μεγαλύτερη από την τιμή επτά. Αν ικανοποιείται η συγκεκριμένη συνθήκη, τότε εκτελείται η μοναδική εντολή που βρίσκεται εντός του μπλοκ εντολών manual_menu = 7, η οποία αποθηκεύει την τιμή επτά στην μεταβλητή (manual_menu). Βγαίνοντας από το μπλοκ εντολών της συνθήκης ελέγχου if (manual_menu > 7), προκαλείτε μια καθυστέρηση των 50msec μέσω της εκτέλεσης της εντολής delay(50), με την οποία και ολοκληρώνεται το μπλοκ εντολών της εντολής ελέγχου else if (x<400).

Από την άλλη, αν δεν ικανοποιείται ούτε η προηγούμενη συνθήκη ελέγχου (else if), τότε πραγματοποιείται εκτέλεση της επόμενης εντολής ελέγχου else if(x<600), η οποία ελέγχει αν η τιμή της μεταβλητής (x) είναι μικρότερη από εξακόσια. Ο μόνος τρόπος για να ικανοποιηθεί η συγκεκριμένη συνθήκη ελέγχου (else if) είναι να πιέσει ο χρήστης το κουμπί (LEFT) και έτσι θα ξεκινήσει η εκτέλεση του μπλοκ εντολών της συνάρτησης. Πιο συγκεκριμένα, εντός του μπλοκ εντολών της συνθήκης (else if) εκτελείται η εντολή manual_menu-=2, η οποία μειώνει την τιμή της μεταβλητής (manual_menu) κατά δύο μονάδες. Στην συνέχεια, εκτελείται η φωλιασμένη εντολή ελέγχου if (manual_menu < 1), η οποία ελέγχει αν η τιμή της μεταβλητής (manual_menu) είναι μικρότερη από το ένα. Στην περίπτωση που ικανοποιείται η συγκεκριμένη συνθήκη ελέγχου, τότε εκτελείται το μπλοκ εντολών της φωλιασμένης συνθήκης ελέγχου (if). Ουσιαστικά, το συγκεκριμένο μπλοκ εντολών εκτελεί την εντολή manual_menu = 1, η οποία αποθηκεύει στην μεταβλητή (manual_menu) την τιμή ένα. Βγαίνοντας από το μπλοκ εντολών της συνθήκης ελέγχου if (manual_menu < 1), προκαλείτε μια καθυστέρηση των 50msec μέσω της εκτέλεσης της εντολής delay(50), με την οποία και ολοκληρώνεται το μπλοκ εντολών της εντολής ελέγχου else if (x<600).

Τέλος, στην περίπτωση που δεν ικανοποιείτε καμία από τις προηγούμενες εντολές ελέγχου (else if), τότε εκτελείται η τελευταία εντολή ελέγχου else if(x<800), η οποία ελέγχει αν η τιμή της μεταβλητής (x) είναι μικρότερη από την τιμή οκτακόσια. Για να ικανοποιηθεί η συγκεκριμένη συνθήκη ελέγχου, θα πρέπει ο χρήστης να πιέσει το πλήκτρο

(SELECT) και έτσι θα ξεκινήσει η εκτέλεση του μπλοκ εντολών της συνθήκης ελέγχου (else if). Πιο αναλυτικά, εντός του μπλοκ εντολών της συνθήκης ελέγχου (else if) εκτελείται η εντολή Manual_Select = 1, η οποία θέτει την τιμή ένα στην σημαία (flag) με όνομα (Manual_Select). Με αυτόν τον τρόπο το σύστημα αντιλαμβάνεται ότι ο χρήστης έχει οριστικοποιήσει την επιλογή του στο χειροκίνητο μενού. Έτσι λοιπόν ολοκληρώνεται το μπλοκ εντολών της συνάρτησης Manual_Button_Read(), η οποία αποσκοπεί στην αναγνώριση των πλήκτρων (UP, DOWN, LEFT, RIGHT & SELECT).

Πίνακας 3.19: Ο κώδικας προγραμματισμού της συνάρτησης Manual_Button_Read() [45].

Κώδικας προγραμματισμού (Μέρος 19ο)

```
void Manual_Button_Read() //Function for button read at manual control
{
  ADCSRA = 0; //Clear ADCSRA register
  ADCSRB = 0; //Clear ADCSRB register
  ADMUX = 0; //Clear ADMUX register
  ADMUX |= (1<<REFS0)|(0<<REFS1); //Set bit REFS0 and clear bit REFS1 at ADMUX register
  ADCSRB |= (1<<ACME); //Set bit ACME at ADCSRB register
  ADCSRA |=(1<<ADEN); //Set bit ADEN at ADCSRA register
  ADCSRA |= (1<<ADSC); //Set bit ADSC at ADCSRA register
  x = ADC; //Store ADC value to x variable
  delay(100); // Delay 100 msec
  Serial.println(x); //Print x value at Serial port

  if (x<60) //If right button has been pressed
  {
    manual_menu+=2; //Add 2 at current variable manual_menu and store the result at the same variable
    if (manual_menu > 7) // If the value of the manual_menu variable is bigger than 7
    {
      manual_menu = 7; //Store number 7 at manual_menu variable
    }
    delay(50); //Delay 50 msec
  }
  else if(x<200) //If up button has been pressed
  {
    manual_menu--; //Reduce manual_menu variable by 1
    if (manual_menu == 0) //If manual_menu variable is equal to 0
    {
      manual_menu = 1; //Store number 1 at manual_menu variable
```

```
   }
   delay(50); //Delay 50 msec
  }
  else if(x<400) //If down button has been pressed
  {
   manual_menu++; //Increase manual_menu variable by 1
   if (manual_menu > 7) //If manual_menu variable is bigger than 7
   {
    manual_menu = 7; //Store number 7 at manual_menu variable
   }
   delay(50); //Delay 50 msec
  }
  else if(x<600) //If left button has been pressed
  {
   manual_menu-=2; //Reduce manual_menu variable by 2
   if (manual_menu < 1) //If manual_menu variable is less than 1
   {
    manual_menu = 1; //Store number 1 at manual_menu variable
   }
   delay(50); //Delay 50 msec
  }
  else if(x<800) //If select button has been pressed
  {
   Manual_Select = 1; //Set Manual_Select
  }
} //Manual_Button_Read()
```

3.1.16 **Ανάλυση της συνάρτησης Manual_Step_Motor_Control() (Κώδικας προγραμματισμού μέρος 20ο)**

Σε αυτή την ενότητα, πραγματοποιείται ανάλυση του κώδικα προγραμματισμού της συνάρτησης Manual_Step_Motor_Control(), η οποία είναι υπεύθυνη για τον χειροκίνητο έλεγχο του βηματικού κινητήρα του κλειδιού της κάθε χορδής (Πίνακας 3.20). Επιπλέον, η συνάρτηση αυτή καλείτε του μπλοκ εντολών της συνάρτησης ButtonRead() μόνο όταν ο χρήστης οριστικοποιήσει την επιλογή του στο μενού χειροκίνητου ελέγχου που εμφανίζεται στην LCD οθόνη. Ουσιαστικά, η συνάρτηση αυτή δίνει την δυνατότητα στον χρήστη να επιλέξει την φορά περιστροφής του άξονα του κάθε βηματικού κινητήρα. Επίσης, προβάλει μηνύματα στην LCD οθόνη, ώστε να προτρέψει τον χρήστη να πιέσει τα κατάλληλα πλήκτρα.

Αφού λοιπόν απαιτείται η χρήση των πλήκτρων που βρίσκονται ενσωματωμένα στην πλακέτα της LCD οθόνης για τον έλεγχο της φοράς περιστροφής του άξονα του

βηματικού κινητήρα, θα πρέπει εντός του μπλοκ εντολών να εκτελούνται εντολές για την αναγνώριση των πλήκτρων. Αρχικά, εντός του μπλοκ εντολών, εκτελούνται οι εντολές ADCSRA = 0, ADCSRB = 0 & ADMUX = 0 οι οποίες αποσκοπούν στην εκκαθάριση όλων των bit των καταχωρητών (ADCSRA, ADCSRB & ADMUX). Στην συνέχεια, εκτελείται η εντολή ADMUX |= (1<<REFS0)|(0<<REFS1), η οποία εναποθέτει (1) στο bit (REFS0) και (0) στο bit (REFS1) του καταχωρητή (ADMUX). Έτσι επιτυγχάνεται, ο καθορισμός τάση αναφοράς του αναλογικό ψηφιακού μετατροπέα. Ύστερα, πραγματοποιείτε εκτέλεση της εντολής ADCSRB |= (1<<ACME), η οποία εναποθέτει (1) στο bit (ACME) του καταχωρητή (ADCSRB). Ουσιαστικά, η εκτέλεση της εντολής αυτής οδηγεί στην ενεργοποίηση του πολυπλέκτη σύγκρισης του αναλογικό-ψηφιακού μετατροπέα. Αμέσως μετά, εκτελείται η εντολή ADCSRA |=(1<<ADEN), η οποία εναποθέτει (1) στο bit (ADEN) του καταχωρητή (ADCSRA). Με αυτόν τον τρόπο, πραγματοποιείται η ενεργοποίηση του αναλογικό-ψηφιακού μετατροπέα. Έπειτα, εκτελείται η εντολή ADCSRA |= (1<<ADSC), η οποία εναποθέτει (1) στο bit (ADSC) του καταχωρητή (ADCSRA). Σκοπός της συγκεκριμένης εντολής, είναι να ξεκινήσει την δειγματοληψία του σήματος ώστε να επακολουθήσει η αναγνώριση των πλήκτρων (UP, DOWN, LEFT, RIGHT & SELECT). Επιπροσθέτως, είναι αξιοσημείωτο ότι όσο διαρκεί η δειγματοληψία, το bit (ADSC) παραμένει (1), ενώ όταν ολοκληρωθεί η μετατροπή τότε το bit (ADSC) μεταβαίνει σε κατάσταση (0). Ωστόσο, αυτό που δεν έχει αναφερθεί μέχρι στιγμής είναι η αναλογική θύρα από την οποία θα πραγματοποιηθεί η δειγματοληψία του σήματος. Στην περίπτωση αυτή, το σύστημα ορίζει αυτόματα την αναλογική θύρα (A0) για την επίτευξη της δειγματοληψίας. Όταν ολοκληρωθεί η μετατροπή του σήματος από τον αναλογικό-ψηφιακό μετατροπέα, τότε προκύπτει ένα αποτέλεσμα το οποίο αποθηκεύεται στον καταχωρητή (ADC). Εν συνεχεία, εκτελείται η εντολή x = ADC, η οποία αποθηκεύει την τιμή που έχει ο καταχωρητής (ADC) στην μεταβλητή (x). Έπειτα, πραγματοποιείται μια καθυστέρηση των 100msec εξαιτίας της εντολής delay(100). Ύστερα, πραγματοποιείται εκτύπωση της τιμής της μεταβλητής (x) στην σειριακή οθόνη μέσω της εντολής Serial.println(x).

Όσον αφορά την ανίχνευση των πλήκτρων (UP, DOWN, LEFT, RIGHT & SELECT), θα πρέπει να γίνει χρήση της τεχνικής (if – else if) ώστε το σύστημα να εντοπίσει πιο πλήκτρο έχει πιέσει ο χρήστης. Πιο αναλυτικά, εκτελείται η εντολή ελέγχου if (x<60), η οποία ελέγχει αν η τιμή της μεταβλητής (x) είναι μικρότερη από εξήντα. Για να ικανοποιηθεί η συγκεκριμένη συνθήκη, θα πρέπει ο χρήστης να πιέσει το κουμπί (RIGHT) και τότε θα εκτελεστεί το μπλοκ εντολών της συνθήκης ελέγχου (if). Εντός του συγκεκριμένου μπλοκ εντολών εκτελείται η εντολή Turn_Right = 1, η οποία θέτει την τιμή ένα στην σημαία (flag) με όνομα (Turn_Right). Έτσι δηλώνεται στο σύστημα ότι ο χρήστης έχει πιέσει το κουμπί (RIGHT) και ο κινητήρας περιστρέφεται κατάλληλα.

Στην περίπτωση όμως που δεν ικανοποιείται η συνθήκη ελέγχου (if), τότε εκτελείται η εντολή ελέγχου else if(x<200), η οποία ελέγχει αν η τιμή της μεταβλητής (x) είναι μικρότερη από διακόσια. Η συγκεκριμένη συνθήκη, ικανοποιείται όταν ο χρήστης έχει πιέσει το κουμπί (UP) και τότε εκτελείται το μπλοκ εντολών της συγκεκριμένης συνθήκης ελέγχου (else if). Πιο συγκεκριμένα, εντός του μπλοκ εντολών της συνθήκης ελέγχου (else if), εκτελούνται με κατάλληλη σειρά οι εντολές lcd.clear(), lcd.setCursor() & lcd.print(), ώστε να εμφανιστεί το μήνυμα (Press Right or Left Buttons) στην LCD οθόνη. Το μήνυμα αυτό παραμένει στην LCD οθόνη για 3sec εξαιτίας της εντολής delay(3000) που εκτελείται αμέσως μετά. Με αυτό τον τρόπο, ο χρήστης ενημερώνεται από το σύστημα ότι

έχει πιέσει λάθος κουμπί και καθοδηγείτε για το ποια είναι τα σωστά κουμπιά που πρέπει να πιέσει. Ύστερα, εκτελούνται ξανά οι εντολές lcd.clear(), lcd.setCursor() & lcd.print() με την κατάλληλη σειρά, ώστε να εμφανιστεί το μήνυμα (Press SELECT to go back) στην LCD οθόνη. Το μήνυμα αυτό αποσκοπεί στο να ενημερώσει τον χρήστη ότι πατώντας το κουμπί (SELECT), παύει ο έλεγχος του βηματικού κινητήρα που είχε επιλέξει και επιστρέφει στο μενού του χειροκίνητου ελέγχου το οποίο εμφανίζεται και στην LCD οθόνη.

Από την άλλη, αν δεν ικανοποιείται η προηγούμενη συνθήκη ελέγχου (else if), τότε εκτελείται η επόμενη εντολή ελέγχου else if(x<400), η οποία ελέγχει αν η τιμή της μεταβλητής (x) είναι μικρότερη από τετρακόσια. Για να ικανοποιηθεί η συγκεκριμένη συνθήκη, θα πρέπει ο χρήστης να πιέσει το κουμπί (DOWN) και τότε θα εκτελεστεί το μπλοκ εντολών της συγκεκριμένης συνθήκης ελέγχου (else if). Πιο αναλυτικά, το μπλοκ εντολών της συνθήκης ελέγχου (else if), εκτελεί με κατάλληλη σειρά τις εντολές lcd.clear(), lcd.setCursor() & lcd.print(), ώστε να εμφανιστεί το μήνυμα (Press Right or Left Buttons) στην LCD οθόνη για 3sec εξαιτίας της εντολής delay(3000) που εκτελείται αμέσως μετά. Με αυτό τον τρόπο, ο χρήστης ενημερώνεται από το σύστημα ότι έχει πιέσει λάθος κουμπί και καθοδηγείτε για το ποια είναι τα σωστά κουμπιά που πρέπει να πιέσει. Έπειτα, εκτελούνται ξανά οι εντολές lcd.clear(), lcd.setCursor() & lcd.print() με την κατάλληλη σειρά, ώστε να εμφανιστεί το μήνυμα (Press SELECT to go back) στην LCD οθόνη. Το μήνυμα αυτό αποσκοπεί στην ενημέρωση του χρήστη ότι πατώντας το κουμπί (SELECT), παύει ο έλεγχος του βηματικού κινητήρα που είχε επιλέξει και επιστρέφει στο μενού του χειροκίνητου ελέγχου το οποίο εμφανίζεται και στην LCD οθόνη.

Αν όμως, δεν έχει ικανοποιηθεί ούτε η προηγούμενη συνθήκη ελέγχου (else if), τότε εκτελείται η επόμενη εντολή ελέγχου else if(x<600), η οποία ελέγχει αν η τιμή της μεταβλητής (x) είναι μικρότερη από εξακόσια. Η συγκεκριμένη συνθήκη, ικανοποιείται όταν ο χρήστης έχει πιέσει το κουμπί (LEFT) και τότε εκτελείται το μπλοκ εντολών της συγκεκριμένης συνθήκης ελέγχου (else if). Πιο συγκεκριμένα, εντός του συγκεκριμένου μπλοκ εντολών, εκτελείται η εντολή Turn_Left = 1, η οποία θέτει την τιμή ένα στην σημαία (flag) με όνομα (Turn_Left). Έτσι δηλώνεται στο σύστημα ότι ο χρήστης έχει πιέσει το κουμπί (RIGHT) και ο κινητήρας περιστρέφεται κατάλληλα.

Τέλος, αν δεν ικανοποιείται καμία από τις προηγούμενες συνθήκες ελέγχου, τότε εκτελείται η εντολή ελέγχου else if(x<800), η οποία ελέγχει αν η τιμή της μεταβλητής (x) είναι μικρότερη από οκτακόσια. Για να ικανοποιηθεί η συγκεκριμένη συνθήκη, θα πρέπει ο χρήστης να πιέσει το κουμπί (SELECT) και τότε θα εκτελεστεί το μπλοκ εντολών της συγκεκριμένης συνθήκης ελέγχου (else if). Πιο αναλυτικά, εντός του συγκεκριμένου μπλοκ εντολών, εκτελείται η εντολή Finish_Control = 1, η οποία θέτει την τιμή ένα στην σημαία (flag) με όνομα (Finish_Control). Με αυτόν τον τρόπο το σύστημα αναγνωρίζει ότι ο χρήστης επιθυμεί την λήξη του ελέγχου του συγκεκριμένου βηματικού κινητήρα και την επιστροφή στο μενού επιλογής του χειροκίνητου ελέγχου. Έπειτα, με την ακόλουθη σειρά γίνεται κλήση των συναρτήσεων eStep_Motor_Stop (), BStep_Motor_Stop (), GStep_Motor_Stop (), DStep_Motor_Stop (), AStep_Motor_Stop () & EStep_Motor_Stop (). Με αυτόν τον τρόπο, απενεργοποιούνται όλες οι ψηφιακές θύρες που ελέγχουν τους βηματικούς κινητήρες, εξοικονομώντας έτσι την ενέργεια των μπαταριών. Το μπλοκ εντολών ολοκληρώνεται με την εκτέλεση της εντολής lcd.clear(), η οποία σβήνει όλους του χαρακτήρες από την LCD οθόνη ώστε να εμφανιστεί ξανά το μενού επιλογής του

χειροκίνητου ελέγχου του συστήματος. Έτσι λοιπόν ολοκληρώνεται και το μπλοκ εντολών της συνάρτησης Manual_Step_Motor_Control().

Πίνακας 3.20: Ο κώδικας προγραμματισμού της συνάρτησης Manual_Step_Motor_Control() [46].

Κώδικας Προγραμματισμού (Μέρος 16ο)

```
void Manual_Step_Motor_Control() //Function for Manual keys control
{
  ADCSRA = 0; //Clear ADCSRA register
  ADCSRB = 0; //Clear ADCSRB register
  ADMUX = 0; //Clear ADMUX register
  ADMUX |= (1<<REFS0)|(0<<REFS1); //Set bit REFS0 and clear bit REFS1 at ADMUX
register
  ADCSRB |= (1<<ACME); //Set bit ACME at ADCSRB register
  ADCSRA |=(1<<ADEN); //Set bit ADEN at ADCSRA register
  ADCSRA |= (1<<ADSC); //Set bit ADSC at ADCSRA register
  x = ADC; //Store ADC value at x variable
  delay(100); //Delay 100 msec
  Serial.println(x); //Print x value at Serial port
  if (x<60) //If right button has been pressed
  {
    Turn_Right = 1; //Set Turn_Right Variable
  }
  else if(x<200) //If up button has been pressed
  {
    lcd.clear(); //Clear LCD screen
    lcd.setCursor(0,0); //Set cursor at 1st column and 1st line
    lcd.print(" Press Right or "); //Print message at LCD screen
    lcd.setCursor(0,1); //Set cursor at 1st column and 2nd line
    lcd.print("  Left Buttons  "); //Print message at LCD screen
    delay(3000); //Delay 3 sec
    lcd.clear(); //Clear LCD screen
    lcd.setCursor(0,0); //Set cursor at 1st column and 1st line
    lcd.print("  Press SELECT  "); //Print message at LCD screen
    lcd.setCursor(0,1); //Set cursor at 1st column and 2nd line
    lcd.print("   to go back   "); //Print message at LCD screen
  }
  else if(x<400) //If down button has been pressed
  {
    lcd.clear(); //Clear LCD  screen
```

```
   lcd.setCursor(0,0); //Set cursor at 1st column and 1st line
   lcd.print(" Press Right or "); //Print message at LCD screen
   lcd.setCursor(0,1); //Set cursor at 1st column and 1st line
   lcd.print("  Left Buttons  "); //Print message at LCD screen
   delay(3000); //Delay 3 sec
   lcd.clear(); //Clear LCD screen
   lcd.setCursor(0,0); //Set cursor at 1st column and 1st line
   lcd.print("  Press SELECT  "); //Print message at LCD screen
   lcd.setCursor(0,1); //Set cursor at 1st column and 1st line
   lcd.print("   to go back   "); //Print message at LCD screen
  }
  else if(x<600) //If left button has been pressed
  {
   Turn_Left = 1; //Set Turn_Left Variable
  }
  else if(x<800) //If select button has been pressed
  {
   Finish_Control = 1; //Set Finish_Control Variable to return at previous Menu
   eStep_Motor_Stop (); //Function to Deactivate e string Step Motor
   BStep_Motor_Stop (); //Function to Deactivate B string Step Motor
   GStep_Motor_Stop (); //Function to Deactivate G string Step Motor
   DStep_Motor_Stop (); //Function to Deactivate D string Step Motor
   AStep_Motor_Stop (); //Function to Deactivate A string Step Motor
   EStep_Motor_Stop (); //Function to Deactivate E string Step Motor
   lcd.clear(); //Clear LCD screen
  }
} //void Manual_Step_Motor_Control()
```

3.2 Οδηγίες χρήσης του συστήματος αυτομάτου ελέγχου για τον συντονισμό των χορδών της κιθάρας

Στην προηγούμενη ενότητα αναλύθηκε ο κώδικας προγραμματισμού που χρησιμοποιεί ο μικροελεγκτής για την επίτευξη του κουρδίσματος των χορδών της κιθάρας. Προφανώς όμως, ο χρήστης δεν μπορεί να αντιληφθεί την λειτουργία του συστήματος μέσα από τον κώδικα προγραμματισμού και ειδικά αν δεν έχει τις απαραίτητες γνώσεις. Γι αυτό λοιπόν, σε αυτή την ενότητα πραγματοποιείται μια απλή ανάλυση του τρόπου λειτουργίας του συστήματος μέσα από απλά βήματα. Σκοπός της συγκεκριμένης ενότητας, είναι να γίνει όσο τον δυνατόν πιο απλός και κατανοητός ο τρόπος λειτουργία του συστήματος στον χρήστη.

3.2.1 Οδηγίες για την επίτευξη του κλασσικού κουρδίσματος της κιθάρας (Standard Tune)

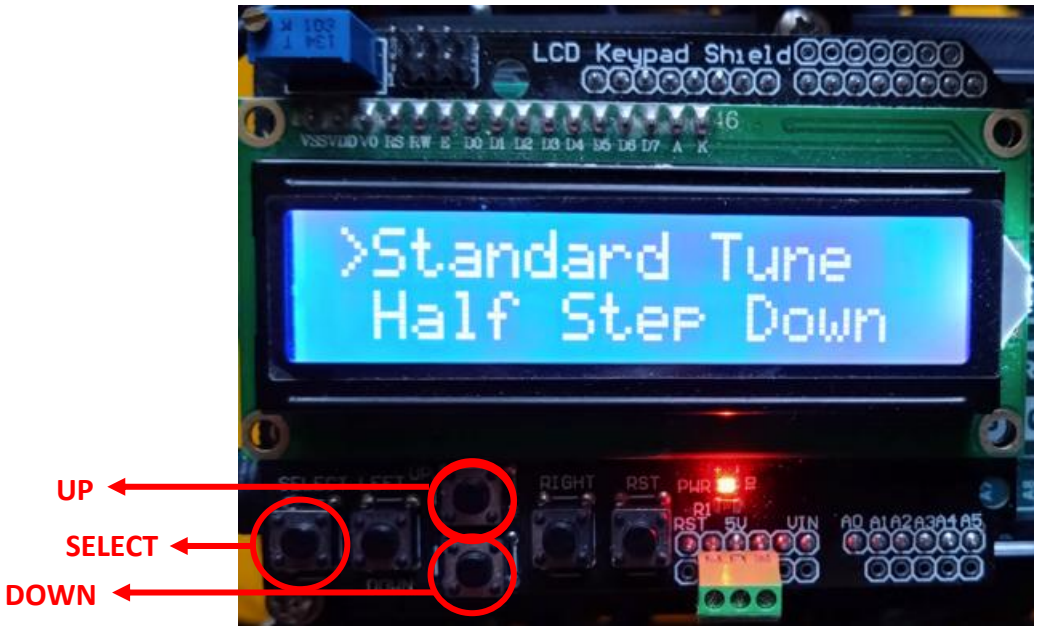
Βήμα 1ο: Ενεργοποίηση του συστήματος με την χρήση του τριπολικού διακόπτη. Αμέσως, ενεργοποιείται το Arduino και κατά συνέπεια η LCD οθόνη. Ο χρήστης, πρώτα θα δει στην LCD το όνομα του συστήματος (Εικόνα 3.1), ύστερα το μήνυμα που τον προτρέπει να επιλέξει κούρδισμα (Εικόνα 3.2) και έπειτα το μενού επιλογή (Εικόνα 3.3).

Εικόνα 3.1: Το όνομα του συστήματος στην LCD οθόνη [242].

Εικόνα 3.2: Το μήνυμα που προτρέπει τον χρήστη να επιλέξει κούρδισμα [243].

Εικόνα 3.3: Το μενού για την επιλογή κουρδίσματος ή του χειροκίνητου ελέγχου του συστήματος [244].

Βήμα 2ο: Για την εκκίνηση του κλασσικού κουρδίσματος της κιθάρας (Standard Tune), θα πρέπει το βέλος (>) να δείχνει την επιλογή (Standard tune) με την χρήση των πλήκτρων (UP & DOWN). Έπειτα, ο χρήστης θα πρέπει να οριστικοποιήσει την επιλογή του πιέζοντας το πλήκτρο (SELECT) (Εικόνα 3.4).



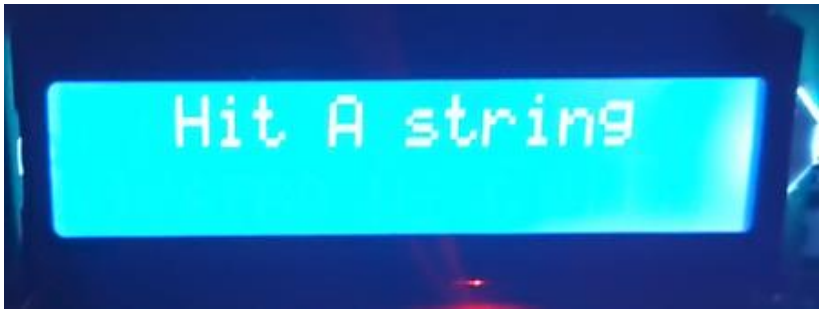
Εικόνα 3.4: Επιλογή του κλασσικού κουρδίσματος (Standard Tune) και οριστικοποίηση της επιλογής [245].

Βήμα 3ο: Στην συνέχεια, εμφανίζεται ένα μήνυμα που προτρέπει τον χρήστη να διεγείρει την χορδή Μι μπάσο (E) (Εικόνα 3.5) και αμέσως μετά εμφανίζεται η επιθυμητή τιμή της συχνότητας ταλάντωσης που πρέπει να έχει η συγκεκριμένη χορδή (Εικόνα 3.6).

Εικόνα 3.5: Το μήνυμα για την διέγερση της χορδής Μι μπάσο (E) [246].

Εικόνα 3.6: Το μήνυμα που εμφανίζει την επιθυμητή συχνότητα ταλάντωσης της χορδής Μι μπάσο (E) [247].

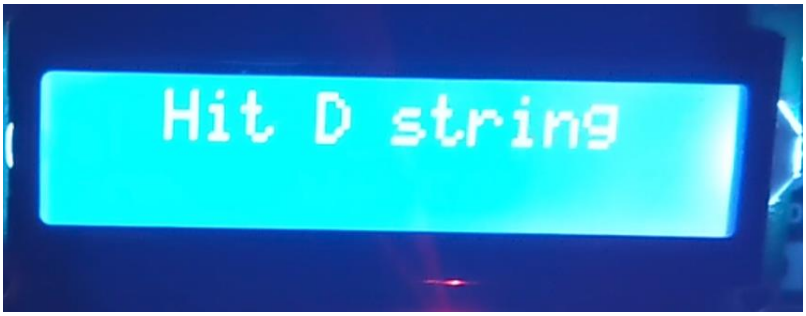
Βήμα 4ο: Αφού ολοκληρωθεί ο συντονισμός της χορδής Μι μπάσο (E), στην οθόνη εμφανίζεται ένα μήνυμα που προτρέπει τον χρήστη να διεγείρει την χορδή Λα (A) (Εικόνα 3.7) και έπειτα, εμφανίζεται η επιθυμητή τιμή της συχνότητας ταλάντωσης που πρέπει να έχει η συγκεκριμένη χορδή (Εικόνα 3.8).



Εικόνα 3.7: Το μήνυμα για την διέγερση της χορδής Λα (A) [248].

Εικόνα 3.8: Το μήνυμα που εμφανίζει την επιθυμητή συχνότητα ταλάντωσης της χορδής Λα (A) [249].

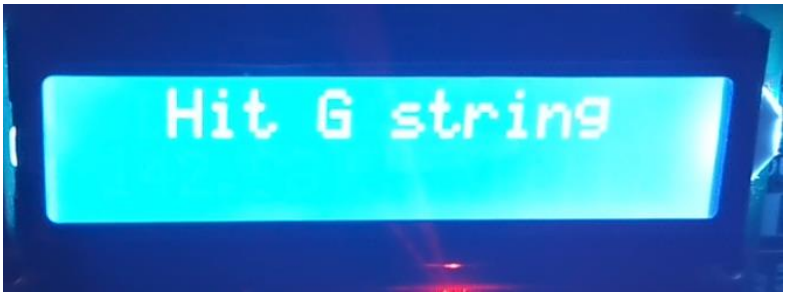
Βήμα 5ο: Μετά την ολοκλήρωση του συντονισμού της χορδής Λα (A), στην οθόνη εμφανίζεται ένα μήνυμα το οποίο προτρέπει τον χρήστη να διεγείρει την χορδή Ρε (D) (Εικόνα 3.9) και έπειτα, εμφανίζεται η επιθυμητή τιμή της συχνότητας ταλάντωσης που πρέπει να έχει η συγκεκριμένη χορδή (Εικόνα 3.10).



Εικόνα 3.9: Το μήνυμα για την διέγερση της χορδής Ρε (D) [250].

Εικόνα 3.10: Το μήνυμα που εμφανίζει την επιθυμητή συχνότητα ταλάντωσης της χορδής Ρε (D) [251].

Βήμα 6ο: Αφού ολοκληρωθεί ο συντονισμός της χορδής Ρε (D), στην οθόνη εμφανίζεται ένα μήνυμα που προτρέπει τον χρήστη να διεγείρει την χορδή Σολ (G) (Εικόνα 3.11) και έπειτα, εμφανίζεται η επιθυμητή τιμή της συχνότητας ταλάντωσης που πρέπει να έχει η συγκεκριμένη χορδή (Εικόνα 3.12).



Εικόνα 3.11: Το μήνυμα για την διέγερση της χορδής Σολ (G) [252].

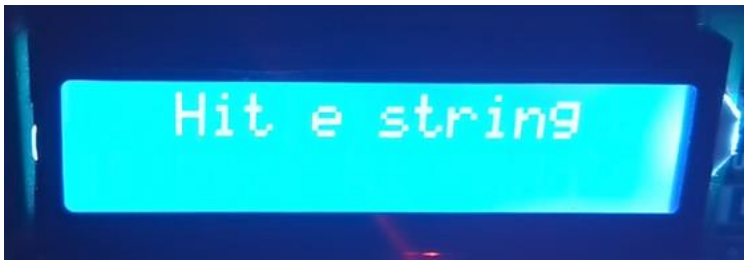
Εικόνα 3.12: Το μήνυμα που εμφανίζει την επιθυμητή συχνότητα ταλάντωσης της χορδής Σολ (G) [253].

Βήμα 7ο: Μετά την ολοκλήρωση του συντονισμού της χορδής Σολ (G), στην LCD οθόνη εμφανίζεται ένα μήνυμα το οποίο προτρέπει τον χρήστη να διεγείρει την χορδή Σι (B) (Εικόνα 3.13) και έπειτα, εμφανίζεται η επιθυμητή τιμή της συχνότητας ταλάντωσης που πρέπει να έχει η συγκεκριμένη χορδή (Εικόνα 3.14).

Εικόνα 3.13: Το μήνυμα για την διέγερση της χορδής Σι (B) [254].

Εικόνα 3.14: Το μήνυμα που εμφανίζει την επιθυμητή συχνότητα ταλάντωσης της χορδής Σι (B) [255].

Βήμα 8ο: Αφού ολοκληρωθεί ο συντονισμός της χορδής Σι (B), τότε εμφανίζεται στην LCD οθόνη ένα μήνυμα το οποίο προτρέπει τον χρήστη να διεγείρει την χορδή Μι καντίνι (e) (Εικόνα 3.15) και στην συνέχεια, εμφανίζεται η επιθυμητή τιμή της συχνότητας ταλάντωσης που πρέπει να έχει η συγκεκριμένη χορδή (Εικόνα 3.16).



Εικόνα 3.15: Το μήνυμα για την διέγερση της χορδής Μι (e) [256].

Εικόνα 3.16: Το μήνυμα που εμφανίζει την επιθυμητή συχνότητα ταλάντωσης της χορδής Μι (e) [257].

Βήμα 9ο: Τέλος, με την ολοκλήρωση του συντονισμού της τελευταίας χορδής Μι καντίνι (e), ολοκληρώνεται το κλασσικό κούρδισμα (Standard Tune), εμφανίζοντας το αντίστοιχο

μήνυμα ολοκλήρωσης (Εικόνα 3.17). Ύστερα, στην LCD οθόνη εμφανίζεται το μενού επιλογών (Εικόνα 3.18) και ο χρήστης μπορεί να επιλέξει είτε κάποια νέα ενέργεια, είτε να απενεργοποιήσει το σύστημα μέσω του τριπολικού διακόπτη.

Εικόνα 3.17: Το μήνυμα ολοκλήρωσης του κλασσικού κουρδίσματος (Standard Tune) [258].

Εικόνα 3.18: Επανεμφάνιση του μενού επιλογών μετά την ολοκλήρωση του κουρδίσματος [259].

3.2.2 Οδηγίες για την επίτευξη του κουρδίσματος των χορδών ένα ημίτονο κάτω (Half-Step Down Tune)

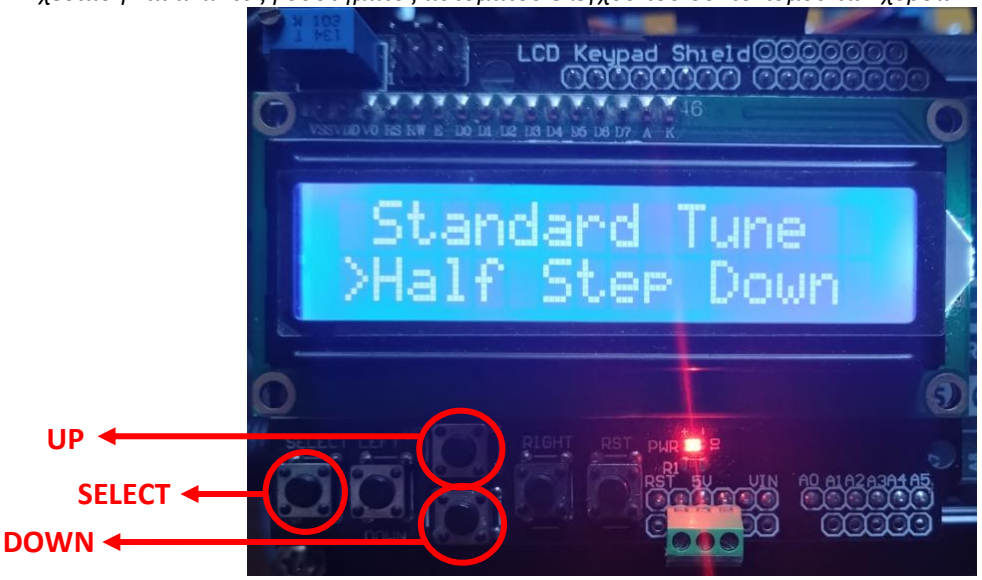
Βήμα 1ο: Ενεργοποίηση του συστήματος με την χρήση του τριπολικού διακόπτη. Αμέσως, ενεργοποιείται το Arduino και κατά συνέπεια η LCD οθόνη. Ο χρήστης, πρώτα θα δει στην LCD το όνομα του συστήματος (Εικόνα 3.19), ύστερα το μήνυμα που τον προτρέπει να επιλέξει κούρδισμα (Εικόνα 3.20) και έπειτα το μενού επιλογή (Εικόνα 3.21).

Εικόνα 3.19: Το όνομα του συστήματος στην LCD οθόνη [260].

Εικόνα 3.20: Το μήνυμα που προτρέπει τον χρήστη να επιλέξει κούρδισμα [261].

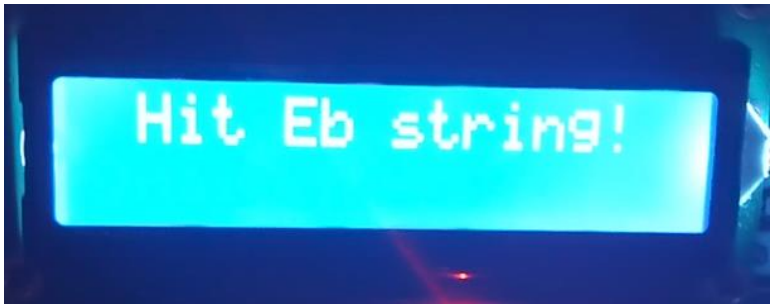
Εικόνα 3.21: Το μενού για την επιλογή κουρδίσματος ή του χειροκίνητου ελέγχου του συστήματος [262].

Βήμα 2ο: Για την εκκίνηση του κουρδίσματος όλων των χορδών της κιθάρας ένα ημίτονο κάτω (Half Step Down Tune), θα πρέπει το βέλος (>) να δείχνει την επιλογή (Half Step Down) με την χρήση των πλήκτρων (UP & DOWN). Έπειτα, ο χρήστης θα πρέπει να οριστικοποιήσει την επιλογή του πιέζοντας το πλήκτρο (SELECT) (Εικόνα 3.22).



Εικόνα 3.22: Επιλογή του κουρδίσματος όλων των χορδών ένα ημίτονο κάτω (Half Step Down Tune) και οριστικοποίηση της επιλογής [263].

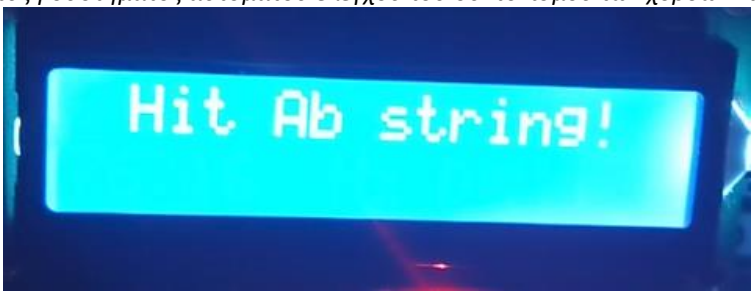
Βήμα 3ο: Στην συνέχεια, εμφανίζεται ένα μήνυμα που προτρέπει τον χρήστη να διεγείρει την χορδή Μιb μπάσο (Eb) (Εικόνα 3.23) και αμέσως μετά εμφανίζεται η επιθυμητή τιμή της συχνότητας ταλάντωσης που πρέπει να έχει η συγκεκριμένη χορδή (Εικόνα 3.24).



Εικόνα 3.23: Το μήνυμα για την διέγερση της χορδής Μιb μπάσο (Eb) [264].

Εικόνα 3.24: Το μήνυμα που εμφανίζει την επιθυμητή συχνότητα ταλάντωσης της χορδής Μιb μπάσο (Eb) [265].

Βήμα 4ο: Αφού ολοκληρωθεί ο συντονισμός της χορδής Μιb μπάσο (Eb), στην οθόνη εμφανίζεται ένα μήνυμα που προτρέπει τον χρήστη να διεγείρει την χορδή Λαb (Ab) (Εικόνα 3.25) και έπειτα, εμφανίζεται η επιθυμητή τιμή της συχνότητας ταλάντωσης που πρέπει να έχει η συγκεκριμένη χορδή (Εικόνα 3.26).



Εικόνα 3.25: Το μήνυμα για την διέγερση της χορδής Λαb (Ab) [266].

Εικόνα 3.26: Το μήνυμα που εμφανίζει την επιθυμητή συχνότητα ταλάντωσης της χορδής Λαb (Ab) [267].

Βήμα 5ο: Μετά την ολοκλήρωση του συντονισμού της χορδής Λαb (Ab), στην οθόνη εμφανίζεται ένα μήνυμα το οποίο προτρέπει τον χρήστη να διεγείρει την χορδή Ρεb (Db) (Εικόνα 3.27) και έπειτα, εμφανίζεται η επιθυμητή τιμή της συχνότητας ταλάντωσης που πρέπει να έχει η συγκεκριμένη χορδή (Εικόνα 3.28).

Εικόνα 3.27: Το μήνυμα για την διέγερση της χορδής Ρεb (Db) [268].

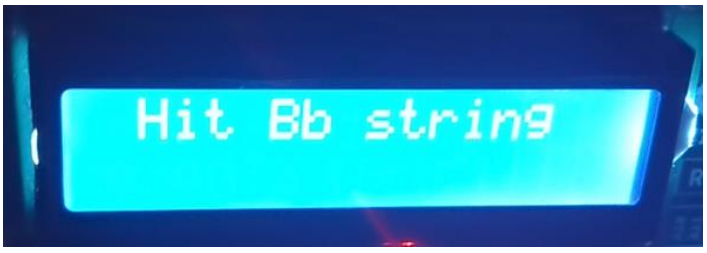
Εικόνα 3.28: Το μήνυμα που εμφανίζει την επιθυμητή συχνότητα ταλάντωσης της χορδής Ρεb (Db) [269].

Βήμα 6ο: Αφού ολοκληρωθεί ο συντονισμός της χορδής Ρεb (Db), στην οθόνη εμφανίζεται ένα μήνυμα που προτρέπει τον χρήστη να διεγείρει την χορδή Σολb (Gb) (Εικόνα 3.29) και έπειτα, εμφανίζεται η επιθυμητή τιμή της συχνότητας ταλάντωσης που πρέπει να έχει η συγκεκριμένη χορδή (Εικόνα 3.30).

Εικόνα 3.29: Το μήνυμα για την διέγερση της χορδής Σολb (Gb) [270].

Εικόνα 3.30: Το μήνυμα που εμφανίζει την επιθυμητή συχνότητα ταλάντωσης της χορδής Σολb (Gb) [271].

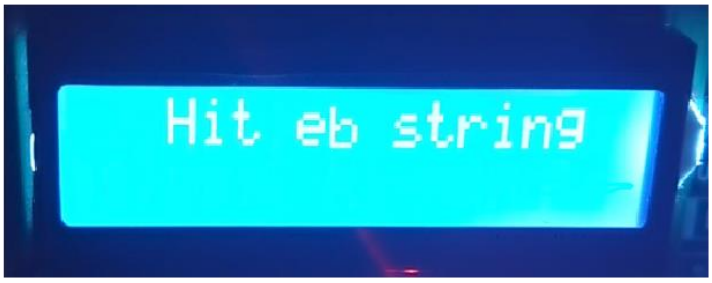
Βήμα 7ο: Μετά την ολοκλήρωση του συντονισμού της χορδής Σολb (Gb), στην LCD οθόνη εμφανίζεται ένα μήνυμα το οποίο προτρέπει τον χρήστη να διεγείρει την χορδή Σιb (Bb) (Εικόνα 3.31) και έπειτα, εμφανίζεται η επιθυμητή τιμή της συχνότητας ταλάντωσης που πρέπει να έχει η συγκεκριμένη χορδή (Εικόνα 3.32).



Εικόνα 3.31: Το μήνυμα για την διέγερση της χορδής Σιb (Bb) [272].

Εικόνα 3.32: Το μήνυμα που εμφανίζει την επιθυμητή συχνότητα ταλάντωσης της χορδής Σιb (Bb) [273].

Βήμα 8ο: Αφού ολοκληρωθεί ο συντονισμός της χορδής Σιb (Bb), τότε εμφανίζεται στην LCD οθόνη ένα μήνυμα το οποίο προτρέπει τον χρήστη να διεγείρει την χορδή Μιb καντίνι (eb) (Εικόνα 3.33) και στην συνέχεια, εμφανίζεται η επιθυμητή τιμή της συχνότητας ταλάντωσης που πρέπει να έχει η συγκεκριμένη χορδή (Εικόνα 3.34).



Εικόνα 3.33: Το μήνυμα για την διέγερση της χορδής Μιb (eb) [274].

Εικόνα 3.34: Το μήνυμα που εμφανίζει την επιθυμητή συχνότητα ταλάντωσης της χορδής Μιb (eb) [275].

Βήμα 9ο: Τέλος, με την ολοκλήρωση του συντονισμού της τελευταίας χορδής Μιb καντίνι (eb), ολοκληρώνεται το κούρδισμα των χορδών της κιθάρας ένα ημίτονο κάτω (Half Step Down Tune), εμφανίζοντας το αντίστοιχο μήνυμα ολοκλήρωσης (Εικόνα 3.35). Ύστερα, στην LCD οθόνη εμφανίζεται το μενού επιλογών (Εικόνα 3.36) και ο χρήστης μπορεί να επιλέξει είτε κάποια νέα ενέργεια, είτε να απενεργοποιήσει το σύστημα μέσω του τριπολικού διακόπτη.

Εικόνα 3.35: Το μήνυμα ολοκλήρωσης του κουρδίσματος των χορδών της κιθάρας ένα ημίτονο κάτω (Half Step Down Tune) [276].

Εικόνα 3.36: Επανεμφάνιση του μενού επιλογών μετά την ολοκλήρωση του κουρδίσματος [277].

3.2.3 Οδηγίες για την επίτευξη του κουρδίσματος της κιθάρας σε «πεσμένη» Ρε (Drop D Tune)

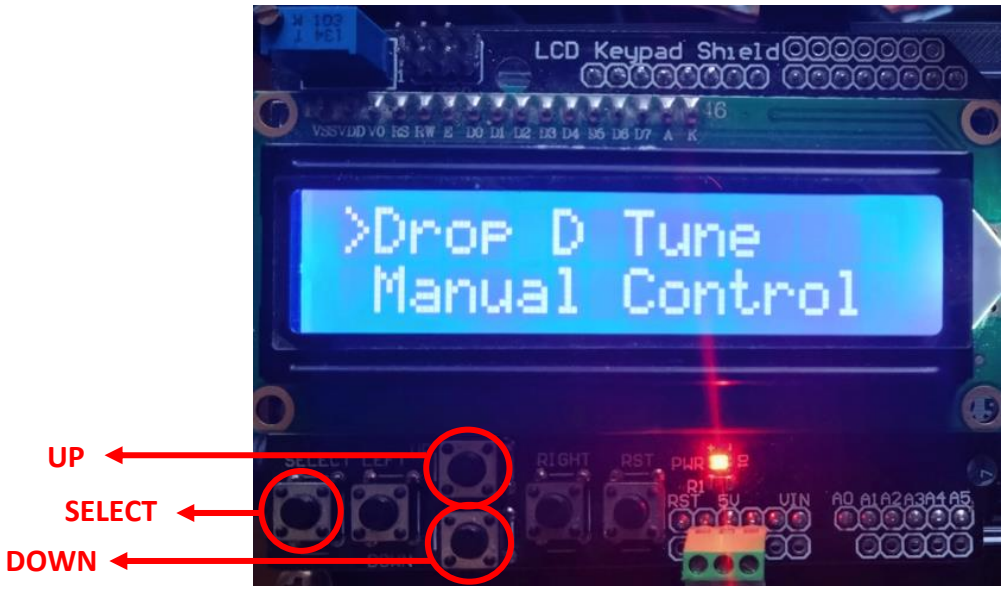
Βήμα 1ο: Ενεργοποίηση του συστήματος με την χρήση του τριπολικού διακόπτη. Αμέσως, ενεργοποιείται το Arduino και κατά συνέπεια η LCD οθόνη. Ο χρήστης, πρώτα θα δει στην LCD το όνομα του συστήματος (Εικόνα 3.37), ύστερα το μήνυμα που τον προτρέπει να επιλέξει κούρδισμα (Εικόνα 3.38) και έπειτα το μενού επιλογή (Εικόνα 3.39).

Εικόνα 3.37: Το όνομα του συστήματος στην LCD οθόνη [278].

Εικόνα 3.38: Το μήνυμα που προτρέπει τον χρήστη να επιλέξει κούρδισμα [279].

Εικόνα 3.39: Το μενού για την επιλογή κουρδίσματος ή του χειροκίνητου ελέγχου του συστήματος [280].

Βήμα 2ο: Για την εκκίνηση του κουρδίσματος της κιθάρας σε «πεσμένη» Ρε (Drop D Tune), θα πρέπει το βέλος (>) να δείχνει την επιλογή (Drop D Tune) με την χρήση των πλήκτρων (UP & DOWN). Έπειτα, ο χρήστης θα πρέπει να οριστικοποιήσει την επιλογή του πιέζοντας το πλήκτρο (SELECT) (Εικόνα 3.40).



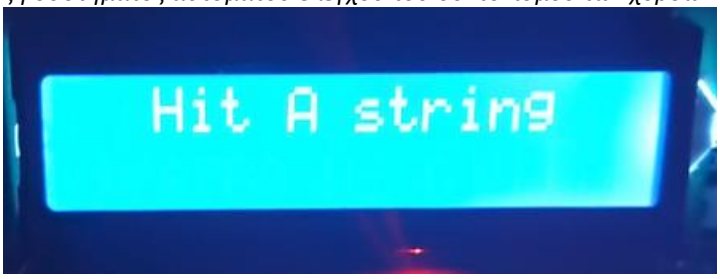
Εικόνα 3.40: Επιλογή του κουρδίσματος σε «πεσμένη» Ρε (Drop D Tune) και οριστικοποίηση της επιλογής [281].

Βήμα 3ο: Στην συνέχεια, εμφανίζεται ένα μήνυμα που προτρέπει τον χρήστη να διεγείρει την χορδή Ρε μπάσο (D) (Εικόνα 3.41) και αμέσως μετά εμφανίζεται η επιθυμητή τιμή της συχνότητας ταλάντωσης που πρέπει να έχει η συγκεκριμένη χορδή (Εικόνα 3.42).

Εικόνα 3.41: Το μήνυμα για την διέγερση της χορδής Ρε μπάσο (D) [282].

Εικόνα 3.42: Το μήνυμα που εμφανίζει την επιθυμητή συχνότητα ταλάντωσης της χορδής Ρε μπάσο (D) [283].

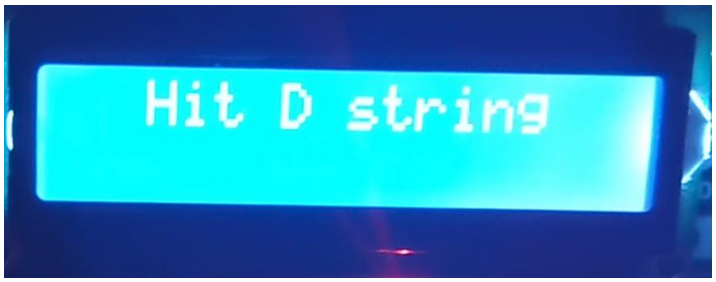
Βήμα 4ο: Αφού ολοκληρωθεί ο συντονισμός της χορδής Ρε μπάσο (D), στην οθόνη εμφανίζεται ένα μήνυμα που προτρέπει τον χρήστη να διεγείρει την χορδή Λα (A) (Εικόνα 3.43) και έπειτα, εμφανίζεται η επιθυμητή τιμή της συχνότητας ταλάντωσης που πρέπει να έχει η συγκεκριμένη χορδή (Εικόνα 3.44).



Εικόνα 3.43: Το μήνυμα για την διέγερση της χορδής Λα (A) [284].

Εικόνα 3.44: Το μήνυμα που εμφανίζει την επιθυμητή συχνότητα ταλάντωσης της χορδής Λα (A) [285].

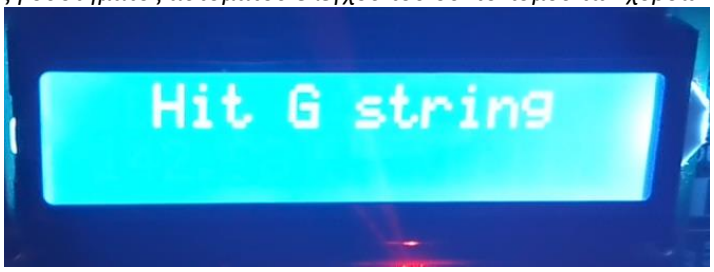
Βήμα 5ο: Μετά την ολοκλήρωση του συντονισμού της χορδής Λα (A), στην οθόνη εμφανίζεται ένα μήνυμα το οποίο προτρέπει τον χρήστη να διεγείρει την χορδή Ρε (D) (Εικόνα 3.45) και έπειτα, εμφανίζεται η επιθυμητή τιμή της συχνότητας ταλάντωσης που πρέπει να έχει η συγκεκριμένη χορδή (Εικόνα 3.46).



Εικόνα 3.45: Το μήνυμα για την διέγερση της χορδής Ρε (D) [286].

Εικόνα 3.46: Το μήνυμα που εμφανίζει την επιθυμητή συχνότητα ταλάντωσης της χορδής Ρε (D) [287].

Βήμα 6ο: Αφού ολοκληρωθεί ο συντονισμός της χορδής Ρε (D), στην οθόνη εμφανίζεται ένα μήνυμα που προτρέπει τον χρήστη να διεγείρει την χορδή Σολ (G) (Εικόνα 3.47) και έπειτα, εμφανίζεται η επιθυμητή τιμή της συχνότητας ταλάντωσης που πρέπει να έχει η συγκεκριμένη χορδή (Εικόνα 3.48).



Εικόνα 3.47: Το μήνυμα για την διέγερση της χορδής Σολ (G) [288].

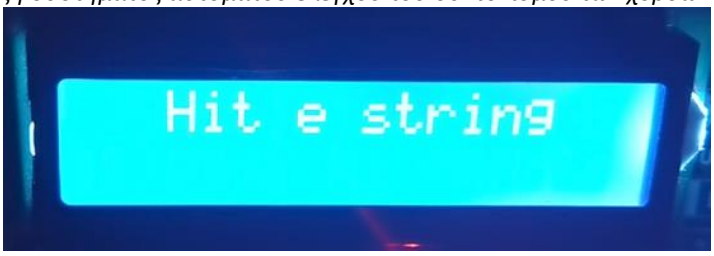
Εικόνα 3.48: Το μήνυμα που εμφανίζει την επιθυμητή συχνότητα ταλάντωσης της χορδής Σολ (G) [289].

Βήμα 7ο: Μετά την ολοκλήρωση του συντονισμού της χορδής Σολ (G), στην LCD οθόνη εμφανίζεται ένα μήνυμα το οποίο προτρέπει τον χρήστη να διεγείρει την χορδή Σι (B) (Εικόνα 3.49) και έπειτα, εμφανίζεται η επιθυμητή τιμή της συχνότητας ταλάντωσης που πρέπει να έχει η συγκεκριμένη χορδή (Εικόνα 3.50).

Εικόνα 3.49: Το μήνυμα για την διέγερση της χορδής Σι (B) [290].

Εικόνα 3.50: Το μήνυμα που εμφανίζει την επιθυμητή συχνότητα ταλάντωσης της χορδής Σι (B) [291].

Βήμα 8ο: Αφού ολοκληρωθεί ο συντονισμός της χορδής Σι (B), τότε εμφανίζεται στην LCD οθόνη ένα μήνυμα το οποίο προτρέπει τον χρήστη να διεγείρει την χορδή Μι καντίνι (e) (Εικόνα 3.51) και στην συνέχεια, εμφανίζεται η επιθυμητή τιμή της συχνότητας ταλάντωσης που πρέπει να έχει η συγκεκριμένη χορδή (Εικόνα 3.52).



Εικόνα 3.51: Το μήνυμα για την διέγερση της χορδής Μι (e) [292].

Εικόνα 3.52: Το μήνυμα που εμφανίζει την επιθυμητή συχνότητα ταλάντωσης της χορδής Μι (e) [293].

Βήμα 9ο: Τέλος, με την ολοκλήρωση του συντονισμού της τελευταίας χορδής Μι καντίνι (e), ολοκληρώνεται το κούρδισμα σε «πεσμένη» Ρε (Drop D Tune), εμφανίζοντας το αντίστοιχο μήνυμα ολοκλήρωσης (Εικόνα 3.53). Ύστερα, στην LCD οθόνη εμφανίζεται το μενού επιλογών (Εικόνα 3.54) και ο χρήστης μπορεί να επιλέξει είτε κάποια νέα ενέργεια, είτε να απενεργοποιήσει το σύστημα μέσω του τριπολικού διακόπτη.

Εικόνα 3.53: Το μήνυμα ολοκλήρωσης του κουρδίσματος της κιθάρας σε «πεσμένη» Ρε (Drop D Tune) [294].

Εικόνα 3.54: Επανεμφάνιση του μενού επιλογών μετά την ολοκλήρωση του κουρδίσματος [295].

3.2.4 Οδηγίες για τον χειροκίνητο έλεγχο του συστήματος

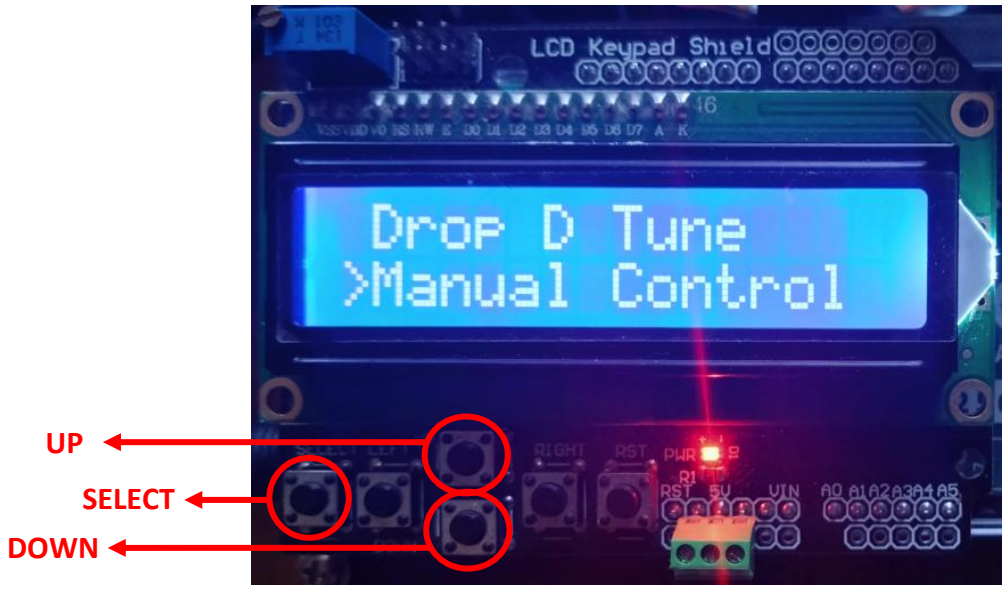
Βήμα 1ο: Ενεργοποίηση του συστήματος με την χρήση του τριπολικού διακόπτη. Αμέσως, ενεργοποιείται το Arduino και κατά συνέπεια η LCD οθόνη. Ο χρήστης, πρώτα θα δει στην LCD το όνομα του συστήματος (Εικόνα 3.55), ύστερα το μήνυμα που τον προτρέπει να επιλέξει κούρδισμα (Εικόνα 3.56) και έπειτα το μενού επιλογή (Εικόνα 3.57).

Εικόνα 3.55: Το όνομα του συστήματος στην LCD οθόνη [296].

Εικόνα 3.56: Το μήνυμα που προτρέπει τον χρήστη να επιλέξει κούρδισμα [297].

Εικόνα 3.57: Το μενού για την επιλογή κουρδίσματος ή του χειροκίνητου ελέγχου του συστήματος [298].

Βήμα 2ο: Για την εκκίνηση του χειροκίνητου ελέγχου του συστήματος (Manual Control), θα πρέπει το βέλος (>) να δείχνει την επιλογή (MANUAL CONTROL) με την χρήση των πλήκτρων (UP & DOWN). Έπειτα, ο χρήστης θα πρέπει να οριστικοποιήσει την επιλογή του πιέζοντας το πλήκτρο (SELECT) (Εικόνα 3.58).



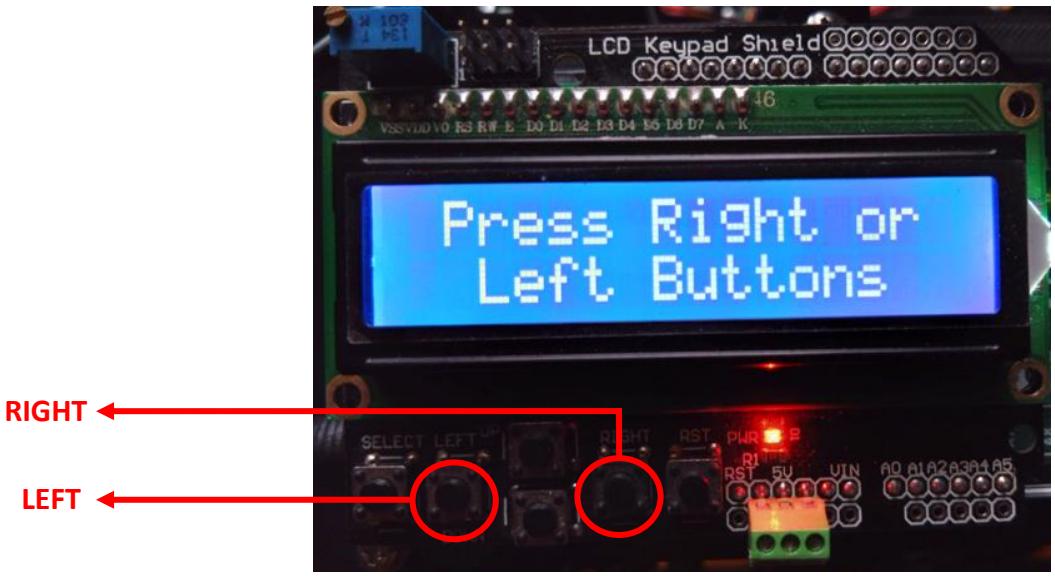
Εικόνα 3.58: Επιλογή του χειροκίνητου ελέγχου του συστήματος (Manual Control) και οριστικοποίηση της επιλογής [299].

Βήμα 3ο: Στην συνέχεια, εμφανίζεται ένα μήνυμα το οποίο προτρέπει τον χρήστη να επιλέξει από το μενού χειροκίνητου ελέγχου την χορδή που επιθυμεί να ελέγξει (Εικόνα 3.59). Για τον χειροκίνητο έλεγχο οποιασδήποτε χορδής, θα πρέπει πρώτα ο χρήστης να επιλέξει με το βελάκι (>) την επιθυμητή χορδή κάνοντας χρήση των πλήκτρων (UP, DOWN, LEFT & RIGHT) και στην συνέχεια να οριστικοποιήσει την επιλογή του με το πλήκτρο (SELECT) (Εικόνα 3.60).

Εικόνα 3.59: Το μήνυμα που προτρέπει τον χρήστη να επιλέξει την χορδή που επιθυμεί να ελέγξει [300].

Εικόνα 3.60: Επιλογή χορδών και οριστικοποίηση επιλογής στο μενού χειροκίνητου ελέγχου [301].

Βήμα 4ο: Αφού οριστικοποιηθεί η επιλογή του, τότε εμφανίζεται ένα μήνυμα που τον προτρέπει να χρησιμοποιήσει τα πλήκτρα (LEFT & RIGHT) για τον χειροκίνητο έλεγχο του βηματικού κινητήρα της αντίστοιχης χορδής (Εικόνα 3.61). Έπειτα, εμφανίζεται ένα μήνυμα που ενημερώνει τον χρήστη ότι πρέπει να πιέσει το πλήκτρο (SELECT) για να ολοκληρώσει των χειροκίνητο έλεγχο της χορδής και να επιστρέψει στο προηγούμενο μενού (Εικόνα 3.62).



Εικόνα 3.61: Το μήνυμα που ενημερώνει τον χρήστη για τα πλήκτρα με τα οποία μπορεί να ελέγξει την περιστροφή του βηματικού κινητήρα της χορδής που έχει επιλέξει [302].

Εικόνα 3.62: Το μήνυμα που ενημερώνει τον χρήστη για το πως μπορεί να ολοκληρώσει τον χειροκίνητο έλεγχο της χορδής που έχει επιλέξει [303].

Βήμα 5ο: Στην περίπτωση που ο χρήστης επιλέξει μέσω των κουμπιών (UP, DOWN, LEFT & RIGHT) την επιλογή επιστροφής (Back) (Εικόνα 3.63) και οριστικοποιήσει την επιλογή του μέσω του πλήκτρου (SELECT), τότε εμφανίζεται στην LCD οθόνη το αρχικό μενού (Εικόνα 3.64).

Εικόνα 3.63: Επιλογή και οριστικοποίηση της επιλογής (Back) στο μενού χειροκίνητου ελέγχου [304].

Εικόνα 3.64: Το αρχικό μενού για την επιλογή κουρδίσματος ή του χειροκίνητου ελέγχου του συστήματος [305].

Βήμα 6ο: Τέλος, ο χρήστης σε αυτό το σημείο έχει την επιλογή είτε να επιλέξει κάποια ενέργεια μέσα από το μενού, είτε να απενεργοποιήσει το σύστημα μέσω του τριπολικού διακόπτη.

4 ΣΥΜΠΕΡΑΣΜΑΤΑ

Αρχικά, η συγκεκριμένη διπλωματική εργασία αποτέλεσε το κίνητρο για την ασχολία με αρκετά πρωτόγνωρα γεγονότα. Πιο συγκεκριμένα, η επαφή με την τρισδιάστατη εκτύπωση, η λογική πίσω από τον τρόπο σχεδίασης των τρισδιάστατων κομματιών στήριξης, η κατασκευή μιας στοιχειώδους πλακέτας για την ενίσχυση του σήματος από το μικρόφωνο επαφής, καθώς και το κομμάτι του κώδικα προγραμματισμού του μικροελεγκτή άνοιξαν νέους ορίζοντες. Ουσιαστικά, η εκπόνηση της συγκεκριμένης διπλωματικής εργασίας, αποτέλεσε σε όλη της την πορεία τεράστια εμπειρία και γνώση για το αντικείμενο.

Όσον αφορά το συγκεκριμένο σύστημα αυτομάτου ελέγχου για τον συντονισμό των χορδών της κιθάρας, μπορεί να ειπωθεί ότι καλύπτει τους στόχους του σε έναν μεγάλο βαθμό. Ο πρώτος και μάλιστα καίριας σημασίας στόχος που επιτυγχάνεται, αποσκοπεί στην ασφάλεια του χρήστη, στην ασφάλεια του συστήματος και στην λειτουργικότητα του συστήματος. Πιο συγκεκριμένα, ο τρόπος σχεδίασης και λειτουργίας του συστήματος είναι τέτοιος που δεν εκθέτει σε κίνδυνο ούτε τον χρήστη, ούτε το ίδιο το σύστημα. Επίσης, η οικονομική σχεδίαση και ανάπτυξη του συστήματος αποτελεί έναν εξίσου σημαντικό στόχο που επιτεύχθηκε σε αυτή την διπλωματική εργασία. Πιο αναλυτικά, στο Παράρτημα Α παρουσιάζεται ένας πίνακας με το κόστος κατασκευής του συγκεκριμένου συστήματος. Επιπροσθέτως, το σύστημα αυτομάτου ελέγχου για τον συντονισμό των χορδών της κιθάρας, ενεργειακά έχει μεγάλη αυτονομία. Αυτό οφείλεται, στις ξεχωριστές τροφοδοσίες που έχει το κάθε υποσύστημα. Με αυτόν τον τρόπο, το σύστημα προσφέρει μεγάλο αριθμό κουρδισμάτων με μία μόνο φόρτιση. Ακόμα, έχει επιτευχθεί ο στόχος που αφορά την ευελιξία του συστήματος. Πιο αναλυτικά, η συγκεκριμένη συσκευή δεν έχει κατασκευαστεί για να πραγματοποιεί ένα μόνο κούρδισμα, αλλά ο χρήστης έχει την δυνατότητα επιλογής μέσα από ένα μενού τεσσάρων ενεργειών. Δηλαδή, το σύστημα παρέχει τρεις επιλογές κουρδισμάτων και μία επιλογή για τον χειροκίνητο έλεγχο του συστήματος σε περίπτωση που ο χρήστης επιθυμεί να αλλάξει χορδές στην κιθάρα. Επιπλέον, είναι σημαντικό να αναφερθεί η ακρίβεια του κουρδίσματος που πετυχαίνει το σύστημα. Πιο συγκεκριμένα, στο κλασσικό κούρδισμα, η χορδή Μι μπάσο (E), κουρδίζεται εντός τους εύρους συχνοτήτων ±0.06Hz από την επιθυμητή τιμή της συχνότητας ταλάντωσης που πρέπει να έχει η συγκεκριμένη χορδή. Όσον αφορά την χορδή Λα (A), κουρδίζεται εντός του εύρους συχνοτήτων ±0.2Hz από την επιθυμητή τιμή της συχνότητας ταλάντωσης που πρέπει να έχει η συγκεκριμένη χορδή. Επίσης, η χορδή Ρε (D), κουρδίζεται εντός του εύρους συχνοτήτων ±0.1 από την επιθυμητή τιμή της συχνότητας ταλάντωσης που πρέπει να έχει η συγκεκριμένη χορδή. Ακόμα, η χορδή Σολ (G), κουρδίζεται εντός του εύρους συχνοτήτων -0.1Hz και +0.25Hz από την επιθυμητή τιμή της συχνότητας ταλάντωσης της συγκεκριμένης χορδής. Επίσης, η χορδή Σι (B), κουρδίζεται εντός του εύρους -0.39Hz και +0.1Hz από την επιθυμητή τιμή της συχνότητας ταλάντωσης που πρέπει να έχει η συγκεκριμένη χορδή. Ενώ, η χορδή Μι καντίνι (e), κουρδίζεται εντός του εύρους συχνοτήτων -0.9Hz και +1.95Hz από την επιθυμητή τιμή της συχνότητας ταλάντωσης που πρέπει να έχει η συγκεκριμένη χορδή. Όσον αφορά το κούρδισμα όλων των χορδών της κιθάρας ένα ημίτονο κάτω (Half Step Down Tune), η ακρίβεια με την οποία επιτυγχάνεται είναι αρκετά ικανοποιητική. Πιο αναλυτικά, η χορδή Μιb μπάσο (Eb), κουρδίζεται εντός τους εύρους συχνοτήτων ±0.1Hz από την επιθυμητή τιμή της συχνότητας ταλάντωσης που πρέπει να έχει η συγκεκριμένη χορδή. Επίσης, η

χορδή Λαb (Ab), κουρδίζεται εντός του εύρους συχνοτήτων -0.17Hz και +0.13Hz από την επιθυμητή τιμή της συχνότητας ταλάντωσης που πρέπει να έχει η συγκεκριμένη χορδή. Επιπλέον, η χορδή Ρεb (Db), κουρδίζεται εντός του εύρους συχνοτήτων -0.24Hz και 0.26Hz από την επιθυμητή τιμή της συχνότητας ταλάντωσης που πρέπει να έχει η συγκεκριμένη χορδή. Επιπροσθέτως, η χορδή Σολb (Gb), κουρδίζεται εντός του εύρους συχνοτήτων ±0.1Hz από την επιθυμητή τιμή της συχνότητας ταλάντωσης που πρέπει να έχει η συγκεκριμένη χορδή. Ακόμα, η χορδή Σιb (Bb), κουρδίζεται εντός του εύρους ±0.1Hz από την επιθυμητή τιμή της συχνότητας ταλάντωσης της συγκεκριμένης χορδής. Ωστόσο, η χορδή Μιb καντίνι (eb), κουρδίζεται εντός του εύρους συχνοτήτων -0.95Hz και +1.57Hz από την επιθυμητή τιμή της συχνότητας ταλάντωσης που πρέπει να έχει η συγκεκριμένη χορδή. Όσον αφορά τώρα το κούρδισμα των χορδών της κιθάρας σε «πεσμένη» Ρε (Drop D Tune), η ακρίβεια ακόμη και εδώ είναι αρκετά ικανοποιητική. Πιο συγκεκριμένα, η χορδή Ρε μπάσο (D), κουρδίζεται εντός τους εύρους συχνοτήτων ±0.05Hz από την επιθυμητή τιμή της συχνότητας ταλάντωσης που πρέπει να έχει η συγκεκριμένη χορδή. Όσον αφορά την χορδή Λα (A), κουρδίζεται εντός του εύρους συχνοτήτων ±0.2Hz από την επιθυμητή τιμή της συχνότητας ταλάντωσης που πρέπει να έχει η συγκεκριμένη χορδή. Επίσης, η χορδή Ρε (D), κουρδίζεται εντός του εύρους συχνοτήτων ±0.1 από την επιθυμητή τιμή της συχνότητας ταλάντωσης που πρέπει να έχει η συγκεκριμένη χορδή. Ακόμα, η χορδή Σολ (G), κουρδίζεται εντός του εύρους συχνοτήτων -0.1Hz και +0.25Hz από την επιθυμητή τιμή της συχνότητας ταλάντωσης της συγκεκριμένης χορδής. Επίσης, η χορδή Σι (B), κουρδίζεται εντός του εύρους -0.39Hz και +0.1Hz από την επιθυμητή τιμή της συχνότητας ταλάντωσης που πρέπει να έχει η συγκεκριμένη χορδή. Ενώ, η χορδή Μι καντίνι (e), κουρδίζεται εντός του εύρους συχνοτήτων -0.9Hz και +1.95Hz από την επιθυμητή τιμή της συχνότητας ταλάντωσης που πρέπει να έχει η συγκεκριμένη χορδή. Ένας ακόμα στόχος που επιτεύχθηκε, αφορά την σχεδόν ακαριαία ανταπόκριση του συστήματος όταν διεγερθεί η κατάλληλη χορδή. Πιο συγκεκριμένα, το σύστημα μετά τον υπολογισμό της συχνότητας ταλάντωσης της διεγερμένης χορδής, ενεργοποιεί σχεδόν ακαριαία τον βηματικό κινητήρα περιστρέφοντας ανάλογα το κλειδί της αντίστοιχης χορδής. Επίσης, το συγκεκριμένο σύστημα αυτομάτου ελέγχου για τον συντονισμό των χορδών της κιθάρας επιτυγχάνει την εξυπηρέτηση των αρχάριων μουσικών μιας και το κούρδισμα γι αυτούς αποτελεί δύσκολη και χρονοβόρα διαδικασία. Με αυτόν τον τρόπο, το συγκεκριμένο σύστημα επιτυγχάνει έναν ακόμη στόχο που έχει τεθεί. Επιπλέον, η συγκεκριμένη διπλωματική εργασία έχει καινοτομίες οι οποίες δεν υπήρχαν σε παρόμοιες διπλωματικές που μελετήθηκαν. Πιο συγκεκριμένα, μία καινοτομία είναι η LCD οθόνη με το ενσωματωμένο πληκτρολόγιο, το οποίο αποτελεί την διεπαφή μεταξύ χρήστη και συστήματος. Επίσης, μία άλλη καινοτομία που περιέχει το συγκεκριμένο σύστημα, είναι οι έξι βηματικοί κινητήρες ώστε το κούρδισμα της κάθε χορδής να γίνεται αυτόνομα. Με αυτόν τον τρόπο, η εμπλοκή του χρήστη μειώνεται στο ελάχιστο ενώ η ταχύτητα κουρδίσματος αυξάνεται. Επιπλέον, μία εξίσου σημαντική καινοτομία είναι η δυνατότητα της επιλογής που δίνει το σύστημα στον χρήστη. Ουσιαστικά, ο χρήστης έχει την δυνατότητα να επιλέξει ανάμεσα σε τρία κουρδίσματα που περιλαμβάνει το σύστημα καθώς και στην επιλογή του χειροκίνητου ελέγχου. Ο χειροκίνητος έλεγχος που παρέχει το σύστημα αποτελεί μία ακόμα σημαντική καινοτομία, διότι ο χρήστης μπορεί με αυτόν τον τρόπο να αλλάξει τις χορδές της κιθάρας χωρίς την απεγκατάσταση του συστήματος από αυτήν. Ακόμα, η εκπόνηση της συγκεκριμένης εργασίας, επιτυγχάνει εξάλειψη του θορύβου από το περιβάλλον, με την χρήση του

μικροφώνου επαφής Korg CM300. Με αυτόν τον τρόπο, το κούρδισμα είναι πιο ακριβές και αξιόπιστο.

Από την άλλη πλευρά όμως, υπήρξαν αρκετές δυσκολίες, εμπόδια και στόχοι που κατά την εκπόνηση της συγκεκριμένης εφαρμογής. Πιο αναλυτικά, έγιναν πολλές τρισδιάστατες εκτυπώσεις των βάσεων και των συνδέσμων στήριξης μέχρι να επιτευχθεί το επιθυμητό αποτέλεσμα. Επίσης, οι βάσεις αποτελούν τα μεγαλύτερα κομμάτια στήριξης του συστήματος και χρειάστηκε αρκετός χρόνο για την ολοκλήρωση της τρισδιάστατης εκτύπωσης της κάθε μίας. Επιπροσθέτως, η τρισδιάστατη εκτύπωση αποτέλεσε μία νέα εμπειρία και χρειάστηκε χρόνος μέχρι την προσαρμογή. Έτσι, ένα μεγάλο μέρος του χρόνου αφιερώθηκε στο κομμάτι της τρισδιάστατης σχεδίασης και της τρισδιάστατης εκτύπωσης των κομματιών στήριξης και σύνδεσης. Ακόμα, το μεγαλύτερο κομμάτι του χρόνου αφιερώθηκε στην μελέτη παρόμοιων διπλωματικών εργασιών καθώς και στην εύρεση των υλικών που χρειάστηκαν για την υλοποίηση του συγκεκριμένου συστήματος. Επιπλέον, για την δημιουργία και την δοκιμή του κάθε υπό-συστήματος ξεχωριστά, καθώς και για την σύνδεση όλων μαζί, αφιερώθηκε ένα σημαντικό μέρος του χρόνου. Ακόμα και ο προγραμματισμός όλου του συστήματος, ήταν κάτι πρωτόγνωρο και πολύπλοκο, με αποτέλεσμα να ξοδευτεί και εκεί αρκετός χρόνος. Επιπροσθέτως, ο τρόπος στήριξης του συγκεκριμένου συστήματος επάνω στην κιθάρα είναι αρκετά ογκώδες γεγονός το οποίο δυσκολεύει τον χρήστη να δει στην LCD οθόνη. Επίσης, η χρήση των τριών παροχών τροφοδοσίας σε συνδυασμό με τον όγκο του συστήματος, προσθέτουν στην κεφαλή της κιθάρας ένα βάρος σχετικά αισθητό. Ακόμα, χρησιμοποιείται ένας σημαντικός αριθμός βιδών, ροδελών και παξιμαδιών ώστε να επιτευχθεί η στιβαρότητα του συστήματος. Αυτό έχει ως αποτέλεσμα την προσθήκη ακόμα περισσότερου βάρους στην κεφαλή της κιθάρας. Επιπλέον, το ο τρόπος στήριξης του συστήματος επάνω στα κλειδιά, δημιουργεί ένα πρόβλημα στην περιστροφή των βηματικών κινητήρων κυρίως στα κλειδιά Σολ (G) και Ρε (D). Πιο συγκεκριμένα, η περιστροφή των συγκεκριμένων βηματικών κινητήρων και ιδιαίτερα κατά το σφίξιμο των αντίστοιχων χορδών, γίνεται πολύ αργά. Αυτό έχει ως αποτέλεσμα, να αυξάνεται ο χρόνος κουρδίσματος των χορδών εξαιτίας της αργής περιστροφής των δύο αυτών βηματικών κινητήρων. Επίσης, οι χορδές Σολ (G), Σι (B) και Μι καντίνι (e) κατά την διαδικασία οποιουδήποτε κουρδίσματος, θα πρέπει να διεγείρονται με σχετικά αρκετή δύναμη ώστε το σήμα από το μικρόφωνο επαφής να έχει όσο το δυνατόν μεγαλύτερο πλάτος. Επιπλέον, πρέπει να σημειωθεί ότι ο συγκεκριμένος τρόπος στήριξης και συναρμολόγησης του συστήματος επάνω στην κιθάρα είναι μια πολύπλοκη και χρονοβόρα διαδικασία. Αυτό οφείλεται στο πλήθος των τρισδιάστατων κομματιών, των βιδών και των καλωδίων που περιλαμβάνει το συγκεκριμένο σύστημα. Ένα ακόμα μειονέκτημα του συγκεκριμένου συστήματος είναι ότι δεν μπορεί να τοποθετηθεί σε άλλη κιθάρα. Αυτό οφείλεται στις συγκεκριμένες διαστάσεις και αποστάσεις που έχουν τα κλειδιά μεταξύ τους. Θα μπορούσε όμως, το σύστημα αυτό να τοποθετηθεί σε μία άλλη κιθάρα που θα έχει τα ίδια χαρακτηριστικά με αυτήν που χρησιμοποιήθηκε για την υλοποίηση της συγκεκριμένης πτυχιακής.

Μελλοντικά, το συγκεκριμένο σύστημα, με κάποιες βελτιώσεις, θα μπορούσε να γίνει προϊόν και να κυκλοφορήσει στην αγορά. Πιο αναλυτικά, με την χρήση νέων τεχνολογιών, θα μπορούσε να μικρύνει ο όγκος του συστήματος και να γίνει πιο καλαίσθητο. Ουσιαστικά, μία καθοριστική βελτίωση που θα αποσκοπούσε στην περαιτέρω μείωση του όγκου του συστήματος, θα αποτελούσε η σχεδίαση μιας πολύ μικρότερης πλακέτας. Πιο συγκεκριμένα, η πλακέτα αυτή θα περιλαμβάνει τον

μικροελεγκτή, τους οδηγούς των βηματικών κινητήρων, το κύκλωμα ενίσχυσης και μετατόπισης του σήματος από το μικρόφωνο επαφής και φίσες ή κλέμες για την σύνδεση της παροχής και των υπόλοιπων υποσυστημάτων. Αυτό θα έχει ως αποτέλεσμα, την δημιουργία μικρότερων βάσεων στήριξης του συστήματος επάνω στην κιθάρα. Έτσι, θα μειωθεί τόσο ο όγκος όσο και το συνολικό βάρος του συστήματος. Επίσης, μία σημαντική βελτίωση θα ήταν η χρήση πιο μικρών βηματικών κινητήρων με μεγαλύτερη ροπή κατά την περιστροφή. Με αυτόν τον τρόπο, τα κλειδιά των χορδών θα περιστρέφονται πιο εύκολα και θα μειωθεί κατά ένα μεγάλο ποσοστό το συνολικό βάρος του συστήματος. Επίσης, μία εξίσου σημαντική βελτίωση, αποτελεί η χρήση μίας πηγής για την συνολική τροφοδοσία του συστήματος. Έτσι επιτυγχάνεται, περαιτέρω μείωση του όγκου και του συνολικού βάρους του συστήματος. Ωστόσο, θα πρέπει να ληφθεί υπ' όψιν ότι το κάθε ένα υποσύστημα λειτουργεί με διαφορετική τιμή τάσης. Συνεπώς σε αυτήν την βελτίωση, θα πρέπει να συμπεριληφθεί η σχεδίαση και η ανάπτυξη των αναγκαίων ηλεκτρονικών κυκλωμάτων για την πτώση της τάσης στην τιμή που απαιτεί το κάθε υποσύστημα. Επιπροσθέτως, μία εξίσου σημαντική βελτίωση του συστήματος θα αποτελούσε η προσθήκη περισσότερων κουρδισμάτων. Ακόμα, μία πολύ καλή ιδέα για το συγκεκριμένο σύστημα, θα αποτελούσε η δυνατότητα δημιουργίας και αποθήκευσης προσωπικών κουρδισμάτων. Επιπλέον, μία σημαντική εξέλιξη του συστήματος θα μπορούσε να είναι η προσθήκη Bluetooth ή Wi-Fi. Με αυτόν τον τρόπο, ο χρήστης θα μπορούσε να κατεβάσει την αντίστοιχη εφαρμογή στο smart-phone του και να δημιουργήσει πιο εύκολα τα προσωπικά του κουρδίσματα. Επίσης, μία σημαντική βελτίωση θα μπορούσε να αποτελέσει μια πιο έξυπνη σχεδίαση για την στήριξη του συστήματος επάνω στην κεφαλή της κιθάρας. Πιο αναλυτικά, θα μπορούσε να σχεδιαστεί μία μορφή στήριξης, η οποία θα αλλάζει το σχηματισμό της ανάλογα με την κιθάρα στην οποία θα εφαρμοστεί. Με αυτό τον τρόπο, ο χρήστης θα μπορεί να τοποθετήσει το σύστημα αυτομάτου ελέγχου για τον συντονισμό των χορδών σε οποιαδήποτε από τις κιθάρες του επιθυμεί. Επιπλέον, η συγκεκριμένη διπλωματική εργασία θα μπορούσε να αποτελέσει έμπνευση καθώς και την αρχή για την σχεδίαση και την ανάπτυξη παρόμοιων συστημάτων για άλλα έγχορδα όργανα. Πιο συγκεκριμένα, θα μπορούσε να σχεδιαστεί ένα αντίστοιχο σύστημα για κιθάρες με περισσότερες χορδές, για μπάσο, για μπουζούκι και για λαούτο. Τέλος, η συγκεκριμένη διπλωματική εργασία, θα μπορούσε να αποτελέσει την βάση για την σχεδίαση και την ανάπτυξη παρόμοιων συστημάτων αυτομάτου ελέγχου για μεγαλύτερα έγχορδα μουσικά όργανα. Πιο αναλυτικά, το σύστημα αυτό σαφώς θα είναι πολύ μεγαλύτερο από αυτό της συγκεκριμένης διπλωματικής εργασίας, αλλά θα μπορούσε να εφαρμοστεί σε πιάνα, ή σε κανονάκια ή ακόμα και σε σαντούρι. Ένα τέτοιο σύστημα θα εξυπηρετούσε τέτοιου είδους όργανα διότι το κούρδισμά τους είναι δύσκολο και αρκετά χρονοβόρο. Συνεπώς, η δημιουργία ενός τέτοιου συστήματος για το κούρδισμα τέτοιων οργάνων, θα επωφελούνται όχι μόνο οι αρχάριοι, αλλά και οι επαγγελματίες μουσικοί.

Βιβλιογραφία – Αναφορές - Διαδικτυακές Πηγές

[1] Αμαλία Κ. Ηλιάδη, «[άλλα \'μουσικολογικά\'] Η μουσική στην Αρχαία Ελλάδα (της Αμαλίας Ηλιάδη)», tar.gr. [Online]. Available at: https://www.tar.gr/alla_moysikologika_moysiki_stin_arxaia_ellada_br_tis_amalias_iliadi-article-729.html?category_id=13 [Accessed 15 March 2024]

[2] «Τι είναι κιθάρα;», ti-einai.gr, [Online] Available at:https://ti-einai.gr/kithara/ [Accessed 15 March 2024]

[3] «Κιθάρα», el.wikipedia.org/wiki/Πύλη:Κύρια. [Online]. Available at: https://el.wikipedia.org/wiki/%CE%9A%CE%B9%CE%B8%CE%AC%CF%81%CE%B1 [Accessed 15 March 2024]

[4] Ελένη Πάλλη, «Μουσικά όργανα της αρχαίας Ελλάδας», e-telescope.gr/. [Online]. Available at: https://www.e-telescope.gr/arts/music-in-ancient-greece [Accessed 15 March 2024]

[5] «Αρχαία ελληνική μουσική», el.wikipedia.org/wiki/Πύλη:Κύρια. [Online]. Available at:https://el.wikipedia.org/wiki/%CE%91%CF%81%CF%87%CE%B1%CE%AF%CE%B1_%CE%B5%CE%BB%CE%BB%CE%B7%CE%BD%CE%B9%CE%BA%CE%AE_%CE%BC%CE%BF%CF%85%CF%83%CE%B9%CE%BA%CE%AE [Accessed 15 March 2024]

[6] «Κιθάρα (αρχαιοελληνική μουσική)», el.wikipedia.org/wiki/Πύλη:Κύρια. [Online]. Available at: https://el.wikipedia.org/wiki/%CE%9A%CE%B9%CE%B8%CE%AC%CF%81%CE%B1_(%CE%B1%CF%81%CF%87%CE%B1%CE%B9%CE%BF%CE%B5%CE%BB%CE%BB%CE%B7%CE%BD%CE%B9%CE%BA%CE%AE_%CE%BC%CE%BF%CF%85%CF%83%CE%B9%CE%BA%CE%AE) [Accessed 15 March 2024]

[7] "Vihuela", en.wikipedia.org/wiki/Main_Page. [Online]. Available at: https://en.wikipedia.org/wiki/Vihuela [Accessed 16 March 2024]

[8] «βιχουέλα», el.wiktionary.org/wiki/Βικιλεξικό:Κύρια_Σελίδα. [Online]. Available at:https://el.wiktionary.org/wiki/%CE%B2%CE%B9%CF%87%CE%BF%CF%85%CE%AD%CE%BB%CE%B1 [Accessed 16 March 2024]

[9] «Ηλεκτρική κιθάρα», el.wikipedia.org/wiki/Πύλη:Κύρια. [Online]. Available at: https://el.wikipedia.org/wiki/%CE%97%CE%BB%CE%B5%CE%BA%CF%84%CF%81%CE%B9%CE%BA%CE%AE_%CE%BA%CE%B9%CE%B8%CE%AC%CF%81%CE%B1 [Accessed 17 March 2024]

[10] Λεωνίδας Πάσιος, "Electric Guitar History, Η Ιστορία Της Ηλεκτρικής Κιθάρας", rockvision.gr. [Online]. Available at: https://rockvision.gr/2017/12/08/electric-guitar-history/ [Accessed 17 March 2024]

[11] "Electric guitar", en.wikipedia.org/wiki/Main_Page. [Online]. Available at: https://en.wikipedia.org/wiki/Electric_guitar [Accessed 17 March 2024]

[12] «Η ιστορία της ηλεκτρικής κιθάρας», rockwave.gr. [Online]. Available at: https://rockwave.gr/i-istoria-tis-ilektrikis-kitharas/ [Accessed 20 March 2024]

[13] Chris Fleisher, "Closer Look: Going Electric", carnegiemuseums.org. [Online]. Available at: https://carnegiemuseums.org/carnegie-magazine/summer-2022/objects-of-our-affection-going-electric/ [Accessed 17 March 2024]

[14] "Frying Pan (guitar)", en.wikipedia.org/wiki/Main_Page . [Online]. Available at: https://en.wikipedia.org/wiki/Frying_Pan_%28guitar%29 [Accessed 17 March 2024]

[15] "Rickenbacker", en.wikipedia.org/wiki/Main_Page. [Online]. Available at: https://en.wikipedia.org/wiki/Rickenbacker [Accessed 17 March 2024]

[16] "George Beaushamp", en.wikipedia.org/wiki/Main_Page. [Online]. Available at: https://en.wikipedia.org/wiki/George_Beauchamp [Accessed 17 March 2024]

[17] "Lap steel guitar", en.wikipedia.org/wiki/Main_Page. [Online]. Available at: https://en.wikipedia.org/wiki/Lap_steel_guitar [Accessed 19 March 2024]

[18] "National String Instrument Corporation", en.wikipedia.org/wiki/Main_Page. [Online]. Available at: https://en.wikipedia.org/wiki/National_String_Instrument_Corporation#Dobro [Accessed 19 March 2024]

[19] "Dobro", en.wikipedia.org/wiki/Main_Page. [Online]. Available at: https://en.wikipedia.org/wiki/Dobro [Accessed 19 March 2024]

[20] Tom Gray, "A Guide to Vintage Dobros", vintageguitar.com. [Online]. Available at: https://www.vintageguitar.com/30152/a-guide-to-vintage-dobros/ [Accessed 19 March 2024]

[21] "National/Dobro History", chasingguitars.com. [Online]. Available at: https://chasingguitars.com/nationaldobro-history/ [Accessed 19 March 2024]

[22] "Resonator Guitar", en.wikipedia.org/wiki/Main_Page. [Online]. Available at: https://en.wikipedia.org/wiki/Resonator_guitar [Accessed 19 March 2024]

[23] "PAUL TUTMARC & The Mystery of Who Invented The Electric Guitar", jivetimerecords.com. [Online]. Available at: https://jivetimerecords.com/northwest/paula-tutmarc/ [Accessed 21 March 2024]

[24] Peter Blecha, "Tutmarc, Paul (1896-1972) and his Audiovox Electric Guitars", historylink.org. [Online]. Available at: https://www.historylink.org/File/7479 [Accessed 21 March 2024]

[25] "Paul Tutmarc", en.wikipedia.org/wiki/Main_Page. [Online]. Available at: https://en.wikipedia.org/wiki/Paul_Tutmarc [Accessed 21 March 2024]

[26] John Teagle, "Antique Guitar Amps 1928-1934", vintageguitar.com. [Online]. Available at: https://www.vintageguitar.com/1885/antique-guitar-amps-1928-1934-2/ [Accessed 21 March 2024]

[27] "Vega Vegaphone C-75 Arch Top Acoustic Guitar , c. 1938", retrofret.com. [Online]. Available at: https://retrofret.com/product.asp?ProductID=7911 [Accessed 21 March 2024]

[28] "Vega Company", en.wikipedia.org/wiki/Main_Page. [Online]. Available at: https://en.wikipedia.org/wiki/Vega_Company [Accessed 21 March 2024]

[29] "A Brief History of Epiphone Guitars", guitar.com. [Online]. Available at: https://guitar.com/features/a-brief-history-of-epiphone-guitars/ [Accessed 22 March 2024]

[30] "Epiphone", en.wikipedia.org/wiki/Main_Page. [Online]. Available at: https://en.wikipedia.org/wiki/Epiphone [Accessed 22 March 2024]

[31] "Epiphone Recording D Archtop with cutaway – U.S.A. – 1928", theguitardatabase.com. [Online]. Available at: https://www.theguitardatabase.com/main_database/epiphone-recording-d-archtop-with-cutaway-u-s-a-1928/ [Accessed 22 March 2024]

[32] "1931 Epiphone Broadway Masterbilt", archtop.com. [Online]. Available at: https://www.archtop.com/ac_31bwy.html [Accessed 22 March 2024]

[33] "1935 Epiphone Electar Electrophone Lap Steel Guitar w/ OHSC", reverb.com/en-gr/. [Online]. Available at: https://reverb.com/item/2532627-1935-epiphone-electar-electrophone-lap-steel-guitar-w-ohsc [Accessed 22 March 2024]

[34] "Gibson L-5 1922-1933", reverb.com/en-gr/. [Online]. Available at: https://reverb.com/p/gibson-l-5-archtop-1922-1933 [Accessed 23 March 2024]

[35] "Gibson (guitar company)", en.wikipedia.org/wiki/Main_Page. [Online]. Available at: https://en.wikipedia.org/wiki/Gibson_Brands [Accessed 22 March 2024]

[36] "Gibson History", chasingguitars.com. [Online]. Available at: https://chasingguitars.com/gibson-history/ [Accessed 22 March 2024]

[37] "A brief history of Gibson", guitar.com. [Online]. Available at: https://guitar.com/features/brief-history-of-gibson/ [Accessed 22 March 2024]

[38] "Fender (company)", en.wikipedia.org/wiki/Main_Page. [Online]. Available at: https://en.wikipedia.org/wiki/Fender_(company) [Accessed 23 March 2024]

[39] "About Fender", fender.com/en-GR/start. [Online]. Available at: https://www.fender.com/pages/about [Accessed 23 March 2024]

[40] "A brief history of Fender guitars", guitar.com. [Online]. Available at: https://guitar.com/features/brief-history-of-fender-guitars/ [Accessed 23 March 2024]

[41] "Fender Stratocaster 1954", reverb.com/en-gr/. [Online]. Available at: https://reverb.com/p/fender-stratocaster-1954 [Accessed 23 March 2024]

[42] "Fender Telecaster since 1950", fineartamerica.com. [Online]. Available at: https://fineartamerica.com/featured/fender-telecaster-since-1950-wb-johnston.html [Accessed 23 March 2024]

[43] "Uber-Rare Fender 1946 Esquire Prototype collectible Snake Head 50th Anniversary Custom Shop #26/50", reverb.com/en-gr/. [Online]. Available at: https://reverb.com/item/2192489-uber-rare-fender-1946-esquire-prototype-collectible-snake-head-50th-anniversary-custom-shop-26-50 [Accessed 23 March 2024]

[44] "Fender Jaguar Solid Body Electric Guitar (1962)", retrofret.com. [Online]. Available at: https://www.retrofret.com/product.asp?ProductID=11799&name=Fender-Jaguar-Solid-Body-Electric-Guitar-1962 [Accessed 23 March 2024]

[45] "Fender Jazzmaster Solid Body Electric Guitar (1959)", retrofret.com. [Online]. Available at: https://www.retrofret.com/product.asp?ProductID=9299 [Accessed 23 March 2024]

[46] "Gibson Robot Guitar", en.wikipedia.org/wiki/Main_Page. [Online]. Available at: https://en.wikipedia.org/wiki/Gibson_Robot_Guitar [Accessed 24 March 2024]

[47] "Gibson's Self-Tuning Guitar", premierguitar.com. [Online]. Available at: https://www.premierguitar.com/gear/gibsons-self-tuning-guitar [Accessed 24 March 2024]

[48] Nate Pallesen, "G Force Tuning System: Gibson?s Automatic Tuning System", sixstringacoustic.com. [Online]. Available at: https://sixstringacoustic.com/g-force-tuning-system-gibsons-automatic-tuning-system [Accessed 24 March 2024]

[49] "Piezo acoustic tune-o-matic bridges (piezo crystal saddles)", alegree.co.uk. [Online]. Available at: https://www.alegree.co.uk/products/piezo-acoustic-tone-tune-o-matic-bridges-crystal-saddles [Accessed 24 March 2024]

[50] Mike Hanlon, "The Transperformance self-tuning guitar", newatlas.com. [Online]. Available at: https://newatlas.com/the-transperformance-self-tuning-guitar/4951/ [Accessed 24 March 2024]

[51] "Gibson G-Force Tuning System", artisanluthiers.com/blog/. [Online]. Available at: https://www.artisanluthiers.com/blog/gibson-g-force-tuning-system/ [Accessed 24 March 2024]

[52] "How it works? – Guitar Tuner AI full automatic for … - Tronical", tronicaltune.net. [Online]. Available at: https://www.tronicaltune.net/how-it-works/?v=f214a7d42e0d [Accessed 24 March 2024]

[53] Σπύρος Δελτα, «Τα μέρη της κιθάρας», musicheaven.gr/html/. [Online]. Available at:https://www.musicheaven.gr/html/modules.php?name=News&file=article&id=4424 [Accessed 26 March 2024]

[54] "Seven-string guitar", en.wikipedia.org/wiki/Main_Page. [Online]. Available at: https://en.wikipedia.org/wiki/Seven-string_guitar [Accessed 26 March 2024]

[55] Jef, "Gibson Les Paul Standard 7 String: Magnificent Seven?", gearnews.com. [Online]. Available at: https://www.gearnews.com/gibson-les-paul-standard-7-string-magnificent-seven/ [Accessed 26 March 2024]

[56] "RG5328 | RG | ELECTRIC GUITARS | PRODUCTS", ibanez.com/na/. [Online]. Available at: https://www.ibanez.com/na/products/detail/rg5328_00_02.html [Accessed 26 March 2024]

[57] "Ten-string guitar", en.wikipedia.org/wiki/Main_Page. [Online]. Available at: https://en.wikipedia.org/wiki/Ten-string_guitar [Accessed 26 March 2024]

[58] "New Halo 10-String Guitar: Limited Edition OCTAVIA-10 (Fanned Fret)", haloguitars.com/store/. [Online]. Available at: https://www.haloguitars.com/store/new-halo-10-string-guitar-limited-edition-octavia-10-fanned-fret [Accessed 26 March 2024]

[59] "J-45 Standard 12-String, Vintage Sunburst", gibson.com/en-US. [Online]. Available at: https://www.gibson.com/en-US/Acoustic-Guitar/ACCDMY224/Vintage-Sunburst [Accessed 26 March 2024]

[60] "Gibson Custom Shop EDS-1275 Double Neck - Cherry", long-mcquade.com. [Online]. Available at: https://www.long-mcquade.com/181760/Guitars/Electric-Guitars/Gibson/EDS-1275-Double-Neck---Cherry.htm [Accessed 26 March 2024]

[61] "Guitar tunings", en.wikipedia.org/wiki/Main_Page. [Online]. Available at: https://en.wikipedia.org/wiki/Guitar_tunings [Accessed 30 March 2024]

[62] "List of guitar tunings", en.wikipedia.org/wiki/Main_Page. [Online]. Available at: https://en.wikipedia.org/wiki/List_of_guitar_tunings [Accessed 30 March 2024]

[63] Stringjoy, "Alternate Guitar Tunings: The Ultimate Guide", stringjoy.com. [Online]. Available at: https://stringjoy.com/alternate-tunings/ [Accessed 30 March 2024]

[64] Peter Mitchell , "What Are the Guitar String Frequencies? Explanation and Sound Samples", soundadventurer.com. [Online]. Available at: https://soundadventurer.com/what-are-the-guitar-string-frequencies/ [Accessed 30 March 2024]

[65] Aaron Matthies, "Fanned Frets Explained: Everything You Need to Know", guitargearfinder.com. [Online]. Available at: https://guitargearfinder.com/faq/fanned-frets-explained/ [Accessed 30 March 2024]

[66] "Multi-scale fingerboard", en.wikipedia.org/wiki/Main_Page. [Online]. Available at: https://en.wikipedia.org/wiki/Multi-scale_fingerboard [Accessed 30 March 2024]

[67] "Popular Metal Drop Tunings: and how to tune for them", bedroomguitarist.com. [Online]. Available at: https://www.bedroomguitarist.com/articles/popular-metal-drop-tunings [Accessed 30 March 2024]

[68] "Double Drop D Tuning – DADGBD", theacousticguitarist.com. [Online]. Available at: https://theacousticguitarist.com/double-drop-d-tuning-dadgbd/ [Accessed 30 March 2024]

[69] "The Ultimate Guide to Half Step Down Tuning: Why and How to Use It", breakthroughguitar.com. [Online]. Available at: https://breakthroughguitar.com/the-ultimate-guide-to-half-step-down-tuning-why-and-how-to-use-it/ [Accessed 30 March 2024]

[70] "Understanding Note Frequency Charts (And Why You Should Be Using One)", producelikeapro.com/blog/. [Online]. Available at: https://producelikeapro.com/blog/note-frequency-chart/ [Accessed 30 March 2024]

[71] "Fully Automated Guitar Tuner", by Benjamin Wang, Brandon Ramos & Cooper Ge. (2020, April 17) [Online]. Available at: https://courses.engr.illinois.edu/ece445/getfile.asp?id=18170

[72] "Automatic Acoustic Guitar Tuner", by Alfredo Bocanegra. (2005, Jun. 8). [Online]. Available at: https://dspace.mit.edu/bitstream/handle/1721.1/32878/62588821-MIT.pdf;sequence=2

[73] "KORG GA-30 CHROMATIC TUNER", musiclandlive.com. [Online]. Available at: https://musiclandlive.com/products/korg-ga-30-chromatic-tuner [Accessed 16 April 2024]

[74] Mary Lourde R. & Anjali Kuppayil Saji, "A Digital Guitar Tuner", *IJCSIS*, vol. 6, no. 2, pp. 82-88, Dec. 2009 [Online]. Available at: https://www.researchgate.net/publication/45888076_A_Digital_Guitar_Tuner

[75] "Robotic Electric Guitar Tuner", by Martin Gylling & Ruben Svensson. (2018. April 24) [Online]. Available at: https://www.diva-portal.org/smash/get/diva2:1200615/FULLTEXT01.pdf

[76] Gomathy S, Prabha Maheswari M, Aparna P, Prabhakaran S & Yaswanth Kumar N, "A Review on Guitar Tuners", *IJRAR*, vol. 9, no. 2, pp. 4-10, April 2022. [Online]. Available at: https://ijrar.org/papers/IJRAR1COP002.pdf

[77] "Automatic Guitar Tuner", by Trenton Ahrens, Alex Capo & Ernesto Wong. (2014, April 12). [Online]. Available at: https://www.ece.ucf.edu/seniordesign/fa2014sp2015/g01/Final_SD1_Proj.pdf

[78] "Final Project ECE 470 Microcontroller Based Guitar Tuner", by Benjamin Bachman, Clifford Villero & Karlou Putrus, [Online]. Available at: https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.710.11&rep=rep1&type=pdf

[79] "Design of an Electric Guitar Tuner", by Bhimsen Budhathoki. (2018, Sep. 21). [Online]. Available at: https://www.theseus.fi/bitstream/handle/10024/153013/Bhimsen_Budhathoki.pdf;jsessionid=FE1EE4018461CE1927615AA0185C0C22?sequence=1

[80] "DESIGN AND DEVELOPMENT OF A LOW COST AUTOMATIC STRINGED INSTRUMENT TUNER", by J. Sevilla Salcedo, D. Mart´ınez Gila, I. Ruano Ruano, A. S´anchez Garc´ıa, E. Est´evez Est´evez, J. G´omez Ortega y J. & G´amez Garc´ıa. (2019. Sep. 4-6) [Online]. Available at: https://ruc.udc.es/dspace/bitstream/handle/2183/23780/2019_Sevilla-Salcedo_Design-developmet-low-cost-instrument-tuner.pdf;jsessionid=B4AE4F89A05C6DE021C7E95CD89E8C4C?sequence=3

[81] "Automatic guitar tuner", by Antonio Hinojosa Cabrera. (2020. Feb. 5). [Online]. Available at: https://upcommons.upc.edu/bitstream/handle/2117/178047/pfg-antonio-hinojosa.pdf

[82] Παναγιώτης Παπάζογλου, Σπύρος-Πολυχρόνης Λιώνης, *Ανάπτυξη εφαρμογών με το Arduino*, Θεσσαλονίκη: Τζιόλα, 2021, ISBN:978-960-418-937-3, Σελ. 3-143.

[83] "Arduino Uno Rev3", store.arduino.cc. [Online] Available at: https://store.arduino.cc/products/arduino-uno-rev3 [Accessed 22 April 2022]

[84] "Arduino Mega 2560 Rev3", store.arduino.cc. [Online] Available at:. https://store.arduino.cc/products/arduino-mega-2560-rev3 [Accessed 22 April 2022]

[85] "LilyPad Arduino Main Board", docs.arduino.cc. [Online] Available at: https://docs.arduino.cc/retired/boards/lilypad-arduino-main-board/ [Accessed 22 April 2022]

[86] "Arduino Nano", store.arduino.cc. [Online] Available at: https://store.arduino.cc/products/arduino-nano [Accessed 22 April 2022]

[87] "Arduino Nano 33 BLE Sense" , store-usa.arduino.cc. [Online] Available at: https://store-usa.arduino.cc/products/arduino-nano-33-ble-sense [Accessed 22 April 2022]

[88] "Arduino Nano 33 IoT", store.arduino.cc. [Online] Available at: https://store.arduino.cc/products/arduino-nano-33-iot [Accessed 22 April 2022]

[89] "Arduino Due", store.arduino.cc. [Online] Available at: https://store.arduino.cc/products/arduino-due [Accessed 22 April 2022]

[90] "ARDUINO UNO WiFi REV2" , store.arduino.cc. [Online] Available at: https://store.arduino.cc/products/arduino-uno-wifi-rev2 [Accessed 22 April 2022]

[91] "Software | Arduino", arduino.cc. [Online] Available at: https://www.arduino.cc/en/software [Accessed 22 April 2022]

[92] "Cloud Editor", arduino.cc. [Online] Available at: https://app.arduino.cc/sketches/2e59fca7-4c44-4c07-8c0e-b29f5f87d2c0 [Accessed 22 April 2022]

[93] "Install the Arduino Cloud Agent", support.arduino.cc/hc/en-us. [Online] Available at: https://support.arduino.cc/hc/en-us/articles/360014869820-Install-the-Arduino-Create-Agent [Accessed 22 April 2022]

[94] "What Are the Different Types of 3D Printing?", futurelearn.com. [Online] Available at: https://www.futurelearn.com/info/courses/getting-started-with-digital-manufacturing/0/steps/184102 [Accessed 25 April 2022]

[95] "Stereolithography", custompartnet.com. [Online] Available at: https://www.custompartnet.com/wu/stereolithography [Accessed 25 April 2022]

[96] "Stereolithography", en.wikipedia.org/wiki/Main_Page. [Online] Available at: https://en.wikipedia.org/wiki/Stereolithography [Accessed 25 April 2022]

[97] "Selective laser sintering", en.wikipedia.org/wiki/Main_Page. [Online] Available at: https://en.wikipedia.org/wiki/Selective_laser_sintering [Accessed 25 April 2022]

[98] "What is FDM (fused deposition modeling) 3D printing?", hubs.com. [Online] Available at: https://www.hubs.com/knowledge-base/what-is-fdm-3d-printing/ [Accessed 27 April 2022]

[99] "Πολυγαλακτικό οξύ", el.wikipedia.org/wiki/Πύλη:Κύρια. [Online] Available at: https://el.wikipedia.org/wiki/%CE%A0%CE%BF%CE%BB%CF%85%CE%B3%CE%B1%CE%BB%CE%B1%CE%BA%CF%84%CE%B9%CE%BA%CF%8C_%CE%BF%CE%BE%CF%8D [Accessed 27 April 2022]

[100] "Acrylonitrile butadiene styrene", en.wikipedia.org/wiki/Main_Page. [Online] Available at: https://en.wikipedia.org/wiki/Acrylonitrile_butadiene_styrene [Accessed 27 April 2022]

[101] "3D Printing Industry Awards – Enterprise 3D printer of the year (FFF/FDM)", 3dprintingindustry.com. [Online] Available at: https://3dprintingindustry.com/news/3d-printing-industry-awards-enterprise-3d-printer-year-ffffdm-111574/ [Accessed 28 April 2022]

[102] "Ender-3 S1 ED Printer – Creality 3D", creality.com. [Online] Available at: https://www.creality.com/products/creality-ender-3-s1-3d-printer [Accessed 28 April 2022]

[103] "What is Digital Light Processing (DLP)?", markforged.com. [Online] Available at: https://markforged.com/resources/learn/3d-printing-basics/3d-printing-processes/what-is-digital-light-processing-dlp [Accessed 28 April 2022]

[104] "Digital Light Processing (DLP) 3D Printer Market 2022 - 2029: Top Players to Deliver Prominent Growth and Striking Opportunities For Newcomers: XYZprinting, Formlabs", openpr.com. [Online] Available at: https://www.openpr.com/news/2855514/digital-light-processing-dlp-3d-printer-market-2022-2029-top [Accessed 28 April 2022]

[105] "What is MJF (HP's Multi Jet Fusion) 3D printing?", hubs.com. [Online] Available at: https://www.hubs.com/knowledge-base/what-is-multi-jet-fusion/#how-does-multi-jet-fusion-work [Accessed 28 April 2022]

[106] "Schematic of Multi Jet Fusion (MJF) process", researchgate.net. [Online] Available at: https://www.researchgate.net/figure/Schematic-of-Multi-Jet-Fusion-MJF-process-a-HP-3D-4200-printer-showing-fusing_fig2_331599125 [Accessed 30 April 2022]

[107] "What is the Polyjet process in 3D Printing? Pros and Cons", jiga.io. [Online] Available at: https://jiga.io/processes/polyjet/ [Accessed 30 April 2022]

[108] "SLA vs. PolyJet: What You Need to Know", cadimensions.com. [Online] Available at: https://resources.cadimensions.com/cadimensions-resources/sla-vs-polyjet-need-know [Accessed 30 April 2022]

[109] "Direct Metal Laser Sintering(DMLS)", 3m3drobotics.com. [Online] Available at: https://www.3m3drobotics.com/direct-metal-laser-sinteringdmls/ [Accessed 30 April 2022]

[110] "What is Direct Metal Laser Sintering (DMLS)?", markforged.com. [Online] Available at: https://markforged.com/resources/learn/3d-printing-basics/3d-printing-processes/what-is-direct-metal-laser-sintering-dmls [Accessed 3 May 2022]

[111] "What is Electron Beam Melting (EBM)?", markforged.com. [Online] Available at: https://markforged.com/resources/learn/3d-printing-basics/3d-printing-processes/what-is-electron-beam-melting-ebm [Accessed 3 May 2022]

[112] "Electron Beam Melting - an overview", sciencedirect.com. [Online] Available at: https://www.sciencedirect.com/topics/chemistry/electron-beam-melting [Accessed 3 May 2022]

[113] "Fusion Quick Start Guide", autodesk.com. [Online] Available at: https://www.autodesk.com/learn/ondemand/curated/fusion-360-quick-start-guide [Accessed 3 May 2022]

[114] "How to start your first print in Ultimaker Cura", support.makerbot.com/s/. [Online] Available at: https://support.makerbot.com/s/article/1872100983001 [Accessed 3 May 2022]

[115] "Specifications | CM-300 - CONTACT MICROPHONE", korg.com/us/. [Online] Available at: https://www.korg.com/us/products/tuners/cm_300/specifications.php [Accessed 5 May 2022]

[116] "CM-300 – CONTACT MICROPHONE | KORG (USA)", korg.com/us/. [Online] Available at: https://www.korg.com/us/products/tuners/cm_300/index.php [Accessed 5 May 2022]

[117] "DC Motors", grobotronics.com. [Online] Available at: https://grobotronics.com/mechanics-and-wheels/servo/dc-motors/?sl=en&features_hash= [Accessed 5 May 2022]

[118] "DC Motors - page 4", grobotronics.com. [Online] Available at: https://grobotronics.com/mechanics-and-wheels/servo/dc-motors/page-4/?sl=en&features_hash= [Accessed 5 May 2022]

[119] "Servo", grobotronics.com. [Online] Available at: https://grobotronics.com/mechanics-and-wheels/servo/servo-el/page-2/ [Accessed 5 May 2022]

[120] "Stepper Motors", grobotronics.com. [Online] Available at: https://grobotronics.com/mechanics-and-wheels/servo/stepper-motors/ [Accessed 5 May 2022]

[121] "ULN2003 Stepper Motor Driver Board", devobox.com/el/. [Online] Available at: https://www.devobox.com/el/motor-control/221-uln2003-stepper-motor-driver-board.html?search_query=step+motor+shield&results=130 [Accessed 6 May 2022]

[122] "Stepper Motor 28BYJ-48, 5VDC", devobox.com/el/. [Online] Available at: https://www.devobox.com/el/motors/109-step-motor-28byj-48-5vdc.html [Accessed 6 May 2022]

[123] "LCD Display 16x2 Blue Backlight (for Arduino)", devobox.com/el/. [Online] Available at: https://www.devobox.com/el/lcd/170-lcd-display-16x2-blue-blacklight-for-arduino.html [Accessed 6 May 2022]

[124] "LCD 1602 & Keypad Shield for Arduino", devobox.com/el/. [Online] Available at: https://www.devobox.com/el/lcd/630-lcd-1602-keypad-shield-for-arduino.html?search_query=LCD&results=38 [Accessed 6 May 2022]

[125] "Texas Instruments TL082CP Bi-Fet Dual Operational Amplifier", rapidonline.com. [Online] Available at: https://www.rapidonline.com/texas-instruments-tl082cp-bi-fet-dual-operational-amplifier-82-0064 [Accessed 6 May 2022]

[126] Texas Instruments, "TL082 Wide Bandwidth Dual JFET Input Operational Amplifier datasheet", Dallas, Texas, USA, SNOSBW5C, (1998). [Online] Available at: https://www.ti.com/lit/ds/symlink/tl082-n.pdf [Accessed 6 May 2022]

[127] "Arduino Audio Input: 8 Steps (with Pictures) - Instructables", instructables.com. [Online] Available at: https://www.instructables.com/Arduino-Audio-Input/ [Accessed 6 May 2022]

[128] "Arduino Frequency Detection : 4 Steps (with Pictures) - Instructables", instructables.com. [Online] Available at: https://www.instructables.com/Arduino-Frequency-Detection/ [Accessed 6 May 2022]

[129] Atmel Corporation, "ATmega640/1280/1281/2560/2561 datasheet", San Jose, California, USA, 2549Q–AVR–02/2014. [Online] Available at: https://ww1.microchip.com/downloads/en/devicedoc/atmel-2549-8-bit-avr-microcontroller-atmega640-1280-1281-2560-2561_datasheet.pdf [Accessed 7 May 2022]

[130] "Stepper Motor 28BYJ-48, 5VDC", devobox.com/el/. [Online] Available at: https://www.devobox.com/el/motors/109-step-motor-28byj-48-5vdc.html?search_query=step+motor+shield&results=130 [Accessed 7 May 2022]

[131] "288YJ-48 - 5V Stepper Motor Pinout Wiring, Specifications, Uses Guide", components101.com. [Online] Available at: https://components101.com/motors/28byj-48-stepper-motor [Accessed 7 May 2022]

[132] "28BYJ-48 Stepper Motor" , components101.com. [Online] Available at: https://components101.com/sites/default/files/component_datasheet/28byj48-step-motor-datasheet.pdf [Accessed 7 May 2022]

[133] "Control 28BYJ-48 Stepper Motor with ULN2003 Driver & Arduino", lastminuteengineers.com. [Online] Available at: https://lastminuteengineers.com/28byj48-stepper-motor-arduino-tutorial/ [Accessed 10 May 2022]

[134] "Arduino: Stepper Motor Example Sketch Fixed", instructables.com. [Online] Available at: https://www.instructables.com/Arduino-Stepper-Motor-Example-Sketch-Fixed/ [Accessed 10 May 2022]

[135] "10 Arduino Stepper Motor Control", 36projectsblog.wordpress.com. [Online] Available at: https://36projectsblog.wordpress.com/10-arduino-stepper-motor-control/ [Accessed 10 May 2022]

[136] "The ULN2003 Driver Board", lastminuteengineers.com. [Online] Available at: https://www.hadex.cz/spec/m513.pdf [Accessed 10 May 2022]

[137] "4 Phase ULN2003 Stepper Motor Driver PCB", electronicoscaldas.com/es/. [Online] Available at: https://www.electronicoscaldas.com/datasheet/ULN2003A-PCB.pdf [Accessed 10 May 2022]

[138] "Darlington Transistor and the Sziklai Darlington Pair", electronics-tutorials.ws. [Online] Available at: https://www.electronics-tutorials.ws/transistor/darlington-transistor.html [Accessed 10 May 2022]

[139] "Darlington transistor", en.wikipedia.org/wiki/Main_Page. [Online] Available at: https://en.wikipedia.org/wiki/Darlington_transistor [Accessed 10 May 2022]

[140] "LCD Keypad Shield V1.0_ Schematic.pdf",openimpulse.com/blog/. [Online] Available at: https://www.openimpulse.com/blog/wp-content/uploads/wpsc/downloadables/LCD-Keypad-Shield-V1.0-Schematic.pdf [Accessed 12 May 2022]

[141] "Μπαταρία Varta Recharge Accu Power 9V Επαναφορτιζόμενη 200mAh", batteries.gr/gr/. [Online] Available at: https://www.batteries.gr/gr/audio-and-video/rechargeable-batteries/mpataria-varta-recharge-accu-power-9v-epanafortizomeni-200mah.html [Accessed 12 May 2022]

[142] "Samsung INR18650-35E 3500mAh 3.6V - 3.7V", devobox.com/el/. [Online] Available at: https://www.devobox.com/el/18650/580-samsung-inr-18650-35e-3500mah-36v-37v.html [Accessed 12 May 2022]

[143] "Toggle Switch ON-OFF-ON 3PDT (3A/250V)", grobotronics.com. [Online] Available at: https://grobotronics.com/toggle-switch-on-off-on-3pdt-3a-250v.html [Accessed 12 May 2022]

[144] "Ender-3 Pro 3D Printer", creality.com. [Online] Available at: https://www.creality.com/products/ender-3-pro-3d-printer [Accessed 12 May 2022]

[145] "ΠΑΧΥΜΕΤΡΟ MITUTOYO", intool.gr. [Online] Available at: https://www.intool.gr/index.php?id_product=247&controller=product [Accessed 12 May 2022]

Παράρτημα Α

Σε αυτό το Παράρτημα, παρουσιάζεται ένας πίνακας (Πίνακας Α) με τα υλικά που αγοράστηκαν για την υλοποίηση της συγκεκριμένης διπλωματικής εργασίας. Πιο συγκεκριμένα, αναφέρεται το κόστος του καθενός, το συνολικό κόστος και η ποσότητα που απαιτείται για την ανάπτυξη του συστήματος αυτομάτου ελέγχου για τον συντονισμό των χορδών της κιθάρας.

Πίνακας Α: Αναλυτικό κοστολόγιο των υλικών για την ανάπτυξη του συστήματος αυτομάτου ελέγχου για τον συντονισμό των χορδών [47].

Components	Quantity	Cost (€)
Contact Microphone Korg	1	11,00 €
LCD 16x2 with Keypad Shield	1	6,70 €
Microcontroller Arduino Mega	1	52,00 €
Stepper Motor	6	6 x 2€ = 12€
Stepper Motor Driver	6	6 x 1€ = 6€
TL082 Amplifier	1	2,70 €
Capacitors, Resistors, Terminal boxes	1	5,00 €
Rechargeable Batteries Varta	3	3 x 9,24€ = 27,72€
Rechargeable Batteries Samsung 18650 3100mAh	2	2 x 7,20€ = 14,40€
Creality3D PLA Black Filament 1,75mm 1kg	1	15,86 €
Total Cost (€)	-	153,38 €

Παράρτημα Β

Σε αυτό το παράρτημα, βρίσκεται ολόκληρος ο κώδικας προγραμματισμού του συστήματος αυτομάτου ελέγχου για το συντονισμό των χορδών της κιθάρας.

```
#include <LiquidCrystal.h> //Liquid crystal library

#define eStepper_Pin1 42 //define 1st pin for e string step motor
#define eStepper_Pin2 43 //define 2nd pin for e string step motor
#define eStepper_Pin3 44 //define 3rd pin for e string step motor
#define eStepper_Pin4 45 //define 4th pin for e string step motor
#define BStepper_Pin1 46 //define 1st pin for B string step motor
```

```
#define BStepper_Pin2 47 //define 2nd pin for B string step motor
#define BStepper_Pin3 48 //define 3rd pin for B string step motor
#define BStepper_Pin4 49 //define 4th pin for B string step motor
#define GStepper_Pin1 50 //define 1st pin for G string step motor
#define GStepper_Pin2 51 //define 2nd pin for G string step motor
#define GStepper_Pin3 52 //define 3rd pin for G string step motor
#define GStepper_Pin4 53 //define 4th pin for G string step motor
#define DStepper_Pin1 22 //define 1st pin for D string step motor
#define DStepper_Pin2 23 //define 2nd pin for D string step motor
#define DStepper_Pin3 24 //define 3rd pin for D string step motor
#define DStepper_Pin4 25 //define 4th pin for D string step motor
#define AStepper_Pin1 26 //define 1st pin for A string step motor
#define AStepper_Pin2 27 //define 2nd pin for A string step motor
#define AStepper_Pin3 28 //define 3rd pin for A string step motor
#define AStepper_Pin4 29 //define 4th pin for A string step motor
#define EStepper_Pin1 30 //define 1st pin for E string step motor
#define EStepper_Pin2 31 //define 2nd pin for E string step motor
#define EStepper_Pin3 32 //define 3rd pin for E string step motor
#define EStepper_Pin4 33 //define 4th pin for E string step motor

boolean clipping = 0, string_1 = 0, string_2 = 0, string_3 = 0, string_4 = 0, string_5 = 0,
string_6 = 0; //Name and determine boolean variables
char note = ' '; //Name and determine char variable
byte Manual_Finish = 0, Manual_Select = 0, Finish_Control = 0, Turn_Right = 0,
Turn_Left = 0; //Name and determine byte variable
byte newData = 0; //Name and determine byte variable
byte prevData = 0; //Name and determine byte variable
unsigned int time = 0; //Name and determine unsigned integer variable
int timer[10]; //Name and determine integer variable
int slope[10]; //Name and determine integer variable
unsigned int totalTimer; //Name and determine unsigned integer variable
unsigned int period; //Name and determine unsigned integer variable
byte index = 0; //Name and determine byte variable
float frequency; //Name and determine float variable
int maxSlope = 0; //Name and determine integer variable
int newSlope; //Name integer variable
```

```
byte noMatch = 0; //Name and determine byte variable
byte slopeTol = 3; //Name and determine byte variable
int timerTol = 10; //Name and determine integer variable
unsigned int ampTimer = 0; //Name and determine unsigned integer variable
byte maxAmp = 0; //Name and determine byte variable
byte checkMaxAmp; //Name byte variable
byte ampThreshold = 30; //Name and determine byte variable

LiquidCrystal lcd(8, 9, 4, 5, 6, 7); // Digital pin that Liquid Crystal screen will use

int x, k = 0, menu = 1, manual_menu = 1, i; //Name and determine integer variables

void setup() { //Set up function
  Serial.begin(9600); //Begin Serial communicate at 9600 Baud rate
  lcd.begin(16,2); //Determine Liquid Crystal dimensions
  pinMode(eStepper_Pin1, OUTPUT); //determine 1st pin for e string step motor as an
output
  pinMode(eStepper_Pin2, OUTPUT); //determine 2nd pin for e string step motor as an
output
  pinMode(eStepper_Pin3, OUTPUT); //determine 3rd pin for e string step motor as an
output
  pinMode(eStepper_Pin4, OUTPUT); //determine 4th pin for e string step motor as an
output
  pinMode(BStepper_Pin1, OUTPUT); //determine 1st pin for B string step motor as an
output
  pinMode(BStepper_Pin2, OUTPUT); //determine 2nd pin for B string step motor as an
output
  pinMode(BStepper_Pin3, OUTPUT); //determine 3rd pin for B string step motor as an
output
  pinMode(BStepper_Pin4, OUTPUT); //determine 4th pin for B string step motor as an
output
  pinMode(GStepper_Pin1, OUTPUT); //determine 1st pin for G string step motor as an
output
  pinMode(GStepper_Pin2, OUTPUT); //determine 2nd pin for G string step motor as an
output
```

```
  pinMode(GStepper_Pin3, OUTPUT); //determine 3rd pin for G string step motor as an
output
  pinMode(GStepper_Pin4, OUTPUT); //determine 4th pin for G string step motor as an
output
  pinMode(DStepper_Pin1, OUTPUT); //determine 1st pin for D string step motor as an
output
  pinMode(DStepper_Pin2, OUTPUT); //determine 2nd pin for D string step motor as an
output
  pinMode(DStepper_Pin3, OUTPUT); //determine 3rd pin for D string step motor as an
output
  pinMode(DStepper_Pin4, OUTPUT); //determine 4th pin for D string step motor as an
output
  pinMode(AStepper_Pin1, OUTPUT); //determine 1st pin for A string step motor as an
output
  pinMode(AStepper_Pin2, OUTPUT); //determine 2nd pin for A string step motor as an
output
  pinMode(AStepper_Pin3, OUTPUT); //determine 3rd pin for A string step motor as an
output
  pinMode(AStepper_Pin4, OUTPUT); //determine 4th pin for A string step motor as an
output
  pinMode(EStepper_Pin1, OUTPUT); //determine 1st pin for E string step motor as an
output
  pinMode(EStepper_Pin2, OUTPUT); //determine 2nd pin for E string step motor as an
output
  pinMode(EStepper_Pin3, OUTPUT); //determine 3rd pin for E string step motor as an
output
  pinMode(EStepper_Pin4, OUTPUT); //determine 4th pin for E string step motor as an
output

  lcd.setCursor(0,0); //Set cursor at position 0 column and 0 line
  lcd.print("AUTOMATIC GUITAR"); //Print message at LCD screen
  lcd.setCursor(5,1); //Set cursor at position 5 column and 1 line
  lcd.print("TUNER!"); //Print message at LCD
  delay(2500); //Time delay for 2.5 sec
  lcd.clear(); //Clear LCD screen
  lcd.setCursor(0,0); //Set cursor at position 0 column and 0 line
```

```
  lcd.print("Choose your Tune"); //Print message at LCD screen
  for (i=0;i<=5;i++) //for 6 times repeat
  {
    lcd.setCursor((i+5),1); //Set Cursor at position i+5 column and 1 line
    lcd.print("."); //Print a Dot at LCD screen
    delay(500); //Delay for 500 msec
  }
  Menu(); //Call Menu(); function
}//void setup ()

void loop() { //Loop function
  ButtonRead(); //Go at ButtonRead(); function
}//loop end

void Menu() { //Menu function
  switch (menu){ //Check menu number with switch
  case 0: //In case that menu = 0
  menu = 1; //Store number 1 at menu variable
  break; //Break the switch loop
  case 1: //In case that menu = 1
  lcd.clear(); //Clear LCD screen
  lcd.print(">Standard Tune"); //Print message at LCD screen
  lcd.setCursor(0,1); //Set cursor at 1st column and 2nd line
  lcd.print(" Half Step Down"); //Print message at LCD screen
  break; //Break the switch loop
  case 2: //In case that menu = 2
  lcd.clear(); //Clear LCD screen
  lcd.print(" Standard Tune"); //Print message at LCD screen
  lcd.setCursor(0,1); //Set cursor at 1st column and 2nd line
  lcd.print(">Half Step Down"); //Print message at LCD screen
  break; //Break switch loop
  case 3: //In case that menu = 3
  lcd.clear(); //Clear LCD screen
  lcd.print(">Drop D Tune"); //Print message at LCD screen
  lcd.setCursor(0,1); //Set cursor at 1st column and 2nd line
  lcd.print(" Manual Control"); //Print message at LCD screen
```

```
  break; //Break switch loop
  case 4: //In case that menu = 4
  lcd.clear(); //Clear LCD screen
  lcd.print(" Drop D Tune"); //Print message at LCD screen
  lcd.setCursor(0,1); //Set cursor at 1st column and 2nd line
  lcd.print(">Manual Control"); //Print message at LCD screen
  break; //Break switch loop
  case 5: //In case that menu = 5
  menu = 4; //Store number 4 at menu Variable
  break; //Break switch loop
  } //switch menu
}

void ButtonRead() //ButtonRead
{
  //x=analogRead(A0);
  ADCSRA = 0; //Clear ADCSRA register
  ADCSRB = 0; //Clear ADCSRB register
  ADMUX = 0; //Clear ADMUX register
  ADMUX |= (1<<REFS0)|(0<<REFS1); //Set bit REFS0 and clear bit REFS1 at ADMUX
register, set reference voltage
  ADCSRB |= (1<<ACME); //Set bit ACME at ADCSRB register and enable interrupts
when measurement complete
  ADCSRA |=(1<<ADEN); //Set bit ADEN at ADCSRA register and enable ADC
  ADCSRA |= (1<<ADSC); //Set bit ADSC at ADCSRA register and start ADC
measurements
  x = ADC; //Store value from ADC to x variable
  delay(200); //Delay 200 msec
  Serial.println(x); //Print x value at Serial port

  if (x<60 && x>0) //If RIGHT button has been pressed, then
  {
    lcd.clear(); //Clear LCD screen
    lcd.setCursor(1,0); //Set cursor at first column and first line
    lcd.print("USE UP OR DOWN"); //Print message at LCD screen
    lcd.setCursor(5,1); //Set cursor at first column and first line
```

```
  lcd.print("KEYS!!"); //Print message at LCD screen
  delay(2500); //Delay 2.5sec
  Menu(); //Call Menu function
  delay(200); //Delay 200 msec
 }
 else if(x<200) //If UP button has been pressed
 {
  menu--; //Decrease menu by one point
  Menu(); //Call function Menu
  delay(200); //Delay 200 msec
 }
 else if(x<400) //If DOWN button has been pressed
 {
  menu++; //Increase menu by one point
  Menu(); //Call Menu function
  delay(200); //Delay 200 msec
 }
 else if(x<600 && x>402) //If LEFT button has been pressed
 {
  lcd.clear(); //Clear LCD screen
  lcd.setCursor(1,0); //Set cursor at first column and first line
  lcd.print("USE UP OR DOWN"); //Print message at LCD screen
  lcd.setCursor(5,1); //Set cursor at first column and first line
  lcd.print("KEYS!!"); //Print message at LCD screen
  delay(2500); //Delay 2.5sec
  Menu(); //Call Menu function
  delay(200); //Delay 200 msec
 }
 else if(x<800 && x!=689) //If SELECT button has been
 {
  x = 1023 ; //Store value at x variable
  if (menu==1) //If menu value is 1, then
  {
   lcd.clear(); //Clear LCD screen
   lcd.setCursor(0,0); //Set cursor at first column and first line
   lcd.print("  Standard Tune "); //Print message at LCD screen
```

```
    lcd.setCursor(4,1); //Set cursor at fifth column and first line
    lcd.print("Selected"); //Print message at LCD screen
    delay(1000); //Delay 1 sec
    lcd.clear(); //Clear LCD screen
    lcd.setCursor(2,0); //Set Cursor at third column and first line
    lcd.print("Hit E string"); //Print message at LCD screen
    delay(1000); //Delay 1 sec
    lcd.clear(); //Clear LCD screen
    lcd.setCursor(1,0); //Set Cursor at third column and first line
    lcd.print("E -> 82.41 Hz"); //Print message at LCD screen
    lcd.setCursor(7,1); //Set Cursor at first column and second line
    lcd.print("Hz   Note"); //print message at LCD screen
    k = 21; //Store value at k variable
    string_6 = 0;//Reset the value at string_6 variable
    string_5 = 0;//Reset the value at string_5 variable
    string_4 = 0;//Reset the value at string_4 variable
    string_3 = 0;//Reset the value at string_3 variable
    string_2 = 0;//Reset the value at string_2 variable
    string_1 = 0;//Reset the value at string_1 variable
    while(string_6 != 1) //While string_6 flag isn't set repeat
    {
     if (k>20) //If k variable is bigger than 20, then
     {
      pinMode(39,OUTPUT); //Determine digital pin 39 as Output
      digitalWrite(39,LOW); //Set low voltage (GND) at pin 39
      delay(10); //Delay 10 msec
      pinMode(39,INPUT); //Determine digital pin 39 as Input
      k = 0; //Reset k variable
     } //if (k>20)
     FREQ_DETECTION(); //Call FREQ_DETECTION function (Change analog port for
the income signal)
     delay(500); //Delay 0.5 sec
     FREQ_DETECTION1(); //Call FREQ_DETECTION1 function (Calculate frequency
and stop interrupts by ADC)
     FREQ_DETECTION_STOP(); //Call FREQ_DETECTION_STOP function (Stop
frequency detection)
```

```
      Serial.println("E String"); //Print message at Serial Port
      Serial.println(frequency); //Print frequency value at Serial Port
      lcd.setCursor(0,1); //Set cursor at first column and second line
      lcd.print("       "); //Print message at lcd screen
      lcd.setCursor(0,1); //Set cursor at first column and second line
      lcd.print(frequency); //Print frequency value at LCD screen
      k++; //Increase variable k value by one
      if (frequency < 82.35 && frequency > 65 && frequency > 0) //If frequency value is
smaller than 82.35Hz and bigger than 65Hz and bigger than 0Hz, then
      {
       frequency = 0.00; //Reset frequency value
       for (i=0; i<=20; i++) //For 21 times repeat
       {
        Counterclockwise_EStep_Motor (); //Call function for E bass string step motor
       }
       delay(2); //Delay 2 msec
       EStep_Motor_Stop (); //Disable E bass string step motor
      } //if (frequency...
      else if (frequency > 82.47 && frequency < 103 && frequency > 0) //Else if
frequency value is bigger than 82.47Hz and smaller than 103Hz and bigger than 0Hz, then
      {
       frequency = 0.00; //Reset frequency value
       for(i=0; i<=20; i++) //For 21 times repeat
       {
        Clockwise_EStep_Motor (); //Call function for E bass string step motor
       }
       delay(2); //Delay 2 msec
       EStep_Motor_Stop (); //Disable E bass step motor
      } //else if (frequency...
      else if (frequency >=82.35 && frequency <=82.47) //Else if frequency value is bigger
or equal with 82.35Hz an smaller or equal with 82.47Hz, then E bass string has been tuned
      {
       string_6 = 1; //Set string_6 flag to finish tuning with E bass string
       k = 21; //Store value at k variable
       frequency = 0.00; //Reset frequency value
       lcd.clear(); //Clear LCD screen
```

```
        lcd.setCursor(2,0); //Set Cursor at third column and first line
        lcd.print("Hit A string"); //Print message at LCD screen
        delay(1000); //Delay 1 sec
        lcd.clear(); //Clear LCD screen
        lcd.setCursor(1,0); //Set Cursor at third column and first line
        lcd.print("A -> 110 Hz"); //Print message at LCD screen
        lcd.setCursor(7,1); //Set Cursor at first column and second line
        lcd.print("Hz   Note"); //print message at LCD screen
       } //else if (frequency...
       else //Else
       {
        frequency = 0.00; //Reset frequency value
       } //else
      } //while(string_6...
      while(string_5 != 1) //While string_5 flag isn't set repeat
      {
       if (k>20) //If k variable is bigger than 20, then
       {
        pinMode(39,OUTPUT); //Determine digital pin 39 as Output
        digitalWrite(39,LOW); //Set low voltage (GND) at pin 39
        delay(10); //Delay 10 msec
        pinMode(39,INPUT); //Determine digital pin 39 as Input
        k = 0;//Reset k variable
       }//if (k>20)
       FREQ_DETECTION(); //Call FREQ_DETECTION function (Change analog port for
the income signal)
       delay(500); //Delay 0.5 sec
       FREQ_DETECTION1(); //Call FREQ_DETECTION1 function (Calculate frequency
and stop interrupts by ADC)
       FREQ_DETECTION_STOP(); //Call FREQ_DETECTION_STOP function (Stop
frequency detection)
       Serial.println("A String"); //Print message at Serial Port
       Serial.println(frequency); //Print frequency value at Serial Port
       lcd.setCursor(0,1); //Set cursor at first column and second line
       lcd.print("       "); //Print message at lcd screen
       lcd.setCursor(0,1); //Set cursor at first column and second line
```

```
lcd.print(frequency); //Print frequency value at LCD screen
k++; //Increase variable k value by one
if (frequency < 109.80 && frequency > 92 && frequency > 0) //If frequency value is smaller than 109.80Hz and bigger than 92Hz and bigger than 0Hz, then
{
 frequency = 0.00; //Reset frequency value
 for (i=0; i<=20; i++) //For 21 times repeat
 {
  Counterclockwise_AStep_Motor (); //Call function for A string step motor
 }
 delay(2); //Delay 2 msec
 AStep_Motor_Stop (); //Disable A string step motor
} //if (frequency...
else if (frequency > 110.21 && frequency < 130 && frequency > 0)  //Else if frequency value is bigger than 110.21Hz and smaller than 130Hz and bigger than 0Hz, then
{
 frequency = 0.00; //Reset frequency value
 for(i=0; i<=20; i++) //For 21 times repeat
 {
  Clockwise_AStep_Motor (); //Call function for A string step motor
 }
 delay(2); //Delay 2 msec
 AStep_Motor_Stop (); //Disable A string step motor
} //else if (frequency...
else if (frequency >=109.80 && frequency <=110.21) //Else if frequency value is bigger or equal with 109.80Hz an smaller or equal with 110.21Hz, then A string has been tuned
{
 string_5 = 1;//Set string_5 flag to finish tuning with A string
 k = 21; //Store value at k variable
 frequency = 0.00; //Reset frequency value
 lcd.clear(); //Clear LCD screen
 lcd.setCursor(2,0); //Set Cursor at third column and first line
 lcd.print("Hit D string"); //Print message at LCD screen
 delay(1000);
```

```
        lcd.clear(); //Clear LCD screen
        lcd.setCursor(1,0); //Set Cursor at third column and first line
        lcd.print("D -> 146.83 Hz"); //Print message at LCD screen
        lcd.setCursor(7,1); //Set Cursor at first column and second line
        lcd.print("Hz   Note"); //print message at LCD screen
       } //else if (frequency...
       else //Else
       {
        frequency = 0.00; //Reset frequency value
       }
     } //while(string_5...
     while(string_4 != 1) //While string_4 flag isn't set repeat
     {
      if (k>20) //If k variable is bigger than 20, then
      {
       pinMode(39,OUTPUT); //Determine digital pin 39 as Output
       digitalWrite(39,LOW); //Set low voltage (GND) at pin 39
       delay(10); //Delay 10 msec
       pinMode(39,INPUT); //Determine digital pin 39 as Input
       k = 0; //Reset k variable
      } //if (k>20)
      FREQ_DETECTION(); //Call FREQ_DETECTION function (Change analog port for the income signal)
      delay(500); //Delay 0.5 sec
      FREQ_DETECTION1(); //Call FREQ_DETECTION1 function (Calculate frequency and stop interrupts by ADC)
      FREQ_DETECTION_STOP(); //Call FREQ_DETECTION_STOP function (Stop frequency detection)
      Serial.println("D String"); //Print message at Serial Port
      Serial.println(frequency); //Print frequency value at Serial Port
      lcd.setCursor(0,1); //Set cursor at first column and second line
      lcd.print("       "); //Print message at lcd screen
      lcd.setCursor(0,1); //Set cursor at first column and second line
      lcd.print(frequency); //Print frequency value at LCD screen
      k++; //Increase variable k value by one
```

```
if (frequency < 146.73 && frequency > 123 && frequency > 0) //If frequency value is smaller than 146.73Hz and bigger than 123Hz and bigger than 0Hz, then
{
 frequency = 0.00; //Reset frequency value
 for (i=0; i<=40; i++) //For 41 times repeat
 {
  Counterclockwise_DStep_Motor (); //Call function for D string step motor
 }
 delay(2); //Delay 2 msec
 DStep_Motor_Stop (); //Disable D string step motor
} //if (frequency...
else if (frequency > 146.93 && frequency < 174 && frequency > 0)  //Else if frequency value is bigger than 146.93Hz and smaller than 174Hz and bigger than 0Hz, then
{
 frequency = 0.00; //Reset frequency value
 for(i=0; i<=30; i++) //For 31 times repeat
 {
  Clockwise_DStep_Motor ();//Call function for D string step motor
 }
 delay(2); //Delay 2 msec
 DStep_Motor_Stop (); //Disable D string step motor
} //else if (frequency...
else if (frequency >=146.73 && frequency <=146.93) //Else if frequency value is bigger or equal with 146.73Hz an smaller or equal with 146.93Hz, then D string has been tuned
{
 string_4 = 1; //Set string_4 flag to finish tuning with D string
 k = 21; //Store value at k variable
 frequency = 0.00; //Reset frequency value
 lcd.clear(); //Clear LCD screen
 lcd.setCursor(2,0); //Set Cursor at third column and first line
 lcd.print("Hit G string"); //Print message at LCD screen
 delay(1000);
 lcd.clear(); //Clear LCD screen
 lcd.setCursor(1,0); //Set Cursor at third column and first line
```

```
        lcd.print("G -> 196 Hz"); //Print message at LCD screen
        lcd.setCursor(7,1); //Set Cursor at first column and second line
        lcd.print("Hz   Note"); //print message at LCD screen
       } //else if (frequency...
       else //Else
       {
        frequency = 0.00; //Reset frequency value
       }
     } //while(string_4...*/
     while(string_3 != 1) //While string_3 flag isn't set repeat
     {
      if (k>20) //If k variable is bigger than 20, then
      {
       pinMode(39,OUTPUT); //Determine digital pin 39 as Output
       digitalWrite(39,LOW); //Set low voltage (GND) at pin 39
       delay(10); //Delay 10 msec
       pinMode(39,INPUT); //Determine digital pin 39 as Input
       k = 0; //Reset k variable
      } //if (k>20)
      FREQ_DETECTION(); //Call FREQ_DETECTION function (Change analog port for the income signal)
      delay(500); //Delay 0.5 sec
      FREQ_DETECTION1(); //Call FREQ_DETECTION1 function (Calculate frequency and stop interrupts by ADC)
      FREQ_DETECTION_STOP(); //Call FREQ_DETECTION_STOP function (Stop frequency detection)
      Serial.println("G String"); //Print message at Serial Port
      Serial.println(frequency); //Print frequency value at Serial Port
      lcd.setCursor(0,1); //Set cursor at first column and second line
      lcd.print("       "); //Print message at lcd screen
      lcd.setCursor(0,1); //Set cursor at first column and second line
      lcd.print(frequency); //Print frequency value at LCD screen
      k++; //Increase variable k value by one
      if (frequency < 195.90 && frequency > 164 && frequency > 0) //If frequency value is smaller than 195.90Hz and bigger than 164Hz and bigger than 0Hz, then
      {
```

```
        frequency = 0.00; //Reset frequency value
        for (i=0; i<=15; i++) //For 16 times repeat
        {
          Clockwise_GStep_Motor ();//Call function for G string step motor
        }
        delay(2); //Delay 2 msec
        GStep_Motor_Stop (); //Disable G string step motor
      } //if (frequency...
      else if (frequency > 196.25 && frequency < 233 && frequency > 0)  //Else if
frequency value is bigger than 196.25Hz and smaller than 233Hz and bigger than 0Hz,
then
      {
        frequency = 0.00; //Reset frequency value
        for(i=0; i<=15; i++) //For 16 times repeat
        {
          Counterclockwise_GStep_Motor (); //Call function for G string step motor
        }
        delay(2); //Delay 2 msec
        GStep_Motor_Stop (); //Disable G string step motor
      } //else if (frequency...
      else if (frequency >=195.90 && frequency <=196.25) //Else if frequency value is
bigger or equal with 195.90Hz an smaller or equal with 196.25Hz, then G string has been
tuned
      {
        string_3 = 1; //Set string_3 flag to finish tuning with G string
        k = 21; //Store value at k variable
        frequency = 0.00; //Reset frequency value
        lcd.clear(); //Clear LCD screen
        lcd.setCursor(2,0); //Set Cursor at third column and first line
        lcd.print("Hit B string"); //Print message at LCD screen
        delay(1000);
        lcd.clear(); //Clear LCD screen
        lcd.setCursor(1,0); //Set Cursor at third column and first line
        lcd.print("B -> 246.94 Hz"); //Print message at LCD screen
        lcd.setCursor(7,1); //Set Cursor at first column and second line
        lcd.print("Hz   Note"); //print message at LCD screen
```

```
      } //else if (frequency...
      else //Else
      {
        frequency = 0.00; //Reset frequency value
      }
    } //while(string_3...*/
    while(string_2 != 1) //While string_2 flag isn't set repeat
    {
      if (k>20) //If k variable is bigger than 20, then
      {
        pinMode(39,OUTPUT); //Determine digital pin 39 as Output
        digitalWrite(39,LOW); //Set low voltage (GND) at pin 39
        delay(10); //Delay 10 msec
        pinMode(39,INPUT); //Determine digital pin 39 as Input
        k = 0; //Reset k variable
      } //if (k>20)
      FREQ_DETECTION(); //Call FREQ_DETECTION function (Change analog port for the income signal)
      delay(500); //Delay 0.5 sec
      FREQ_DETECTION1(); //Call FREQ_DETECTION1 function (Calculate frequency and stop interrupts by ADC)
      FREQ_DETECTION_STOP(); //Call FREQ_DETECTION_STOP function (Stop frequency detection)
      Serial.println("B String"); //Print message at Serial Port
      Serial.println(frequency); //Print frequency value at Serial Port
      lcd.setCursor(0,1); //Set cursor at first column and second line
      lcd.print("       "); //Print message at lcd screen
      lcd.setCursor(0,1); //Set cursor at first column and second line
      lcd.print(frequency); //Print frequency value at LCD screen
      k++; //Increase variable k value by one
      if (frequency < 246.54 && frequency > 207 && frequency > 0)  //If frequency value is smaller than 246.54Hz and bigger than 207Hz and bigger than 0Hz, then
      {
        frequency = 0.00; //Reset frequency value
        for (i=0; i<=20; i++) //For 21 times repeat
        {
```

```
        Clockwise_BStep_Motor (); //Call function for B string step motor
       }
       delay(2); //Delay 2 msec
       BStep_Motor_Stop (); //Disable B string step motor
      } //if (frequency...
      else if (frequency > 247.04 && frequency < 293 && frequency > 0)  //Else if
frequency value is bigger than 247.04Hz and smaller than 293Hz and bigger than 0Hz,
then
      {
       frequency = 0.00; //Reset frequency value
       for(i=0; i<=20; i++) //For 21 times repeat
       {
        Counterclockwise_BStep_Motor (); //Call function for B string step motor
       }
       delay(2); //Delay 2 msec
       BStep_Motor_Stop (); //Disable B string step motor
      } //else if (frequency...
      else if (frequency >=246.55 && frequency <=247.04) //Else if frequency value is
bigger or equal with 246.55Hz an smaller or equal with 247.04Hz, then B string has been
tuned
      {
       string_2 = 1; //Set string_2 flag to finish tuning with B string
       k = 21; //Store value at k variable
       frequency = 0.00; //Reset frequency value
       lcd.clear(); //Clear LCD screen
       lcd.setCursor(2,0); //Set Cursor at third column and first line
       lcd.print("Hit e string"); //Print message at LCD screen
       delay(1000);
       lcd.clear(); //Clear LCD screen
       lcd.setCursor(1,0); //Set Cursor at third column and first line
       lcd.print("e -> 329.63 Hz"); //Print message at LCD screen
       lcd.setCursor(7,1); //Set Cursor at first column and second line
       lcd.print("Hz   Note"); //print message at LCD screen
      } //else if (frequency...
      else //Else
      {
```

```
        frequency = 0.00; //Reset frequency value
       }
     } //while(string_2...*/
     while(string_1 != 1) //While string_1 flag isn't set repeat
     {
       if (k>20) //If k variable is bigger than 20, then
       {
         pinMode(39,OUTPUT); //Determine digital pin 39 as Output
         digitalWrite(39,LOW); //Set low voltage (GND) at pin 39
         delay(10); //Delay 10 msec
         pinMode(39,INPUT); //Determine digital pin 39 as Input
         k = 0; //Reset k variable
       } //if (k>20)
       FREQ_DETECTION(); //Call FREQ_DETECTION function (Change analog port for the income signal)
       delay(500); //Delay 0.5 sec
       FREQ_DETECTION1(); //Call FREQ_DETECTION1 function (Calculate frequency and stop interrupts by ADC)
       FREQ_DETECTION_STOP(); //Call FREQ_DETECTION_STOP function (Stop frequency detection)
       Serial.println("e String"); //Print message at Serial Port
       Serial.println(frequency); //Print frequency value at Serial Port
       lcd.setCursor(0,1); //Set cursor at first column and second line
       lcd.print("       "); //Print message at lcd screen
       lcd.setCursor(0,1); //Set cursor at first column and second line
       lcd.print(frequency); //Print frequency value at LCD screen
       k++; //Increase variable k value by one
       if (frequency < 328.73 && frequency > 277 && frequency > 0) //If frequency value is smaller than 328.73Hz and bigger than 277Hz and bigger than 0Hz, then
       {
         frequency = 0.00; //Reset frequency value
         for (i=0; i<=20; i++) //For 21 times repeat
         {
           Clockwise_eStep_Motor (); //Call function for e string step motor
         }
         delay(2); //Delay 2 msec
```

```
        eStep_Motor_Stop (); //Disable e string step motor
       } //if (frequency...
       else if (frequency > 331.58 && frequency < 392 && frequency > 0)  //Else if frequency value is bigger than 331.58Hz and smaller than 392Hz and bigger than 0Hz, then
       {
        frequency = 0.00; //Reset frequency value
        for(i=0; i<=20; i++) //For 21 times repeat
        {
         Counterclockwise_eStep_Motor (); //Call function for e string step motor
        }
        delay(2); //Delay 2 msec
        eStep_Motor_Stop (); //Disable e string step motor
       } //else if (frequency...
       else if (frequency >=328.73 && frequency <=331.58) //Else if frequency value is bigger or equal with 328.73Hz an smaller or equal with 331.58Hz, then e string has been tuned
       {
        string_1 = 1; //Set string_1 flag to finish tuning with e string
        k = 21; //Store value at k variable
        frequency = 0.00; //Reset frequency value
       } //else if (frequency...
       else //Else
       {
        frequency = 0.00; //Reset frequency value
       }
     } //while(string_1...*/
     lcd.clear(); //Clear LCD screen
     lcd.setCursor(1,0); //Set cursor at second column and first line
     lcd.print("Tune finished!"); //Print message at LCD
     delay(5000); //Delay 5 sec
     Menu(); //Call Menu () function
    } //if (menu==1)
    else if (menu==2) //Else if menu value is 2
    {
     lcd.clear(); //Clear LCD screen
```

```
    lcd.setCursor(0,0); //Set cursor at first column and first line
    lcd.print(" Half Step Down "); //Print message at LCD screen
    lcd.setCursor(4,1); //Set cursor at fifth column and first line
    lcd.print("Selected"); //Print message at LCD screen
    delay(1000); //Delay 1 sec
    lcd.clear(); //Clear LCD screen
    lcd.setCursor(1,0); //Set Cursor at third column and first line
    lcd.print("Hit Eb string!"); //Print message at LCD screen
    delay(1000); //Delay 1 sec
    lcd.clear(); //Clear LCD screen
    lcd.setCursor(1,0); //Set Cursor at third column and first line
    lcd.print("Eb -> 77.78 Hz"); //Print message at LCD screen
    lcd.setCursor(7,1); //Set Cursor at first column and second line
    lcd.print("Hz   Note"); //print message at LCD screen
    k = 21; //Store value at k variable
    string_6 = 0;//Reset the value at string_6
    string_5 = 0;//Reset the value at string_5
    string_4 = 0;//Reset the value at string_4
    string_3 = 0;//Reset the value at string_3
    string_2 = 0;//Reset the value at string_2
    string_1 = 0;//Reset the value at string_1
    while(string_6 != 1) //While string_6 flag isn't set repeat
    {
     if (k>20) //If k variable is bigger than 20, then
     {
      pinMode(39,OUTPUT); //Determine digital pin 39 as Output
      digitalWrite(39,LOW); //Set low voltage (GND) at pin 39
      delay(10); //Delay 10 msec
      pinMode(39,INPUT); //Determine digital pin 39 as Input
      k = 0; //Reset k variable
     } //if (k>20)
     FREQ_DETECTION(); //Call FREQ_DETECTION function (Change analog port for
the income signal)
     delay(500); //Delay 0.5 sec
     FREQ_DETECTION1(); //Call FREQ_DETECTION1 function (Calculate frequency
and stop interrupts by ADC)
```

```
    FREQ_DETECTION_STOP(); //Call FREQ_DETECTION_STOP function (Stop frequency detection)
    Serial.println("Eb String"); //Print message at Serial Port
    Serial.println(frequency); //Print frequency value at Serial Port
    lcd.setCursor(0,1); //Set cursor at first column and second line
    lcd.print("       "); //Print message at lcd screen
    lcd.setCursor(0,1); //Set cursor at first column and second line
    lcd.print(frequency); //Print frequency value at LCD screen
    k++; //Increase variable k value by one
    if (frequency < 77.68 && frequency > 65 && frequency > 0) //If frequency value is smaller than 77.68Hz and bigger than 65Hz and bigger than 0Hz, then
    {
     frequency = 0.00; //Reset frequency value
     for (i=0; i<=10; i++) //For 11 times repeat
     {
      Counterclockwise_EStep_Motor (); //Call function for E bass string step motor
     }
     delay(2); //Delay 2 msec
     EStep_Motor_Stop (); //Disable E bass string step motor
    } //if (frequency...
    else if (frequency > 77.88 && frequency < 103 && frequency > 0) //Else if frequency value is bigger than 77.88Hz and smaller than 103Hz and bigger than 0Hz, then
    {
     frequency = 0.00; //Reset frequency value
     for(i=0; i<=10; i++) //For 11 times repeat
     {
      Clockwise_EStep_Motor (); //Call function for E bass string step motor
     }
     delay(2); //Delay 2 msec
     EStep_Motor_Stop (); //Disable E bass string step motor
    } //else if (frequency...
    else if (frequency >=77.68 && frequency <=77.88) //Else if frequency value is bigger or equal with 77.68Hz and smaller or equal with 77.88Hz, then Eb bass string has been tuned
    {
     string_6 = 1; //Set string_6 flag to finish tuning with Eb bass string
```

```
        k = 21;  //Store value at k variable
        frequency = 0.00; //Reset frequency value
        lcd.clear(); //Clear LCD screen
        lcd.setCursor(1,0); //Set Cursor at third column and first line
        lcd.print("Hit Ab string!"); //Print message at LCD screen
        delay(1000); //Delay 1 sec
        lcd.clear(); //Clear LCD screen
        lcd.setCursor(1,0); //Set Cursor at third column and first line
        lcd.print("Ab -> 103.83 Hz"); //Print message at LCD screen
        lcd.setCursor(7,1); //Set Cursor at first column and second line
        lcd.print("Hz   Note"); //print message at LCD screen
      } //else if (frequency...
      else //Else
      {
        frequency = 0.00; //Reset frequency value
      }
    } //while(string_6...
    while(string_5 != 1) //While string_5 flag isn't set repeat
    {
      if (k>20) //If k variable is bigger than 20, then
      {
        pinMode(39,OUTPUT); //Determine digital pin 39 as Output
        digitalWrite(39,LOW); //Set low voltage (GND) at pin 39
        delay(10); //Delay 10 msec
        pinMode(39,INPUT); //Determine digital pin 39 as Input
        k = 0; //Reset k variable
      } //if (k>20)
      FREQ_DETECTION(); //Call FREQ_DETECTION function (Change analog port for the income signal)
      delay(500); //Delay 0.5 sec
      FREQ_DETECTION1(); //Call FREQ_DETECTION1 function (Calculate frequency and stop interrupts by ADC)
      FREQ_DETECTION_STOP(); //Call FREQ_DETECTION_STOP function (Stop frequency detection)
      Serial.println("Ab String"); //Print message at Serial Port
      Serial.println(frequency); //Print frequency value at Serial Port
```

```
    lcd.setCursor(0,1); //Set cursor at first column and second line
    lcd.print("       "); //Print message at lcd screen
    lcd.setCursor(0,1); //Set cursor at first column and second line
    lcd.print(frequency); //Print frequency value at LCD screen
    k++; //Increase variable k value by one
    if (frequency < 103.67 && frequency > 92 && frequency > 0) //If frequency value is
smaller than 103.67Hz and bigger than 92Hz and bigger than 0Hz, then
    {
     frequency = 0.00; //Reset frequency value
     for (i=0; i<=20; i++) //For 21 times repeat
     {
      Counterclockwise_AStep_Motor (); //Call function for A string step motor
     }
     delay(2); //Delay 2 msec
     AStep_Motor_Stop (); //Disable A string step motor
    } //if (frequency...
    else if (frequency > 103.96 && frequency < 130 && frequency > 0) //Else if
frequency value is bigger than 103.96Hz and smaller than 130Hz and bigger than 0Hz,
then
    {
     frequency = 0.00; //Reset frequency value
     for(i=0; i<=20; i++) //For 21 times repeat
     {
      Clockwise_AStep_Motor (); //Call function for A string step motor
     }
     delay(2); //Delay 2 msec
     AStep_Motor_Stop (); //Disable A string step motor
    } //else if (frequency...
    else if (frequency >=103.66 && frequency <=103.96) //Else if frequency value is
bigger or equal with 103.66Hz and smaller or equal with 103.96Hz, then Ab string has
been tuned
    {
     string_5 = 1; //Set string_5 flag to finish tuning with Ab string
     k = 21; //Store value at k variable
     frequency = 0.00; //Reset frequency value
     lcd.clear(); //Clear LCD screen
```

```
        lcd.setCursor(1,0); //Set Cursor at third column and first line
        lcd.print("Hit Db string!"); //Print message at LCD screen
        delay(1000); //Delay 1 sec
        lcd.clear(); //Clear LCD screen
        lcd.setCursor(1,0); //Set Cursor at third column and first line
        lcd.print("Db -> 138.59 Hz"); //Print message at LCD screen
        lcd.setCursor(7,1); //Set Cursor at first column and second line
        lcd.print("Hz   Note"); //print message at LCD screen
      } //else if (frequency...
      else //Else
      {
        frequency = 0.00; //Reset frequency value
      }
    } //while(string_5...
    while(string_4 != 1) //While string_4 flag isn't set repeat
    {
      if (k>20) //If k variable is bigger than 20, then
      {
        pinMode(39,OUTPUT); //Determine digital pin 39 as Output
        digitalWrite(39,LOW); //Set low voltage (GND) at pin 39
        delay(10); //Delay 10 msec
        pinMode(39,INPUT); //Determine digital pin 39 as Input
        k = 0;//Reset k variable
      } //if (k>20)
      FREQ_DETECTION(); //Call FREQ_DETECTION function (Change analog port for the income signal)
      delay(500); //Delay 0.5 sec
      FREQ_DETECTION1(); //Call FREQ_DETECTION1 function (Calculate frequency and stop interrupts by ADC)
      FREQ_DETECTION_STOP(); //Call FREQ_DETECTION_STOP function (Stop frequency detection)
      Serial.println("Db String"); //Print message at Serial Port
      Serial.println(frequency); //Print frequency value at Serial Port
      lcd.setCursor(0,1); //Set cursor at first column and second line
      lcd.print("       "); //Print message at lcd screen
      lcd.setCursor(0,1); //Set cursor at first column and second line
```

```
lcd.print(frequency); //Print frequency value at LCD screen
k++; //Increase variable k value by one
if (frequency < 138.35 && frequency > 123 && frequency > 0) //If frequency value is smaller than 138.35z and bigger than 123Hz and bigger than 0Hz, then
{
 frequency = 0.00; //Reset frequency value
 for (i=0; i<=30; i++) //For 31 times repeat
 {
  Counterclockwise_DStep_Motor (); //Call function for D string step motor
 }
 delay(2); //Delay 2 msec
 DStep_Motor_Stop (); //Disable D string step motor
} //if (frequency...
else if (frequency > 138.85 && frequency < 174 && frequency > 0) //Else if frequency value is bigger than 138.85Hz and smaller than 174Hz and bigger than 0Hz, then
{
 frequency = 0.00; //Reset frequency value
 for(i=0; i<=30; i++) //For 31 times repeat
 {
  Clockwise_DStep_Motor (); //Call function for D string step motor
 }
 delay(2); //Delay 2 msec
 DStep_Motor_Stop (); //Disable D string step motor
} //else if (frequency...
else if (frequency >=138.35 && frequency <=138.85) //Else if frequency value is bigger or equal with 138.35Hz and smaller or equal with 138.85Hz, then Db string has been tuned
{
 string_4 = 1; //Set string_4 flag to finish tuning with Db string
 k = 21;  //Store value at k variable
 frequency = 0.00; //Reset frequency value
 lcd.clear(); //Clear LCD screen
 lcd.setCursor(1,0); //Set Cursor at third column and first line
 lcd.print("Hit Gb string"); //Print message at LCD screen
 delay(1000); //Delay 1 sec
```

```
        lcd.clear(); //Clear LCD screen
        lcd.setCursor(1,0); //Set Cursor at third column and first line
        lcd.print("Gb -> 185 Hz"); //Print message at LCD screen
        lcd.setCursor(7,1); //Set Cursor at first column and second line
        lcd.print("Hz   Note"); //print message at LCD screen
       } //else if (frequency...
       else //Else
       {
        frequency = 0.00; //Reset frequency value
       }
      } //while(string_4...*/
      while(string_3 != 1) //While string_3 flag isn't set repeat
      {
       if (k>20) //If k variable is bigger than 20, then
       {
        pinMode(39,OUTPUT); //Determine digital pin 39 as Output
        digitalWrite(39,LOW); //Set low voltage (GND) at pin 39
        delay(10); //Delay 10 msec
        pinMode(39,INPUT); //Determine digital pin 39 as Input
        k = 0; //Reset k variable
       } //if (k>20)
       FREQ_DETECTION(); //Call FREQ_DETECTION function (Change analog port for the income signal)
       delay(500); //Delay 0.5 sec
       FREQ_DETECTION1(); //Call FREQ_DETECTION1 function (Calculate frequency and stop interrupts by ADC)
       FREQ_DETECTION_STOP(); //Call FREQ_DETECTION_STOP function (Stop frequency detection)
       Serial.println("Gb String"); //Print message at Serial Port
       Serial.println(frequency); //Print frequency value at Serial Port
       lcd.setCursor(0,1); //Set cursor at first column and second line
       lcd.print("       "); //Print message at lcd screen
       lcd.setCursor(0,1); //Set cursor at first column and second line
       lcd.print(frequency); //Print frequency value at LCD screen
       k++; //Increase variable k value by one
```

```
if (frequency < 184.90 && frequency > 164 && frequency > 0)  //If frequency value is smaller than 184.90Hz and bigger than 164Hz and bigger than 0Hz, then
{
 frequency = 0.00; //Reset frequency value
 for (i=0; i<=15; i++) //For 16 times repeat
 {
  Clockwise_GStep_Motor (); //Call function for G string step motor
 }
 delay(2); //Delay 2 msec
 GStep_Motor_Stop (); //Disable G string step motor
} //if (frequency...
else if (frequency > 185.10 && frequency < 233 && frequency > 0) //Else if frequency value is bigger than 185.10Hz and smaller than 233Hz and bigger than 0Hz, then
{
 frequency = 0.00; //Reset frequency value
 for(i=0; i<=15; i++) //For 16 times repeat
 {
  Counterclockwise_GStep_Motor (); //Call function for G string step motor
 }
 delay(2); //Delay 2 msec
 GStep_Motor_Stop (); //Disable G string step motor
} //else if (frequency...
else if (frequency >=184.90 && frequency <=185.10)//Else if frequency value is bigger or equal with 184.90Hz and smaller or equal with 185.10Hz, then Gb string has been tuned
{
 string_3 = 1; //Set string_3 flag to finish tuning with Gb string
 k = 21;  //Store value at k variable
 frequency = 0.00; //Reset frequency value
 lcd.clear(); //Clear LCD screen
 lcd.setCursor(1,0); //Set Cursor at third column and first line
 lcd.print("Hit Bb string"); //Print message at LCD screen
 delay(1000); //Delay 1 sec
 lcd.clear(); //Clear LCD screen
 lcd.setCursor(1,0); //Set Cursor at third column and first line
```

```
        lcd.print("Bb -> 233.08 Hz"); //Print message at LCD screen
        lcd.setCursor(7,1); //Set Cursor at first column and second line
        lcd.print("Hz   Note"); //print message at LCD screen
       } //else if (frequency...
       else //Else
       {
        frequency = 0.00;
       }
     } //while(string_3...*/
     while(string_2 != 1) //While string_2 flag isn't set repeat
     {
      if (k>20) //If k variable is bigger than 20, then
      {
       pinMode(39,OUTPUT); //Determine digital pin 39 as Output
       digitalWrite(39,LOW); //Set low voltage (GND) at pin 39
       delay(10); //Delay 10 msec
       pinMode(39,INPUT); //Determine digital pin 39 as Input
       k = 0; //Reset k variable
      } //if (k>20)
      FREQ_DETECTION(); //Call FREQ_DETECTION function (Change analog port for the income signal)
      delay(500); //Delay 0.5 sec
      FREQ_DETECTION1(); //Call FREQ_DETECTION1 function (Calculate frequency and stop interrupts by ADC)
      FREQ_DETECTION_STOP(); //Call FREQ_DETECTION_STOP function (Stop frequency detection)
      Serial.println("Bb String"); //Print message at Serial Port
      Serial.println(frequency); //Print frequency value at Serial Port
      lcd.setCursor(0,1); //Set cursor at first column and second line
      lcd.print("       "); //Print message at lcd screen
      lcd.setCursor(0,1); //Set cursor at first column and second line
      lcd.print(frequency); //Print frequency value at LCD screen
      k++; //Increase variable k value by one
      if (frequency < 232.98 && frequency > 207 && frequency > 0) //If frequency value is smaller than 232.98Hz and bigger than 207Hz and bigger than 0Hz, then
      {
```

```
        frequency = 0.00; //Reset frequency value
        for (i=0; i<=20; i++) //For 21 times repeat
        {
          Clockwise_BStep_Motor (); //Call function for B string step motor
        }
        delay(2); //Delay 2 msec
        BStep_Motor_Stop (); //Disable B string step motor
      } //if (frequency...
      else if (frequency > 233.18 && frequency < 293 && frequency > 0) //Else if
frequency value is bigger than 233.18Hz and smaller than 293Hz and bigger than 0Hz,
then
      {
        frequency = 0.00; //Reset frequency value
        for(i=0; i<=20; i++) //For 21 times repeat
        {
          Counterclockwise_BStep_Motor (); //Call function for B string step motor
        }
        delay(2); //Delay 2 msec
        BStep_Motor_Stop (); //Disable B string step motor
      } //else if (frequency...
      else if (frequency >=232.98 && frequency <=233.18) //Else if frequency value is
bigger or equal with 232.98Hz and smaller or equal with 233.18Hz, then Bb string has
been tuned
      {
        string_2 = 1; //Set string_2 flag to finish tuning with Bb string
        k = 21;  //Store value at k variable
        frequency = 0.00; //Reset frequency value
        lcd.clear(); //Clear LCD screen
        lcd.setCursor(2,0); //Set Cursor at third column and first line
        lcd.print("Hit eb string"); //Print message at LCD screen
        delay(1000); //Delay 1 sec
        lcd.clear(); //Clear LCD screen
        lcd.setCursor(1,0); //Set Cursor at third column and first line
        lcd.print("eb -> 311.13 Hz"); //Print message at LCD screen
        lcd.setCursor(7,1); //Set Cursor at first column and second line
        lcd.print("Hz   Note"); //print message at LCD screen
```

```
      } //else if (frequency...
      else //Else
      {
        frequency = 0.00; //Reset frequency value
      }
    } //while(string_2...*/
    while(string_1 != 1) //While string_1 flag isn't set repeat
    {
      if (k>20) //If k variable is bigger than 20, then
      {
        pinMode(39,OUTPUT); //Determine digital pin 39 as Output
        digitalWrite(39,LOW); //Set low voltage (GND) at pin 39
        delay(10); //Delay 10 msec
        pinMode(39,INPUT); //Determine digital pin 39 as Input
        k = 0;  //Reset k variable
      } //if (k>20)
      FREQ_DETECTION(); //Call FREQ_DETECTION function (Change analog port for the income signal)
      delay(500); //Delay 0.5 sec
      FREQ_DETECTION1(); //Call FREQ_DETECTION1 function (Calculate frequency and stop interrupts by ADC)
      FREQ_DETECTION_STOP(); //Call FREQ_DETECTION_STOP function (Stop frequency detection)
      Serial.println("eb String"); //Print message at Serial Port
      Serial.println(frequency); //Print frequency value at Serial Port
      lcd.setCursor(0,1); //Set cursor at first column and second line
      lcd.print("       "); //Print message at lcd screen
      lcd.setCursor(0,1); //Set cursor at first column and second line
      lcd.print(frequency); //Print frequency value at LCD screen
      k++; //Increase variable k value by one
      if (frequency < 310.18 && frequency > 277 && frequency > 0) //If frequency value is smaller than 310.18Hz and bigger than 277Hz and bigger than 0Hz, then
      {
        frequency = 0.00; //Reset frequency value
        for (i=0; i<=20; i++) //For 21 times repeat
        {
```

```
          Clockwise_eStep_Motor (); //Call function for e string step motor
         }
         delay(2); //Delay 2 msec
         eStep_Motor_Stop (); //Disable e string step motor
        } //if (frequency...
        else if (frequency > 312.70 && frequency < 392 && frequency > 0) //Else if
frequency value is bigger than 312.70Hz and smaller than 392Hz and bigger than 0Hz,
then
        {
         frequency = 0.00; //Reset frequency value
         for(i=0; i<=20; i++) //For 21 times repeat
         {
          Counterclockwise_eStep_Motor (); //Call function for e string step motor
         }
         delay(2); //Delay 2 msec
         eStep_Motor_Stop (); //Disable e string step motor
        } //else if (frequency...
        else if (frequency >=310.18 && frequency <=312.70) //Else if frequency value is
bigger or equal with 310.18Hz and smaller or equal with 312.70Hz, then eb string has been
tuned
        {
         string_1 = 1; //Set string_1 flag to finish tuning with eb string
         k = 21;  //Store value at k variable
         frequency = 0.00; //Reset frequency value
        } //else if (frequency...
        else //Else
        {
         frequency = 0.00; //Reset frequency value
        }
      } //while(string_1...*/
      lcd.clear(); //Clear LCD screen
      lcd.setCursor(1,0); //Set cursor at second column and first line
      lcd.print("Tune finished!"); //Print message at LCD screen
      delay(5000); //Delay 5 sec
      Menu(); //Call Menu() function
    } //else if (menu==2)
```

```
  else if (menu==3) //Else if menu variable is 3
  {
   lcd.clear(); //Clear LCD screen
   lcd.setCursor(2,0); //Set cursor at first column and first line
   lcd.print("Drop D Tune"); //Print message at LCD screen
   lcd.setCursor(4,1); //Set cursor at fifth column and first line
   lcd.print("Selected"); //Print message at LCD screen
   delay(1000); //Delay 1 sec
   lcd.clear(); //Clear LCD screen
   lcd.setCursor(2,0); //Set Cursor at third column and first line
   lcd.print("Hit D string"); //Print message at LCD screen
   delay(1000); //Delay 1 sec
   lcd.clear(); //Clear LCD screen
   lcd.setCursor(1,0); //Set Cursor at third column and first line
   lcd.print("D -> 73.42 Hz"); //Print message at LCD screen
   lcd.setCursor(7,1); //Set Cursor at first column and second line
   lcd.print("Hz   Note"); //print message at LCD screen
   //frequency = 0; //Reset the value at frequency variable
   k = 20;  //Store value at k variable
   string_6 = 0;//Reset the value at string_6
   string_5 = 0;//Reset the value at string_5
   string_4 = 0;//Reset the value at string_4
   string_3 = 0;//Reset the value at string_3
   string_2 = 0;//Reset the value at string_2
   string_1 = 0;//Reset the value at string_1
   while(string_6 != 1) //While string_6 flag isn't set repeat
   {
    if (k>20) //If k variable is bigger than 20, then
    {
     pinMode(39,OUTPUT); //Determine digital pin 39 as Output
     digitalWrite(39,LOW); //Set low voltage (GND) at pin 39
     delay(10); //Delay 10 msec
     pinMode(39,INPUT); //Determine digital pin 39 as Input
     k = 0; //Reset k variable
    } //if (k>20)
```

```
    FREQ_DETECTION(); //Call FREQ_DETECTION function (Change analog port for the income signal)
    delay(500); //Delay 0.5 sec
    FREQ_DETECTION1(); //Call FREQ_DETECTION1 function (Calculate frequency and stop interrupts by ADC)
    FREQ_DETECTION_STOP(); //Call FREQ_DETECTION_STOP function (Stop frequency detection)
    Serial.println("D String"); //Print message at Serial Port
    Serial.println(frequency); //Print frequency value at Serial Port
    lcd.setCursor(0,1); //Set cursor at first column and second line
    lcd.print("       "); //Print message at lcd screen
    lcd.setCursor(0,1); //Set cursor at first column and second line
    lcd.print(frequency); //Print frequency value at LCD screen
    k++; //Increase variable k value by one
    if (frequency < 73.37 && frequency > 65 && frequency > 0) //If frequency value is smaller than 73.37Hz and bigger than 65Hz and bigger than 0Hz, then
    {
     frequency = 0.00; //Reset frequency value
     for (i=0; i<=10; i++) //For 11 times repeat
     {
      Counterclockwise_EStep_Motor (); //Call function for E bass string step motor
     }
     delay(2); //Delay 2 msec
     EStep_Motor_Stop (); //Disable E bass string step motor
    } //if (frequency...
    else if (frequency > 73.47 && frequency < 103 && frequency > 0) //Else if frequency value is bigger than 73.47Hz and smaller than 103Hz and bigger than 0Hz, then
    {
     frequency = 0.00; //Reset frequency value
     for(i=0; i<=10; i++) //For 11 times repeat
     {
      Clockwise_EStep_Motor (); //Call function for E bass string step motor
     }
     delay(2); //Delay 2 msec
     EStep_Motor_Stop (); //Disable E bass string step motor
    } //else if (frequency...
```

```
      else if (frequency >=73.37 && frequency <=73.47) //Else if frequency value is bigger or equal with 73.37Hz and smaller or equal with 73.47Hz, then D bass string has been tuned
      {
        string_6 = 1; //Set string_6 flag to finish tuning with D bass string
        k = 21;  //Store value at k variable
        frequency = 0.00; //Reset frequency value
        lcd.clear(); //Clear LCD screen
        lcd.setCursor(2,0); //Set Cursor at third column and first line
        lcd.print("Hit A string"); //Print message at LCD screen
        delay(1000); //Delay 1 sec
        lcd.clear(); //Clear LCD screen
        lcd.setCursor(1,0); //Set Cursor at third column and first line
        lcd.print("a -> 110 Hz"); //Print message at LCD screen
        lcd.setCursor(7,1); //Set Cursor at first column and second line
        lcd.print("Hz   Note"); //print message at LCD screen
      } //else if (frequency...
      else //Else
      {
        frequency = 0.00; //Reset frequency value
      }
    } //while(string_6...
    while(string_5 != 1) //While string_5 flag isn't set repeat
    {
      if (k>20) //If k variable is bigger than 20, then
      {
        pinMode(39,OUTPUT); //Determine digital pin 39 as Output
        digitalWrite(39,LOW); //Set low voltage (GND) at pin 39
        delay(10); //Delay 10 msec
        pinMode(39,INPUT); //Determine digital pin 39 as Input
        k = 0; //Reset k variable
      } //if (k>20)
      FREQ_DETECTION(); //Call FREQ_DETECTION function (Change analog port for the income signal)
      delay(500); //Delay 0.5 sec
```

```
FREQ_DETECTION1(); //Call FREQ_DETECTION1 function (Calculate frequency and stop interrupts by ADC)
FREQ_DETECTION_STOP(); //Call FREQ_DETECTION_STOP function (Stop frequency detection)
Serial.println("A String"); //Print message at Serial Port
Serial.println(frequency); //Print frequency value at Serial Port
lcd.setCursor(0,1); //Set cursor at first column and second line
lcd.print("       "); //Print message at lcd screen
lcd.setCursor(0,1); //Set cursor at first column and second line
lcd.print(frequency); //Print frequency value at LCD screen
k++; //Increase variable k value by one
if (frequency < 109.80 && frequency > 92 && frequency > 0) //If frequency value is smaller than 109.80Hz and bigger than 92Hz and bigger than 0Hz, then
{
 frequency = 0.00; //Reset frequency value
 for (i=0; i<=20; i++) //For 21 times repeat
 {
  Counterclockwise_AStep_Motor (); //Call function for A string step motor
 }
 delay(2); //Delay 2 msec
 AStep_Motor_Stop (); //Disable A string step motor
} //if (frequency...
else if (frequency > 110.21 && frequency < 130 && frequency > 0) //Else if frequency value is bigger than 110.21Hz and smaller than 130Hz and bigger than 0Hz, then
{
 frequency = 0.00; //Reset frequency value
 for(i=0; i<=20; i++) //For 21 times repeat
 {
  Clockwise_AStep_Motor (); //Call function for A string step motor
 }
 delay(2); //Delay 2 msec
 AStep_Motor_Stop (); //Disable A string step motor
} //else if (frequency...
```

```
      else if (frequency >=109.80 && frequency <=110.21) //Else if frequency value is bigger or equal with 109.80Hz and smaller or equal with 110.21Hz, then A string has been tuned
      {
        string_5 = 1; //Set string_5 flag to finish tuning with A string
        k = 21;  //Store value at k variable
        frequency = 0.00; //Reset frequency value
        lcd.clear(); //Clear LCD screen
        lcd.setCursor(2,0); //Set Cursor at third column and first line
        lcd.print("Hit D string"); //Print message at LCD screen
        delay(1000); //Delay 1 sec
        lcd.clear(); //Clear LCD screen
        lcd.setCursor(1,0); //Set Cursor at third column and first line
        lcd.print("D -> 146.83 Hz"); //Print message at LCD screen
        lcd.setCursor(7,1); //Set Cursor at first column and second line
        lcd.print("Hz   Note"); //print message at LCD screen
      } //else if (frequency...
      else  //Else
      {
        frequency = 0.00; //Reset frequency value
      }
    } //while(string_5...
    while(string_4 != 1) //While string_4 flag isn't set repeat
    {
      if (k>20) //If k variable is bigger than 20, then
      {
        pinMode(39,OUTPUT); //Determine digital pin 39 as Output
        digitalWrite(39,LOW); //Set low voltage (GND) at pin 39
        delay(10); //Delay 10 msec
        pinMode(39,INPUT); //Determine digital pin 39 as Input
        k = 0; //Reset k variable
      } //if (k>20)
      FREQ_DETECTION(); //Call FREQ_DETECTION function (Change analog port for the income signal)
      delay(500); //Delay 0.5 sec
```

```
FREQ_DETECTION1(); //Call FREQ_DETECTION1 function (Calculate frequency and stop interrupts by ADC)
FREQ_DETECTION_STOP(); //Call FREQ_DETECTION_STOP function (Stop frequency detection)
Serial.println("D String"); //Print message at Serial Port
Serial.println(frequency); //Print frequency value at Serial Port
lcd.setCursor(0,1); //Set cursor at first column and second line
lcd.print("       "); //Print message at lcd screen
lcd.setCursor(0,1); //Set cursor at first column and second line
lcd.print(frequency); //Print frequency value at LCD screen
k++; //Increase variable k value by one
if (frequency < 146.73 && frequency > 123 && frequency > 0)  //If frequency value is smaller than 146.73Hz and bigger than 123Hz and bigger than 0Hz, then
{
  frequency = 0.00; //Reset frequency value
  for (i=0; i<=30; i++) //For 31 times repeat
  {
    Counterclockwise_DStep_Motor (); //Call function for D string step motor
  }
  delay(2); //Delay 2 msec
  DStep_Motor_Stop (); //Disable D string step motor
} //if (frequency...
else if (frequency > 146.93 && frequency < 174 && frequency > 0) //Else if frequency value is bigger than 146.93Hz and smaller than 174Hz and bigger than 0Hz, then
{
  frequency = 0.00; //Reset frequency value
  for(i=0; i<=30; i++) //For 31 times repeat
  {
    Clockwise_DStep_Motor (); //Call function for D string step motor
  }
  delay(2); //Delay 2 msec
  DStep_Motor_Stop (); //Disable D string step motor
} //else if (frequency...
```

```
      else if (frequency >=146.73 && frequency <=146.93) //Else if frequency value is bigger or equal with 146.73Hz and smaller or equal with 146.93Hz, then D string has been tuned
      {
        string_4 = 1; //Set string_4 flag to finish tuning with D string
        k = 20;  //Store value at k variable
        frequency = 0.00; //Reset frequency value
        lcd.clear(); //Clear LCD screen
        lcd.setCursor(2,0); //Set Cursor at third column and first line
        lcd.print("Hit G string"); //Print message at LCD screen
        delay(1000); //Delay 1 sec
        lcd.clear(); //Clear LCD screen
        lcd.setCursor(1,0); //Set Cursor at third column and first line
        lcd.print("G -> 196 Hz"); //Print message at LCD screen
        lcd.setCursor(7,1); //Set Cursor at first column and second line
        lcd.print("Hz   Note"); //print message at LCD screen
      } //else if (frequency...
      else  //Else
      {
        frequency = 0.00; //Reset frequency value
      }
    } //while(string_4...*/
    while(string_3 != 1) //While string_3 flag isn't set repeat
    {
      if (k>20) //If k variable is bigger than 20, then
      {
        pinMode(39,OUTPUT); //Determine digital pin 39 as Output
        digitalWrite(39,LOW); //Set low voltage (GND) at pin 39
        delay(10); //Delay 10 msec
        pinMode(39,INPUT); //Determine digital pin 39 as Input
        k = 0; //Reset k variable
      } //if (k>20)
      FREQ_DETECTION(); //Call FREQ_DETECTION function (Change analog port for the income signal)
      delay(500); //Delay 0.5 sec
```

```
FREQ_DETECTION1(); //Call FREQ_DETECTION1 function (Calculate frequency and stop interrupts by ADC)
FREQ_DETECTION_STOP(); //Call FREQ_DETECTION_STOP function (Stop frequency detection)
Serial.println("G String"); //Print message at Serial Port
Serial.println(frequency); //Print frequency value at Serial Port
lcd.setCursor(0,1); //Set cursor at first column and second line
lcd.print("       "); //Print message at lcd screen
lcd.setCursor(0,1); //Set cursor at first column and second line
lcd.print(frequency); //Print frequency value at LCD screen
k++; //Increase variable k value by one
if (frequency < 195.90 && frequency > 164 && frequency > 0) //If frequency value is smaller than 195.90Hz and bigger than 164Hz and bigger than 0Hz, then
{
 frequency = 0.00; //Reset frequency value
 for (i=0; i<=15; i++) //For 16 times repeat
 {
  Clockwise_GStep_Motor (); //Call function for G string step motor
 }
 delay(2); //Delay 2 msec
 GStep_Motor_Stop (); //Disable G string step motor
} //if (frequency...
else if (frequency > 196.25 && frequency < 233 && frequency > 0) //Else if frequency value is bigger than 196.25Hz and smaller than 233Hz and bigger than 0Hz, then
{
 frequency = 0.00; //Reset frequency value
 for(i=0; i<=15; i++) //For 16 times repeat
 {
  Counterclockwise_GStep_Motor (); //Call function for G string step motor
 }
 delay(2); //Delay 2 msec
 GStep_Motor_Stop (); //Disable G string step motor
} //else if (frequency...
```

```
      else if (frequency >=195.90 && frequency <=196.25) //Else if frequency value is bigger or equal with 195.90Hz and smaller or equal with 196.25Hz, then G string has been tuned
      {
        string_3 = 1; //Set string_3 flag to finish tuning with G string
        k = 21;  //Store value at k variable
        frequency = 0.00; //Reset frequency value
        lcd.clear(); //Clear LCD screen
        lcd.setCursor(2,0); //Set Cursor at third column and first line
        lcd.print("Hit B string"); //Print message at LCD screen
        delay(1000); //Delay 1 sec
        lcd.clear(); //Clear LCD screen
        lcd.setCursor(1,0); //Set Cursor at third column and first line
        lcd.print("B -> 246.94 Hz"); //Print message at LCD screen
        lcd.setCursor(7,1); //Set Cursor at first column and second line
        lcd.print("Hz   Note"); //print message at LCD screen
      } //else if (frequency...
      else  //Else
      {
        frequency = 0.00; //Reset frequency value
      }
    } //while(string_3...*/
    while(string_2 != 1) //While string_2 flag isn't set repeat
    {
      if (k>20) //If k variable is bigger than 20, then
      {
        pinMode(39,OUTPUT); //Determine digital pin 39 as Output
        digitalWrite(39,LOW); //Set low voltage (GND) at pin 39
        delay(10); //Delay 10 msec
        pinMode(39,INPUT); //Determine digital pin 39 as Input
        k = 0; //Reset k variable
      } //if (k>20)
      FREQ_DETECTION(); //Call FREQ_DETECTION function (Change analog port for the income signal)
      delay(500); //Delay 0.5 sec
```

```
    FREQ_DETECTION1(); //Call FREQ_DETECTION1 function (Calculate frequency and stop interrupts by ADC)
    FREQ_DETECTION_STOP(); //Call FREQ_DETECTION_STOP function (Stop frequency detection)
    Serial.println("B String"); //Print message at Serial Port
    Serial.println(frequency); //Print frequency value at Serial Port
    lcd.setCursor(0,1); //Set cursor at first column and second line
    lcd.print("       "); //Print message at lcd screen
    lcd.setCursor(0,1); //Set cursor at first column and second line
    lcd.print(frequency); //Print frequency value at LCD screen
    k++; //Increase variable k value by one
    if (frequency < 246.54 && frequency > 207 && frequency > 0) //If frequency value is smaller than 246.54Hz and bigger than 207Hz and bigger than 0Hz, then
    {
     frequency = 0.00; //Reset frequency value
     for (i=0; i<=20; i++) //For 21 times repeat
     {
      Clockwise_BStep_Motor (); //Call function for B string step motor
     }
     delay(2); //Delay 2 msec
     BStep_Motor_Stop (); //Disable B string step motor
    } //if (frequency...
    else if (frequency > 247.04 && frequency < 293 && frequency > 0) //Else if frequency value is bigger than 247.04Hz and smaller than 293Hz and bigger than 0Hz, then
    {
     frequency = 0.00; //Reset frequency value
     for(i=0; i<=20; i++) //For 21 times repeat
     {
      Counterclockwise_BStep_Motor (); //Call function for B string step motor
     }
     delay(2); //Delay 2 msec
     BStep_Motor_Stop (); //Disable B string step motor
    } //else if (frequency...
```

```
      else if (frequency >=246.55 && frequency <=247.04) //Else if frequency value is bigger or equal with 246.55Hz and smaller or equal with 247.04Hz, then B string has been tuned
      {
        string_2 = 1; //Set string_2 flag to finish tuning with B string
        k = 21;  //Store value at k variable
        frequency = 0.00; //Reset frequency value
        lcd.clear(); //Clear LCD screen
        lcd.setCursor(2,0); //Set Cursor at third column and first line
        lcd.print("Hit e string"); //Print message at LCD screen
        delay(1000); //Delay 1 sec
        lcd.clear(); //Clear LCD screen
        lcd.setCursor(1,0); //Set Cursor at third column and first line
        lcd.print("e -> 329.63 Hz"); //Print message at LCD screen
        lcd.setCursor(7,1); //Set Cursor at first column and second line
        lcd.print("Hz   Note"); //print message at LCD screen
      } //else if (frequency...
      else  //Else
      {
        frequency = 0.00; //Reset frequency value
      }
    } //while(string_2...*/
    while(string_1 != 1) //While string_1 flag isn't set repeat
    {
      if (k>20) //If k variable is bigger than 20, then
      {
        pinMode(39,OUTPUT); //Determine digital pin 39 as Output
        digitalWrite(39,LOW); //Set low voltage (GND) at pin 39
        delay(10); //Delay 10 msec
        pinMode(39,INPUT); //Determine digital pin 39 as Input
        k = 0; //Reset k variable
      } //if (k>20)
      FREQ_DETECTION(); //Call FREQ_DETECTION function (Change analog port for the income signal)
      delay(500); //Delay 0.5 sec
```

```
FREQ_DETECTION1(); //Call FREQ_DETECTION1 function (Calculate frequency and stop interrupts by ADC)
FREQ_DETECTION_STOP(); //Call FREQ_DETECTION_STOP function (Stop frequency detection)
Serial.println("e String"); //Print message at Serial Port
Serial.println(frequency); //Print frequency value at Serial Port
lcd.setCursor(0,1); //Set cursor at first column and second line
lcd.print("       "); //Print message at lcd screen
lcd.setCursor(0,1);  //Set cursor at first column and second line
lcd.print(frequency); //Print frequency value at LCD screen
k++; //Increase variable k value by one
if (frequency < 328.73 && frequency > 277 && frequency > 0) //If frequency value is smaller than 328.73Hz and bigger than 277Hz and bigger than 0Hz, then
{
 frequency = 0.00; //Reset frequency value
 for (i=0; i<=20; i++) //For 21 times repeat
 {
  Clockwise_eStep_Motor (); //Call function for e string step motor
 }
 delay(2); //Delay 2 msec
 eStep_Motor_Stop (); //Disable e string step motor
} //if (frequency...
else if (frequency > 331.58 && frequency < 392 && frequency > 0) //Else if frequency value is bigger than 331.58Hz and smaller than 392Hz and bigger than 0Hz, then
{
 frequency = 0.00; //Reset frequency value
 for(i=0; i<=20; i++) //For 21 times repeat
 {
  Counterclockwise_eStep_Motor (); //Call function for e string step motor
 }
 delay(2); //Delay 2 msec
 eStep_Motor_Stop (); //Disable e string step motor
} //else if (frequency...
```

```
      else if (frequency >=328.73 && frequency <=331.58) //Else if frequency value is bigger or equal with 328.73Hz and smaller or equal with 331.58Hz, then e string has been tuned
      {
        string_1 = 1; //Set string_1 flag to finish tuning with e string
        k = 21;  //Store value at k variable
        frequency = 0.00; //Reset frequency value
      } //else if (frequency...
      else //Else
      {
        frequency = 0.00; //Reset frequency value
      }
    } //while(string_1...*/
    lcd.clear(); //Clear LCD screen
    lcd.setCursor(1,0); //Set cursor at second column and first line
    lcd.print("Tune finished!"); //Print message at LCD screen
    delay(5000); //Delay 5 sec
    Menu(); //Call Menu() function
  } //else if (menu = = 3
  else if (menu==4) //Else if menu variable is 4 then
  {
    Manual_Finish = 0; //Reset Manual_Finish variable
    manual_menu = 1; //Store number 1 at manual_menu variable
    lcd.clear(); //Clear LCD screen
    lcd.setCursor(0,0); //Set cursor at 1st column and 1st line
    lcd.print("Choose string!"); //Print message at LCD screen
    delay(2500); //Delay 2.5 sec
    lcd.clear(); //Clear LCD screen
    while(Manual_Finish == 0) //While Manual_Finish is 0 then
    {
      Manual_Button_Read(); //Call Function for LCD's button read
      if(manual_menu == 1) //If E string has been selected //If manual_menu variable is 1 then
      {
        lcd.setCursor(0,0); //Move the cursor to the 1st line and the 1st column
        lcd.print(">E"); //Print message to the lcd screen
```

```
        lcd.setCursor(0,1); //Move the cursor to the 2nd line and the 1st column
        lcd.print(" A"); //Print message to the lcd screen
        lcd.setCursor(2,0); //Move the cursor to the 1st line and the 3rd column
        lcd.print(" D"); //Print message to the lcd screen
        lcd.setCursor(2,1); //Move the cursor to the 2nd line and the 3rd column
        lcd.print(" G"); //Print message to the lcd screen
        lcd.setCursor(4,0); //Move the cursor to the 1st line and the 5th column
        lcd.print(" B"); //Print message to the lcd screen
        lcd.setCursor(4,1); //Move the cursor to the 2nd line and the 5th column
        lcd.print(" e"); //Print message to the lcd screen
        lcd.setCursor(6,0); //Move the cursor to the 1st line and the 7th column
        lcd.print(" Back"); //Print message to the lcd screen
        if (Manual_Select == 1) //If Manual_Select is 1, If the E string has been selected
then
        {
          Manual_Select = 0; //Reset Manual_Selected variable
          Finish_Control = 0; //Reset Finish_Control variable
          lcd.clear(); //Clear LCD screen
          lcd.setCursor(0,0); //Set cursor at 1st column and 1st line
          lcd.print(" Press Right or "); //Print message at LCD screen
          lcd.setCursor(0,1); //Set cursor at 1st column and 2nd line
          lcd.print("  Left Buttons  "); //Print message at LCD screen
          delay(3000); //Delay 3 sec
          lcd.clear(); //Clear LCD screen
          lcd.setCursor(0,0); //Set cursor at 1st column and 1st line
          lcd.print("  Press SELECT  "); //Print message at LCD screen
          lcd.setCursor(0,1); //Set cursor at 1st column and 2nd line
          lcd.print("   to go back   "); //Print message at LCD screen
          while (Finish_Control == 0) //While Finish_Control variable is 0 then
          {
            Manual_Step_Motor_Control(); //Call Function for reading LCD's shield buttons
            delay(50); //Delay 50 msec
            if (Turn_Right == 1) //If Turn_Right flag is 1 then
            {
              //Serial.println("Turn Right");
              for(i=0;i<=20;i++) //For 21 reps
```

```
            {
              Clockwise_EStep_Motor (); //Turn E string's step motor clockwise
            }
            Turn_Right = 0; //Reset Turn_Right flag
          } //if
          else if (Turn_Left == 1) //Else if Turn_Left flag is 1 then
          {
            for(i=0;i<=20;i++) //For 21 reps
            {
              Counterclockwise_EStep_Motor (); //Turn E string's step motor
counterclickwise
            }
            Turn_Left = 0; //Reset Turn_Left flag
          } //else if (Turn_Left...
        } //while (Finish_Control...
      } //if (Manual_Select...
    } //if (manual_menu...
    else if (manual_menu == 2) //If A string has been selected
    {
      lcd.setCursor(0,0); //Move the cursor to the 1st line and the 1st column
      lcd.print(" E"); //Print message to the lcd screen
      lcd.setCursor(0,1); //Move the cursor to the 2nd line and the 1st column
      lcd.print(">A"); //Print message to the lcd screen
      lcd.setCursor(2,0); //Move the cursor to the 1st line and the 3rd column
      lcd.print(" D"); //Print message to the lcd screen
      lcd.setCursor(2,1); //Move the cursor to the 2nd line and the 3rd column
      lcd.print(" G"); //Print message to the lcd screen
      lcd.setCursor(4,0); //Move the cursor to the 1st line and the 5th column
      lcd.print(" B"); //Print message to the lcd screen
      lcd.setCursor(4,1); //Move the cursor to the 2nd line and the 5th column
      lcd.print(" e"); //Print message to the lcd screen
      lcd.setCursor(6,0); //Move the cursor to the 1st line and the 7th column
      lcd.print(" Back"); //Print message to the lcd screen
      if (Manual_Select == 1) //If Manual_Select flag is 1 then
      {
        Manual_Select = 0; //Reset Manual_Select flag
```

```
        Finish_Control = 0; //Reset Finish_Control flag
        lcd.clear(); //Clear LCD screen
        lcd.setCursor(0,0); //Set cursor at 1st column and 1st line
        lcd.print(" Press Right or "); //Print message at LCD screen
        lcd.setCursor(0,1); //Set cursor at 1st column and 2nd line
        lcd.print("  Left Buttons  "); //Print message at LCD screen
        delay(3000); //Delay 3 sec
        lcd.clear(); //Clear LCD screen
        lcd.setCursor(0,0); //Set cursor at 1st column and 1st line
        lcd.print("  Press SELECT  "); //Print message at LCD screen
        lcd.setCursor(0,1); //Set cursor at 1st column and 2nd line
        lcd.print("   to go back   "); //Print message at LCD screen
        while (Finish_Control == 0) //While Finish_Control flag is 0 then
        {
         Manual_Step_Motor_Control(); //Call of Manual_Step_Motor_Control function
         delay(50); //Delay 50 msec
         if (Turn_Right == 1) //If Turn right flag is 1 then
         {
          for(i=0;i<=20;i++) //For 21 reps
          {
           Clockwise_AStep_Motor (); //Turn A Step Motor Clockwise
          }
          Turn_Right = 0; //Reset Turn_Right flag
         }//if (Turn_Right)...
         else if (Turn_Left == 1) //Else if Turn_Left flag is 1 then
         {
          for(i=0;i<=20;i++) //For 21 reps
          {
           Counterclockwise_AStep_Motor (); //Turn A step motor counter clockwise
          }
          Turn_Left = 0; //Reset Turn Left flag
         } //else if (Turn_Left)...
        } //while (Finish_Control)...
       } //if (Manual_Select == 1)...
      } //else if (manual_menu == 1)...
      else if (manual_menu == 3) //If D sting has been selected
```

```
{
 lcd.setCursor(0,0); //Move the cursor to the 1st line and the 1st column
 lcd.print(" E"); //Print message to the lcd screen
 lcd.setCursor(0,1); //Move the cursor to the 2nd line and the 1st column
 lcd.print(" A"); //Print message to the lcd screen
 lcd.setCursor(2,0); //Move the cursor to the 1st line and the 3rd column
 lcd.print(">D"); //Print message to the lcd screen
 lcd.setCursor(2,1); //Move the cursor to the 2nd line and the 3rd column
 lcd.print(" G"); //Print message to the lcd screen
 lcd.setCursor(4,0); //Move the cursor to the 1st line and the 5th column
 lcd.print(" B"); //Print message to the lcd screen
 lcd.setCursor(4,1); //Move the cursor to the 2nd line and the 5th column
 lcd.print(" e"); //Print message to the lcd screen
 lcd.setCursor(6,0); //Move the cursor to the 1st line and the 7th column
 lcd.print(" Back"); //Print message to the lcd screen
 if (Manual_Select == 1) //If Manual_Select flag is 1 then
 {
  Manual_Select = 0; //Reset Manual_Reset
  Finish_Control = 0; //Reset Finish_Control
  lcd.clear(); //Clear LCD screen
  lcd.setCursor(0,0); //Set cursor at 1st column and 1st line
  lcd.print(" Press Right or "); //Print message at LCD screen
  lcd.setCursor(0,1); //Set cursor at 1st column and 2nd line
  lcd.print("  Left Buttons  "); //Print message at LCD screen
  delay(3000); //Delay 3 sec
  lcd.clear(); //Clear LCD screen
  lcd.setCursor(0,0); //Set cursor at 1st column and 1st line
  lcd.print("  Press SELECT  "); //Print message at LCD screen
  lcd.setCursor(0,1); //Set cursor at 1st column and 2nd line
  lcd.print("   to go back   "); //Print message at LCD screen
  while (Finish_Control == 0) // While Finish_Control is 0 then
  {
   Manual_Step_Motor_Control(); //Call Manual_Step_Motor_Control
   delay(50); //Delay 50 msec
   if (Turn_Right == 1) //If Turn_Right flag is 1 then
   {
```

```
          for(i=0;i<=20;i++) //For 21 reps
          {
            Clockwise_DStep_Motor (); //Turn D step motor clockwise
          }
          Turn_Right = 0; //Reset Turn_Right flag
        } //if (Turn_Right == 1)...
        else if (Turn_Left == 1) //Else if Turn_Left flag is 1 then
        {
          for(i=0;i<=20;i++) //For 21 reps
          {
            Counterclockwise_DStep_Motor (); //Turn D step motor Counter Clockwise
          }
          Turn_Left = 0; //Reset Turn_Left flag
        } //else if (Turn_Left == 1)
      } //while (Finish_Control == 0)
    } //if (Manual_Select == 1)
  } //else if (manual_menu ==3)
  else if (manual_menu == 4) //If G string has been selected
  {
    lcd.setCursor(0,0); //Move the cursor to the 1st line and the 1st column
    lcd.print(" E"); //Print message to the lcd screen
    lcd.setCursor(0,1); //Move the cursor to the 2nd line and the 1st column
    lcd.print(" A"); //Print message to the lcd screen
    lcd.setCursor(2,0); //Move the cursor to the 1st line and the 3rd column
    lcd.print(" D"); //Print message to the lcd screen
    lcd.setCursor(2,1); //Move the cursor to the 2nd line and the 3rd column
    lcd.print(">G"); //Print message to the lcd screen
    lcd.setCursor(4,0); //Move the cursor to the 1st line and the 5th column
    lcd.print(" B"); //Print message to the lcd screen
    lcd.setCursor(4,1); //Move the cursor to the 2nd line and the 5th column
    lcd.print(" e"); //Print message to the lcd screen
    lcd.setCursor(6,0); //Move the cursor to the 1st line and the 7th column
    lcd.print(" Back"); //Print message to the lcd screen
    if (Manual_Select == 1) //If Manual_Select flag is 1 then
    {
      Manual_Select = 0; //Reset Manual_Select
```

```
      Finish_Control = 0; //Reset Finish_Control
      lcd.clear(); //Clear LCD screen
      lcd.setCursor(0,0); //Set cursor at 1st column and 1st line
      lcd.print(" Press Right or "); //Print message at LCD screen
      lcd.setCursor(0,1); //Set cursor at 1st column and 2nd line
      lcd.print("  Left Buttons  "); //Print message at LCD screen
      delay(3000); //Delay 3 sec
      lcd.clear(); //Clear LCD screen
      lcd.setCursor(0,0); //Set cursor at 1st column and 1st line
      lcd.print("  Press SELECT  "); //Print message at LCD screen
      lcd.setCursor(0,1); //Set cursor at 1st column and 2nd line
      lcd.print("   to go back   "); //Print message at LCD screen
      while (Finish_Control == 0) //While Finish_Control flag is 0 then
      {
       Manual_Step_Motor_Control(); //Call Manual_Step_Motor_Control function
       delay(50); //Delay 50 msec
       if (Turn_Right == 1) //If Turn_Right flag is 1 then
       {
        for(i=0;i<=20;i++) //For 21 reps
        {
         Clockwise_GStep_Motor (); //Turn G step motor clockwise
        }
        Turn_Right = 0; //Reset Turn_Right flag
       } //if (Turn_Right...
       else if (Turn_Left == 1) //Else if Turn_Left flag is 1 then
       {
        for(i=0;i<=20;i++) //For 21 reps
        {
         Counterclockwise_GStep_Motor (); //Turn G step motor counter clockwise
        }
        Turn_Left = 0; //Reset Turn_Left flag
       } //else if (Turn_Left...
      } //while (Finish_Control...
     } //if (Manual_Select...
    } //else if (manual_menu...
    else if (manual_menu == 5) //if the B string has been selected
```

```
{
 lcd.setCursor(0,0); //Move the cursor to the 1st line and the 1st column
 lcd.print(" E"); //Print message to the lcd screen
 lcd.setCursor(0,1); //Move the cursor to the 2nd line and the 1st column
 lcd.print(" A"); //Print message to the lcd screen
 lcd.setCursor(2,0); //Move the cursor to the 1st line and the 3rd column
 lcd.print(" D"); //Print message to the lcd screen
 lcd.setCursor(2,1); //Move the cursor to the 2nd line and the 3rd column
 lcd.print(" G"); //Print message to the lcd screen
 lcd.setCursor(4,0); //Move the cursor to the 1st line and the 5th column
 lcd.print(">B"); //Print message to the lcd screen
 lcd.setCursor(4,1); //Move the cursor to the 2nd line and the 5th column
 lcd.print(" e"); //Print message to the lcd screen
 lcd.setCursor(6,0); //Move the cursor to the 1st line and the 7th column
 lcd.print(" Back"); //Print message to the lcd screen
 if (Manual_Select == 1) //if  Manual_Select is 1 then
 {
  Manual_Select = 0; //Reset Manual_Select
  Finish_Control = 0; //Reset Finish_Control
  lcd.clear(); //Clear LCD screen
  lcd.setCursor(0,0); //Set cursor at 1st column and 1st line
  lcd.print(" Press Right or "); //Print message at LCD screen
  lcd.setCursor(0,1); //Set cursor at 1st column and 2nd line
  lcd.print("  Left Buttons  "); //Print message at LCD screen
  delay(3000); //Delay 3 sec
  lcd.clear(); //Clear LCD screen
  lcd.setCursor(0,0); //Set cursor at 1st column and 1st line
  lcd.print("  Press SELECT  "); //Print message at LCD screen
  lcd.setCursor(0,1); //Set cursor at 1st column and 2nd line
  lcd.print("   to go back   "); // Print message at LCD screen
  while (Finish_Control == 0) //While Finish_Control is 0 then
  {
   Manual_Step_Motor_Control(); //Call Manual_Step_Motor_Control function
   delay(50); //delay 50 msec
   if (Turn_Right == 1) //If Turn_Right flag is 1 then
   {
```

```
          for(i=0;i<=20;i++) //For 21 reps
          {
            Clockwise_BStep_Motor (); //Turn B step motor clockwise
          }
          Turn_Right = 0; //Reset Turn_Right flag
        } //if (Turn_Right...
        else if (Turn_Left == 1) //Else if Turn_Left is 1 then
        {
          for(i=0;i<=20;i++) //For 21 reps
          {
            Counterclockwise_BStep_Motor (); //Turn B step motor counter clockwise
          }
          Turn_Left = 0; // Reset Turn_Left flag
        } //else if (Turn_Left...
      } //while (Finish_Right...
    } //if (Manual_Select...
  } //else if (manual_menu...
  else if (manual_menu == 6) //If the e string has been selected
  {
    lcd.setCursor(0,0); //Move the cursor to the 1st line and the 1st column
    lcd.print(" E"); //Print message to the lcd screen
    lcd.setCursor(0,1); //Move the cursor to the 2nd line and the 1st column
    lcd.print(" A"); //Print message to the lcd screen
    lcd.setCursor(2,0); //Move the cursor to the 1st line and the 3rd column
    lcd.print(" D"); //Print message to the lcd screen
    lcd.setCursor(2,1); //Move the cursor to the 2nd line and the 3rd column
    lcd.print(" G"); //Print message to the lcd screen
    lcd.setCursor(4,0); //Move the cursor to the 1st line and the 5th column
    lcd.print(" B"); //Print message to the lcd screen
    lcd.setCursor(4,1); //Move the cursor to the 2nd line and the 5th column
    lcd.print(">e"); //Print message to the lcd screen
    lcd.setCursor(6,0); //Move the cursor to the 1st line and the 7th column
    lcd.print(" Back"); //Print message to the lcd screen
    if (Manual_Select == 1) //If Manual_Select is 1 then
    {
      Manual_Select = 0; //Reset Manual_Select
```

```
        Finish_Control = 0; //Reset Finish_Control
        lcd.clear(); //Clear LCD screen
        lcd.setCursor(0,0); //Set cursor at 1st column and 1st line
        lcd.print(" Press Right or "); //Print message at LCD screen
        lcd.setCursor(0,1); //Set cursor at 1st column and 2nd line
        lcd.print("  Left Buttons  "); //Print message at LCD screen
        delay(3000); //Delay 3 sec
        lcd.clear(); //Clear LCD screen
        lcd.setCursor(0,0); //Set cursor at 1st column and 1st line
        lcd.print("  Press SELECT  "); //Print ,message at LCD screen
        lcd.setCursor(0,1); //Set cursor at 1st column and 2nd line
        lcd.print("   to go back   "); //Print message at LCD screen
        while (Finish_Control == 0) //While Finish_Control is 0 then
        {
          Manual_Step_Motor_Control(); //Call Manual_Step_Motor_Control function
          delay(50); //Delay 50 msec
          if (Turn_Right == 1) //If Turn_Right flag is 1 then
          {
            for(i=0;i<=20;i++) //For 21 reps
            {
              Clockwise_eStep_Motor (); //Turn e Step Motor Clockwise
            }
            Turn_Right = 0; //Reset Turn_Right flag
          } //if (Turn_Right...
          else if (Turn_Left == 1) //Else if Turn_Left flag is 1 then
          {
            for(i=0;i<=20;i++) //For 21 reps
            {
              Counterclockwise_eStep_Motor (); //Turn e Step_Motor Counterclockwise
            }
            Turn_Left = 0; //Reset Turn_Left flag
          } //else if (Turn_Left...
        } //while (Finish_Control...
      } //if (Manual_Select...
    } //else if (manual_menu...
    else if (manual_menu == 7) //If Back to the main menu has been selected
```

```
      {
        lcd.setCursor(0,0); //Move the cursor to the 1st line and the 1st column
        lcd.print(" E"); //Print message to the lcd screen
        lcd.setCursor(0,1); //Move the cursor to the 2nd line and the 1st column
        lcd.print(" A"); //Print message to the lcd screen
        lcd.setCursor(2,0); //Move the cursor to the 1st line and the 3rd column
        lcd.print(" D"); //Print message to the lcd screen
        lcd.setCursor(2,1); //Move the cursor to the 2nd line and the 3rd column
        lcd.print(" G"); //Print message to the lcd screen
        lcd.setCursor(4,0); //Move the cursor to the 1st line and the 5th column
        lcd.print(" B"); //Print message to the lcd screen
        lcd.setCursor(4,1); //Move the cursor to the 2nd line and the 5th column
        lcd.print(" e"); //Print message to the lcd screen
        lcd.setCursor(6,0); //Move the cursor to the 1st line and the 7th column
        lcd.print(">Back"); //Print message to the lcd screen
        if (Manual_Select == 1) //If Manual_Select is 1 then
        {
          Manual_Select = 0; //Reset Manual_Select flag
          Manual_Finish = 1; //Set Manual_Finish flag
        } //if (Manual_Select...
      } //else if (manual_menu...
      delay(50); //delay 50 msec
    } //while (Manual_Finish...
  } //else if (menu == 7...
  Menu(); //Call Menu() function
  delay(200); //Delay 200 msec
 } //else if (x<800...
}//void ButtonRead

void reset(){ // reset () function
 index=0; //Reset value of index variable
 noMatch=0; //Reset value of noMatch variable
 maxSlope=0; //Reset value of maxSlope variable
}//void reset()

void checkClipping(){ //checkClipping () function
```

```
  if (clipping) //If clipping is positive, then
  {
    clipping = 0; //Reset value of clipping variable
  }//if (clipping)
}//void checkClipping...

void FREQ_DETECTION() //FREQ_DETECTION Function
{
  Serial.println("Into freq_detection function"); //Print message at Serial Port
  cli(); //disable interrupts
  ADCSRA=0; //Clear ADCSRA register
  ADCSRB=0; //Clear ADCSRB register
  ADMUX=0; //Clear ADMUX register
  ADMUX|=(1<<REFS0); //Set REFS0 bit at ADMUX register. Set reference voltage
  ADMUX|=(1<<ADLAR); //Set ADLAR bit at ADMUX register. Left align the ADC
value to read highest 8 bits from ADCH register only
  ADCSRB|=(1<<MUX5); //Set MUX5 bit at ADCSRB register. Analog port 15 for use
  ADMUX|=(1<<MUX0)|(1<<MUX1)|(1<<MUX2); //Set MUX0, MUX1 & MUX2 bit at
ADMUX register. Analog port 15 for use
  ADCSRA|=(1<<ADPS2)|(1<<ADPS0); //Set ADPS2 & ADPS0 bit at ADCSRA register.
Set ADC clock with 32 prescaler, 16MHz/32=500KHz
  ADCSRA|=(1<<ADATE); //Set ADATE bit at ADCSRA register. Enable Auto trigger
  ADCSRA|=(1<<ADIE); //Set ADIE bit at ADCSRA register. Enable interrupts when
measurement complete
  ADCSRA|=(1<<ADEN); //Set ADEN bit at ADCSRA register. Enable ADC.
  ADCSRA|=(1<<ADSC); //Set ADSC bit at ADCSRA register. Start ADC measurement
  sei(); //Enable interrupts
}//void FREQ_DETECTION

void FREQ_DETECTION_STOP() //FREQ_DETECTION_STOP() function
{
  ADCSRA = 0; //Clear ADCSRA register
  ADCSRB = 0; //Clear ADCSRB register
  ADMUX = 0; //Clear ADMUX register
} //void FREQ_DETECTION_STOP()
```

```
ISR(ADC_vect){ //interrupt routine that runs when a new ADC value is ready
  prevData = newData; //Store previous value
  newData = ADCH; //Read value from ADCH register
  if(prevData < 127 && newData>=127){ //If prevData is smaller than 127 and newData is bigger than or equal with 127, then (increasing and crossing midpoint)
    newSlope = newData - prevData; //Store the deference between newData minus prevData to newSlope variable (calculate slope)
    if(abs(newSlope-maxSlope)<slopeTol){ //If the absolute deference between newData minus maxSlope is smaller than slopeTol, then (if slopes are equal)
      slope[index] = newSlope; //Store newSlope value at slope Array in position that index show
      timer[index] = time; //Store time value at timer Array in position that index show
      time = 0; //Reset time
      if(index == 0){ //if index is equal with 0, then (new max slope just reset)
        noMatch = 0; //Store 0 value at noMatch variable
        index++; // Increase index by 1
      }//if(index ==0);
      else if(abs(timer[0]-timer[index])<timerTol&&abs(slope[0]-newSlope)<slopeTol){ //Else if the absolute deference between timer[0] minus timer [index] is smaller than timerTol and the absolute difference between slope [0] minus new slope is smaller than slopeTol, then
        totalTimer = 0; //Reset totalTime variable
        for (byte k=0;k<index;k++){ //For k is equal with 0, k smaller than index and k increase by 1 every time, then
            totalTimer+=timer[k]; //Store at totalTimer variable the sum of timer array
        }//for
        period = totalTimer; //Store totalTimer value at period variable (set period)
        timer[0] = timer[index]; //Store timer[index] value at timer[0]
        slope[0] = slope[index]; //Store slope[index] value at slope[0]
        index = 1; //Store number 1 at index variable
        noMatch = 0; //Reset noMatch value
      }//else if (abs(timer[0]...
      else { //Else (Crossing midpoint but not match)
        index++; //Increase index by 1
        if (index>9){ //If index is bigger than 9, then
          reset(); //Call reset() function
```

```
      }//if
     }//else
   }//if(abs(newSlope...
   else if (newSlope>maxSlope){ //Else if newSlope variable is bigger than maxSlope,
then (newSlope is much larger than maxSlope)
     maxSlope = newSlope; //Store newSlope value atmaxSlope variable
     time = 0; //Reset time
     noMatch = 0; //Reset noMatch value
     index = 0; //Reset index value
   }//else if(newSlope...
   else{ //Else (slope no steep enough)
     noMatch++; //Increase noMatch by 1
     if(noMatch>9){ //If noMatch value is bigger than 9, then
       reset(); //Call reset() function
     }//if(noMatch>9)...
   }//else
  }//if(prevData...
  if(newData == 0 || newData == 1023){ //If newData is equal to 0 or newData is equal to
1023, then (If clipping)
   clipping = 1; //Store number 1 at clipping variable
   Serial.println("clipping"); //Print a message at serial port
  }//if(newData==0||newData==1023)...
  time++; //Increase time by 1
  ampTimer++; //Increase ampTimer by 1
  if(abs(127-ADCH) > maxAmp){ //If the absolute difference between 127 minus ADCH
is bigger than maxAmp variable, then
   maxAmp=abs(127-ADCH); //Store at maxAmp variable the absolute difference between
127 minus ADCH
  }//if(abs(127...
  if (ampTimer == 1000){ //If ampTimer is equal to 1000, then
   ampTimer = 0; //Reset ampTimer
   checkMaxAmp = maxAmp; //Store maxAmp variable at checkMaxAmp variable
   maxAmp = 0; //Reset maxAmp variable
  }//if(ampTimer...
} //ISR(ADC_vect)
```

```
void FREQ_DETECTION1(){ //FREQ_DETECTION1() function
  checkClipping(); //Call checkClipping function
  if (checkMaxAmp>ampThreshold) //If checkMaxAmp is bigger than ampThreshold, then
  {
     frequency = 38462/float(period); //Calculate frequency
     Serial.print(frequency); //Print message at serial port
     Serial.println("Hz"); //Print message at serial port
    if (frequency > 63.58 && frequency < 67.35) //If frequency value is bigger than
63.58Hz and smaller than 67.35Hz, then
    {
      lcd.setCursor(10,1); //Set cursor at eleventh column and second line
      lcd.print("C"); //Print message at LCD screen
      lcd.setCursor(11,1); //Set cursor at twelfth column and second line
      lcd.print(" "); //Print message at LCD screen
    }
    else if(frequency > 67.36 && frequency < 71.36) //If frequency value is bigger than
67.36Hz and smaller than 71.36Hz, then
    {
      lcd.setCursor(10,1); //Set cursor at eleventh column and second line
      lcd.print("C#"); //Print message at LCD screen
    }
    else if(frequency > 71.37 && frequency < 75.60) //If frequency value is bigger than
71.37Hz and smaller than 75.60Hz, then
    {
      lcd.setCursor(10,1); //Set cursor at eleventh column and second line
      lcd.print("D"); //Print message at LCD screen
      lcd.setCursor(11,1); //Set cursor at twelfth column and second line
      lcd.print(" "); //Print message at LCD screen
    }
    else if(frequency > 75.61 && frequency < 80.09) //If frequency value is bigger than
75.61Hz and smaller than 80.09Hz, then
    {
      lcd.setCursor(10,1); //Set cursor at eleventh column and second line
      lcd.print("D#"); //Print message at LCD screen
    }
```

```
  else if(frequency > 80.10 && frequency < 84.86) //If frequency value is bigger than
80.10Hz and smaller than 84.86Hz, then
  {
   lcd.setCursor(10,1); //Set cursor at eleventh column and second line
   lcd.print(" "); //Print message at LCD screen
   lcd.setCursor(11,1); //Set cursor at twelfth column and second line
   lcd.print("E"); //Print message at LCD screen
  }
  else if(frequency > 84.87 && frequency < 89.90) //If frequency value is bigger than
84.87Hz and smaller than 89.90Hz, then
  {
   lcd.setCursor(10,1); //Set cursor at eleventh column and second line
   lcd.print("F"); //Print message at LCD screen
   lcd.setCursor(11,1); //Set cursor at twelfth column and second line
   lcd.print(" "); //Print message at LCD screen
  }
  else if (frequency > 89.91 && frequency < 95.25) //If frequency value is bigger than
89.91Hz and smaller than 95.25Hz, then
  {
   lcd.setCursor(10,1); //Set cursor at eleventh column and second line
   lcd.print("F#"); //Print message at LCD screen
  }
  else if (frequency > 95.26 && frequency < 100.91) //If frequency value is bigger than
95.26Hz and smaller than 100.91Hz, then
  {
   lcd.setCursor(10,1); //Set cursor at eleventh column and second line
   lcd.print("G"); //Print message at LCD screen
   lcd.setCursor(11,1); //Set cursor at twelfth column and second line
   lcd.print(" "); //Print message at LCD screen
  }
  else if (frequency > 100.92 && frequency < 106.91) //If frequency value is bigger than
100.92Hz and smaller than 106.90Hz, then
  {
   lcd.setCursor(10,1); //Set cursor at eleventh column and second line
   lcd.print("G#"); //Print message at LCD screen
  }
```

```
  else if (frequency > 106.92 && frequency < 113.27) //If frequency value is bigger than
106.92Hz and smaller than 113.27Hz, then
  {
   lcd.setCursor(10,1); //Set cursor at eleventh column and second line
   lcd.print("A"); //Print message at LCD screen
   lcd.setCursor(11,1); //Set cursor at twelfth column and second line
   lcd.print(" "); //Print message at LCD screen
  }
  else if (frequency > 113.28 && frequency < 120) //If frequency value is bigger than
113.28Hz and smaller than 120Hz, then
  {
   lcd.setCursor(10,1); //Set cursor at eleventh column and second line
   lcd.print("A#"); //Print message at LCD screen
  }
  else if (frequency > 120.01 && frequency < 127.14) //If frequency value is bigger than
120.01Hz and smaller than 127.14Hz, then
  {
   lcd.setCursor(10,1); //Set cursor at eleventh column and second line
   lcd.print("B"); //Print message at LCD screen
   lcd.setCursor(10,1); //Set cursor at twelfth column and second line
   lcd.print(" "); //Print message at LCD screen
  }
  else if (frequency > 127.15 && frequency < 134.81) //If frequency value is bigger than
127.15Hz and smaller than 134.81Hz, then
  {
   lcd.setCursor(10,1); //Set cursor at eleventh column and second line
   lcd.print("C"); //Print message at LCD screen
   lcd.setCursor(11,1); //Set cursor at twelfth column and second line
   lcd.print(" "); //Print message at LCD screen
  }
  else if (frequency > 134.82 && frequency < 142.71)//If frequency value is bigger than
134.82Hz and smaller than 142.71Hz, then
  {
   lcd.setCursor(10,1); //Set cursor at eleventh column and second line
   lcd.print("C#"); //Print message at LCD screen
  }
```

```
  else if (frequency > 142.72 && frequency < 151.19) //If frequency value is bigger than
142.72Hz and smaller than 151.19Hz, then
  {
   lcd.setCursor(10,1); //Set cursor at eleventh column and second line
   lcd.print("D"); //Print message at LCD screen
   lcd.setCursor(11,1); //Set cursor at twelfth column and second line
   lcd.print(" "); //Print message at LCD screen
  }
  else if (frequency > 151.20 && frequency < 160.18) //If frequency value is bigger than
151.207Hz and smaller than 160.18Hz, then
  {
   lcd.setCursor(10,1); //Set cursor at eleventh column and second line
   lcd.print("D#"); //Print message at LCD screen
  }
  else if (frequency > 160.19 && frequency < 169.71) //If frequency value is bigger than
160.19Hz and smaller than 169.71Hz, then
  {
   lcd.setCursor(10,1); //Set cursor at eleventh column and second line
   lcd.print(" "); //Print message at LCD screen
   lcd.setCursor(11,1); //Set cursor at twelfth column and second line
   lcd.print("E"); //Print message at LCD screen
  }
  else if (frequency > 169.72 && frequency < 179.80) //If frequency value is bigger than
169.72Hz and smaller than 179.80Hz, then
  {
   lcd.setCursor(10,1); //Set cursor at eleventh column and second line
   lcd.print("F"); //Print message at LCD screen
   lcd.setCursor(11,1); //Set cursor at twelfth column and second line
   lcd.print(" "); //Print message at LCD screen
  }
  else if (frequency > 179.81 && frequency < 190.50) //If frequency value is bigger than
179.81Hz and smaller than 190.50Hz, then
  {
   lcd.setCursor(10,1); //Set cursor at eleventh column and second line
   lcd.print("F#"); //Print message at LCD screen
  }
```

```
  else if (frequency > 190.51 && frequency < 201.82) //If frequency value is bigger than
190.51Hz and smaller than 201.82Hz, then
  {
   lcd.setCursor(10,1); //Set cursor at eleventh column and second line
   lcd.print("G"); //Print message at LCD screen
   lcd.setCursor(11,1); //Set cursor at twelfth column and second line
   lcd.print(" "); //Print message at LCD screen
  }
  else if (frequency > 201.83 && frequency < 213.82) //If frequency value is bigger than
201.83Hz and smaller than 213.82Hz, then
  {
   lcd.setCursor(10,1); //Set cursor at eleventh column and second line
   lcd.print("G#"); //Print message at LCD screen
  }
  else if (frequency > 213.83 && frequency < 226.54) //If frequency value is bigger than
213.83Hz and smaller than 226.54Hz, then
  {
   lcd.setCursor(10,1); //Set cursor at eleventh column and second line
   lcd.print("A"); //Print message at LCD screen
   lcd.setCursor(11,1); //Set cursor at twelfth column and second line
   lcd.print(" "); //Print message at LCD screen
  }
  else if (frequency > 226.55 && frequency < 240.01) //If frequency value is bigger than
226.55Hz and smaller than 240.01Hz, then
  {
   lcd.setCursor(10,1); //Set cursor at eleventh column and second line
   lcd.print("A#"); //Print message at LCD screen
  }
  else if (frequency > 240.02 && frequency < 254.28) //If frequency value is bigger than
240.02Hz and smaller than 254.28Hz, then
  {
   lcd.setCursor(10,1); //Set cursor at eleventh column and second line
   lcd.print("B"); //Print message at LCD screen
   lcd.setCursor(11,1); //Set cursor at twelfth column and second line
   lcd.print(" "); //Print message at LCD screen
  }
```

```
  else if (frequency > 254.29 && frequency < 269.40) //If frequency value is bigger than
254.29Hz and smaller than 269.40Hz, then
  {
   lcd.setCursor(10,1); //Set cursor at eleventh column and second line
   lcd.print("C"); //Print message at LCD screen
   lcd.setCursor(11,1); //Set cursor at twelfth column and second line
   lcd.print(" "); //Print message at LCD screen
  }
  else if (frequency > 269.41 && frequency < 285.42) //If frequency value is bigger than
269.41Hz and smaller than 285.42Hz, then
  {
   lcd.setCursor(10,1); //Set cursor at eleventh column and second line
   lcd.print("C#"); //Print message at LCD screen
  }
  else if (frequency > 285.43 && frequency < 302.39) //If frequency value is bigger than
285.43Hz and smaller than 302.39Hz, then
  {
   lcd.setCursor(10,1); //Set cursor at eleventh column and second line
   lcd.print("D"); //Print message at LCD screen
   lcd.setCursor(11,1); //Set cursor at twelfth column and second line
   lcd.print(" "); //Print message at LCD screen
  }
  else if (frequency > 302.40 && frequency < 320.38) //If frequency value is bigger than
302.40Hz and smaller than 320.38Hz, then
  {
   lcd.setCursor(10,1); //Set cursor at eleventh column and second line
   lcd.print("D#"); //Print message at LCD screen
  }
  else if (frequency > 320.39 && frequency < 339.43) //If frequency value is bigger than
320.39Hz and smaller than 339.43Hz, then
  {
   lcd.setCursor(10,1); //Set cursor at eleventh column and second line
   lcd.print("E"); //Print message at LCD screen
   lcd.setCursor(11,1); //Set cursor at twelfth column and second line
   lcd.print(" "); //Print message at LCD screen
  }
```

```
  else if (frequency > 339.44 && frequency < 359.61) //If frequency value is bigger than
339.44Hz and smaller than 359.61Hz, then
  {
   lcd.setCursor(10,1); //Set cursor at eleventh column and second line
   lcd.print("F"); //Print message at LCD screen
   lcd.setCursor(11,1); //Set cursor at twelfth column and second line
   lcd.print(" "); //Print message at LCD screen
  }
  else if (frequency > 359.62 && frequency < 380.99) //If frequency value is bigger than
359.62Hz and smaller than 380.99Hz, then
  {
   lcd.setCursor(10,1); //Set cursor at eleventh column and second line
   lcd.print("F#"); //Print message at LCD screen
  }
  else if (frequency > 381 && frequency < 403.65) //If frequency value is bigger than
381Hz and smaller than 403.65Hz, then
  {
   lcd.setCursor(10,1); //Set cursor at eleventh column and second line
   lcd.print("G"); //Print message at LCD screen
   lcd.setCursor(11,1); //Set cursor at twelfth column and second line
   lcd.print(" "); //Print message at LCD screen
  }
  else if (frequency > 403.66 && frequency < 427.65) //If frequency value is bigger than
403.66Hz and smaller than 427.65Hz, then
  {
   lcd.setCursor(10,1); //Set cursor at eleventh column and second line
   lcd.print("G#"); //Print message at LCD screen
  }
  else if (frequency > 427.66 && frequency < 453.08) //If frequency value is bigger than
427.66Hz and smaller than 453.08Hz, then
  {
   lcd.setCursor(10,1); //Set cursor at eleventh column and second line
   lcd.print("A"); //Print message at LCD screen
   lcd.setCursor(11,1); //Set cursor at twelfth column and second line
   lcd.print(" "); //Print message at LCD screen
  }
```

```
  else if (frequency > 453.09 && frequency < 480.02) //If frequency value is bigger than
453.09Hz and smaller than 480.02Hz, then
  {
   lcd.setCursor(10,1); //Set cursor at eleventh column and second line
   lcd.print("A#"); //Print message at LCD screen
  }
  else if (frequency > 480.03 && frequency < 508.56) //If frequency value is bigger than
480.03Hz and smaller than 508.56Hz, then
  {
   lcd.setCursor(10,1); //Set cursor at eleventh column and second line
   lcd.print("B"); //Print message at LCD screen
   lcd.setCursor(11,1); //Set cursor at twelfth column and second line
   lcd.print(" "); //Print message at LCD screen
  }
  else if (frequency > 508.57 && frequency < 538.81) //If frequency value is bigger than
508.57Hz and smaller than 538.81Hz, then
  {
   lcd.setCursor(10,1); //Set cursor at eleventh column and second line
   lcd.print("C"); //Print message at LCD screen
   lcd.setCursor(11,1); //Set cursor at twelfth column and second line
   lcd.print(" "); //Print message at LCD screen
  }
  else //Else
  {
   frequency = 0.00; //Reset frequency value
   lcd.setCursor(10,1); //Set cursor at eleventh column and second line
   lcd.print(" "); //Print message at LCD screen
   lcd.setCursor(11,1); //Set cursor at twelfth column and second line
   lcd.print(" "); //Print message at LCD screen
  }
  Serial.println(note); //Print note value at Serial Port
 }//if (checkMaxAmp...
}//void FREQ_DETECTION

void Clockwise_eStep_Motor () { // Function for clockwise movement of eStep_Motor
 digitalWrite(eStepper_Pin1,LOW); //Set eStepper_Pin1 LOW = 0
```

```
  digitalWrite(eStepper_Pin2,LOW); //Set eStepper_Pin2 LOW = 0
  digitalWrite(eStepper_Pin3,HIGH); //Set eStepper_Pin3 HIGH = 1
  digitalWrite(eStepper_Pin4, HIGH); //Set eStepper_Pin4 HIGH = 1
  delay(2); //Time delay 2 msec

  digitalWrite(eStepper_Pin1,HIGH); //Set eStepper_Pin1 HIGH = 1
  digitalWrite(eStepper_Pin2,LOW); //Set eStepper_Pin2 LOW = 0
  digitalWrite(eStepper_Pin3,LOW); //Set eStepper_Pin3 LOW = 0
  digitalWrite(eStepper_Pin4, HIGH); //Set eStepper_Pin4 HIGH = 1
  delay(2); //Time delay 2 msec

  digitalWrite(eStepper_Pin1,HIGH); //Set eStepper_Pin1 HIGH = 1
  digitalWrite(eStepper_Pin2,HIGH); //Set eStepper_Pin2 HIGH = 1
  digitalWrite(eStepper_Pin3,LOW); //Set eStepper_Pin3 LOW = 0
  digitalWrite(eStepper_Pin4, LOW); //Set eStepper_Pin4 LOW = 0
  delay(2); //Time delay 2 msec

  digitalWrite(eStepper_Pin1,LOW); //Set eStepper_Pin1 LOW = 0
  digitalWrite(eStepper_Pin2,HIGH); //Set eStepper_Pin2 HIGH = 1
  digitalWrite(eStepper_Pin3,HIGH); //Set eStepper_Pin3 HIGH = 1
  digitalWrite(eStepper_Pin4, LOW); //Set eStepper_Pin4 LOW = 0
  delay(2); //Time delay 2 msec
}
void Counterclockwise_eStep_Motor () { // Function for Counterclockwise movement of
eStep_Motor
  digitalWrite(eStepper_Pin1, HIGH); //Set eStepper_Pin1 HIGH = 1
  digitalWrite(eStepper_Pin2, HIGH); //Set eStepper_Pin2 HIGH = 1
  digitalWrite(eStepper_Pin3, LOW); //Set eStepper_Pin1 LOW = 0
  digitalWrite(eStepper_Pin4, LOW); //Set eStepper_Pin1 LOW = 0
  delay(2); //Time delay 2 msec

  digitalWrite(eStepper_Pin1, HIGH); //Set eStepper_Pin1 HIGH = 1
  digitalWrite(eStepper_Pin2, LOW); //Set eStepper_Pin2 LOW = 0
  digitalWrite(eStepper_Pin3, LOW); //Set eStepper_Pin1 LOW = 0
  digitalWrite(eStepper_Pin4, HIGH); //Set eStepper_Pin1 HIGH = 1
  delay(2); //Time delay 2 msec
```

```
  digitalWrite(eStepper_Pin1, LOW); //Set eStepper_Pin1 LOW = 0
  digitalWrite(eStepper_Pin2, LOW); //Set eStepper_Pin2 LOW = 0
  digitalWrite(eStepper_Pin3, HIGH); //Set eStepper_Pin1 HIGH = 1
  digitalWrite(eStepper_Pin4, HIGH); //Set eStepper_Pin1 HIGH = 1
  delay(2); //Time delay 2 msec

  digitalWrite(eStepper_Pin1, LOW); //Set eStepper_Pin1 LOW = 0
  digitalWrite(eStepper_Pin2, HIGH); //Set eStepper_Pin2 HIGH = 1
  digitalWrite(eStepper_Pin3, HIGH); //Set eStepper_Pin1 HIGH = 1
  digitalWrite(eStepper_Pin4, LOW); //Set eStepper_Pin1 LOW = 0
  delay(2); //Time delay 2 msec
}

void Clockwise_BStep_Motor () { // Function for clockwise movement of BStep_Motor
  digitalWrite(BStepper_Pin1,LOW); //Set BStepper_Pin1 LOW = 0
  digitalWrite(BStepper_Pin2,LOW); //Set BStepper_Pin2 LOW = 0
  digitalWrite(BStepper_Pin3,HIGH); //Set BStepper_Pin3 HIGH = 1
  digitalWrite(BStepper_Pin4, HIGH); //Set BStepper_Pin4 HIGH = 1
  delay(2); //Time delay 2 msec

  digitalWrite(BStepper_Pin1,HIGH); //Set BStepper_Pin1 HIGH = 1
  digitalWrite(BStepper_Pin2,LOW); //Set BStepper_Pin2 LOW = 0
  digitalWrite(BStepper_Pin3,LOW); //Set BStepper_Pin3 LOW = 0
  digitalWrite(BStepper_Pin4, HIGH); //Set BStepper_Pin4 HIGH = 1
  delay(2); //Time delay 2 msec

  digitalWrite(BStepper_Pin1,HIGH); //Set BStepper_Pin1 HIGH = 1
  digitalWrite(BStepper_Pin2,HIGH); //Set BStepper_Pin2 HIGH = 1
  digitalWrite(BStepper_Pin3,LOW); //Set BStepper_Pin3 LOW = 0
  digitalWrite(BStepper_Pin4, LOW); //Set BStepper_Pin4 LOW = 0
  delay(2); //Time delay 2 msec

  digitalWrite(BStepper_Pin1,LOW); //Set BStepper_Pin1 LOW = 0
  digitalWrite(BStepper_Pin2,HIGH); //Set BStepper_Pin2 HIGH = 1
  digitalWrite(BStepper_Pin3,HIGH); //Set BStepper_Pin3 HIGH = 1
```

```
  digitalWrite(BStepper_Pin4, LOW); //Set BStepper_Pin4 LOW = 0
  delay(2); //Time delay 2 msec
}

void Counterclockwise_BStep_Motor () { // Function for Counterclockwise movement of
BStep_Motor
  digitalWrite(BStepper_Pin1, HIGH); //Set BStepper_Pin1 HIGH = 1
  digitalWrite(BStepper_Pin2, HIGH); //Set BStepper_Pin2 HIGH = 1
  digitalWrite(BStepper_Pin3, LOW); //Set BStepper_Pin1 LOW = 0
  digitalWrite(BStepper_Pin4, LOW); //Set BStepper_Pin1 LOW = 0
  delay(2); //Time delay 2 msec

  digitalWrite(BStepper_Pin1, HIGH); //Set BStepper_Pin1 HIGH = 1
  digitalWrite(BStepper_Pin2, LOW); //Set BStepper_Pin2 LOW = 0
  digitalWrite(BStepper_Pin3, LOW); //Set BStepper_Pin1 LOW = 0
  digitalWrite(BStepper_Pin4, HIGH); //Set BStepper_Pin1 HIGH = 1
  delay(2); //Time delay 2 msec

  digitalWrite(BStepper_Pin1, LOW); //Set BStepper_Pin1 LOW = 0
  digitalWrite(BStepper_Pin2, LOW); //Set BStepper_Pin2 LOW = 0
  digitalWrite(BStepper_Pin3, HIGH); //Set BStepper_Pin1 HIGH = 1
  digitalWrite(BStepper_Pin4, HIGH); //Set BStepper_Pin1 HIGH = 1
  delay(2); //Time delay 2 msec

  digitalWrite(BStepper_Pin1, LOW); //Set BStepper_Pin1 LOW = 0
  digitalWrite(BStepper_Pin2, HIGH); //Set BStepper_Pin2 HIGH = 1
  digitalWrite(BStepper_Pin3, HIGH); //Set BStepper_Pin1 HIGH = 1
  digitalWrite(BStepper_Pin4, LOW); //Set BStepper_Pin1 LOW = 0
  delay(2); //Time delay 2 msec
}

void Clockwise_GStep_Motor () { // Function for clockwise movement of GStep_Motor
  digitalWrite(GStepper_Pin1,LOW); //Set GStepper_Pin1 LOW = 0
  digitalWrite(GStepper_Pin2,LOW); //Set GStepper_Pin2 LOW = 0
  digitalWrite(GStepper_Pin3,HIGH); //Set GStepper_Pin3 HIGH = 1
  digitalWrite(GStepper_Pin4, HIGH); //Set GStepper_Pin4 HIGH = 1
```

```
  delay(2); //Time delay 2 msec

  digitalWrite(GStepper_Pin1,HIGH); //Set GStepper_Pin1 HIGH = 1
  digitalWrite(GStepper_Pin2,LOW); //Set GStepper_Pin2 LOW = 0
  digitalWrite(GStepper_Pin3,LOW); //Set GStepper_Pin3 LOW = 0
  digitalWrite(GStepper_Pin4, HIGH); //Set GStepper_Pin4 HIGH = 1
  delay(2); //Time delay 2 msec

  digitalWrite(GStepper_Pin1,HIGH); //Set GStepper_Pin1 HIGH = 1
  digitalWrite(GStepper_Pin2,HIGH); //Set GStepper_Pin2 HIGH = 1
  digitalWrite(GStepper_Pin3,LOW); //Set GStepper_Pin3 LOW = 0
  digitalWrite(GStepper_Pin4, LOW); //Set GStepper_Pin4 LOW = 0
  delay(2); //Time delay 2 msec

  digitalWrite(GStepper_Pin1,LOW); //Set GStepper_Pin1 LOW = 0
  digitalWrite(GStepper_Pin2,HIGH); //Set GStepper_Pin2 HIGH = 1
  digitalWrite(GStepper_Pin3,HIGH); //Set GStepper_Pin3 HIGH = 1
  digitalWrite(GStepper_Pin4, LOW); //Set GStepper_Pin4 LOW = 0
  delay(2); //Time delay 2 msec
}

void Counterclockwise_GStep_Motor() { // Function for Counterclockwise movement of
GStep_Motor
  digitalWrite(GStepper_Pin1, HIGH); //Set GStepper_Pin1 HIGH = 1
  digitalWrite(GStepper_Pin2, HIGH); //Set GStepper_Pin2 HIGH = 1
  digitalWrite(GStepper_Pin3, LOW); //Set GStepper_Pin1 LOW = 0
  digitalWrite(GStepper_Pin4, LOW); //Set GStepper_Pin1 LOW = 0
  delay(2); //Time delay 2 msec

  digitalWrite(GStepper_Pin1, HIGH); //Set GStepper_Pin1 HIGH = 1
  digitalWrite(GStepper_Pin2, LOW); //Set GStepper_Pin2 LOW = 0
  digitalWrite(GStepper_Pin3, LOW); //Set GStepper_Pin1 LOW = 0
  digitalWrite(GStepper_Pin4, HIGH); //Set GStepper_Pin1 HIGH = 1
  delay(2); //Time delay 2 msec

  digitalWrite(GStepper_Pin1, LOW); //Set GStepper_Pin1 LOW = 0
```

```
  digitalWrite(GStepper_Pin2, LOW); //Set GStepper_Pin2 LOW = 0
  digitalWrite(GStepper_Pin3, HIGH); //Set GStepper_Pin1 HIGH = 1
  digitalWrite(GStepper_Pin4, HIGH); //Set GStepper_Pin1 HIGH = 1
  delay(2); //Time delay 2 msec

  digitalWrite(GStepper_Pin1, LOW); //Set GStepper_Pin1 LOW = 0
  digitalWrite(GStepper_Pin2, HIGH); //Set GStepper_Pin2 HIGH = 1
  digitalWrite(GStepper_Pin3, HIGH); //Set GStepper_Pin1 HIGH = 1
  digitalWrite(GStepper_Pin4, LOW); //Set GStepper_Pin1 LOW = 0
  delay(2); //Time delay 2 msec
}

void Counterclockwise_DStep_Motor () { // Function for clockwise movement of
DStep_Motor
  digitalWrite(DStepper_Pin1,LOW); //Set DStepper_Pin1 LOW = 0
  digitalWrite(DStepper_Pin2,LOW); //Set DStepper_Pin2 LOW = 0
  digitalWrite(DStepper_Pin3,HIGH); //Set DStepper_Pin3 HIGH = 1
  digitalWrite(DStepper_Pin4, HIGH); //Set DStepper_Pin4 HIGH = 1
  delay(2); //Time delay 2 msec

  digitalWrite(DStepper_Pin1,HIGH); //Set DStepper_Pin1 HIGH = 1
  digitalWrite(DStepper_Pin2,LOW); //Set DStepper_Pin2 LOW = 0
  digitalWrite(DStepper_Pin3,LOW); //Set DStepper_Pin3 LOW = 0
  digitalWrite(DStepper_Pin4, HIGH); //Set DStepper_Pin4 HIGH = 1
  delay(2); //Time delay 2 msec

  digitalWrite(DStepper_Pin1,HIGH); //Set DStepper_Pin1 HIGH = 1
  digitalWrite(DStepper_Pin2,HIGH); //Set DStepper_Pin2 HIGH = 1
  digitalWrite(DStepper_Pin3,LOW); //Set DStepper_Pin3 LOW = 0
  digitalWrite(DStepper_Pin4, LOW); //Set DStepper_Pin4 LOW = 0
  delay(2); //Time delay 2 msec

  digitalWrite(DStepper_Pin1,LOW); //Set DStepper_Pin1 LOW = 0
  digitalWrite(DStepper_Pin2,HIGH); //Set DStepper_Pin2 HIGH = 1
  digitalWrite(DStepper_Pin3,HIGH); //Set DStepper_Pin3 HIGH = 1
  digitalWrite(DStepper_Pin4, LOW); //Set DStepper_Pin4 LOW = 0
```

```
  delay(2); //Time delay 2 msec
}

void Clockwise_DStep_Motor () { // Function for Counterclockwise movement of
DStep_Motor
  digitalWrite(DStepper_Pin1, HIGH); //Set DStepper_Pin1 HIGH = 1
  digitalWrite(DStepper_Pin2, HIGH); //Set DStepper_Pin2 HIGH = 1
  digitalWrite(DStepper_Pin3, LOW); //Set DStepper_Pin1 LOW = 0
  digitalWrite(DStepper_Pin4, LOW); //Set DStepper_Pin1 LOW = 0
  delay(2); //Time delay 2 msec

  digitalWrite(DStepper_Pin1, HIGH); //Set DStepper_Pin1 HIGH = 1
  digitalWrite(DStepper_Pin2, LOW); //Set DStepper_Pin2 LOW = 0
  digitalWrite(DStepper_Pin3, LOW); //Set DStepper_Pin1 LOW = 0
  digitalWrite(DStepper_Pin4, HIGH); //Set DStepper_Pin1 HIGH = 1
  delay(2); //Time delay 2 msec

  digitalWrite(DStepper_Pin1, LOW); //Set DStepper_Pin1 LOW = 0
  digitalWrite(DStepper_Pin2, LOW); //Set DStepper_Pin2 LOW = 0
  digitalWrite(DStepper_Pin3, HIGH); //Set DStepper_Pin1 HIGH = 1
  digitalWrite(DStepper_Pin4, HIGH); //Set DStepper_Pin1 HIGH = 1
  delay(2); //Time delay 2 msec

  digitalWrite(DStepper_Pin1, LOW); //Set DStepper_Pin1 LOW = 0
  digitalWrite(DStepper_Pin2, HIGH); //Set DStepper_Pin2 HIGH = 1
  digitalWrite(DStepper_Pin3, HIGH); //Set DStepper_Pin1 HIGH = 1
  digitalWrite(DStepper_Pin4, LOW); //Set DStepper_Pin1 LOW = 0
  delay(2); //Time delay 2 msec
}

void Counterclockwise_AStep_Motor () { // Function for clockwise movement of
AStep_Motor
  digitalWrite(AStepper_Pin1,LOW); //Set AStepper_Pin1 LOW = 0
  digitalWrite(AStepper_Pin2,LOW); //Set AStepper_Pin2 LOW = 0
  digitalWrite(AStepper_Pin3,HIGH); //Set AStepper_Pin3 HIGH = 1
  digitalWrite(AStepper_Pin4, HIGH); //Set AStepper_Pin4 HIGH = 1
```

```
  delay(2); //Time delay 2 msec

  digitalWrite(AStepper_Pin1,HIGH); //Set AStepper_Pin1 HIGH = 1
  digitalWrite(AStepper_Pin2,LOW); //Set AStepper_Pin2 LOW = 0
  digitalWrite(AStepper_Pin3,LOW); //Set AStepper_Pin3 LOW = 0
  digitalWrite(AStepper_Pin4, HIGH); //Set AStepper_Pin4 HIGH = 1
  delay(2); //Time delay 2 msec

  digitalWrite(AStepper_Pin1,HIGH); //Set AStepper_Pin1 HIGH = 1
  digitalWrite(AStepper_Pin2,HIGH); //Set AStepper_Pin2 HIGH = 1
  digitalWrite(AStepper_Pin3,LOW); //Set AStepper_Pin3 LOW = 0
  digitalWrite(AStepper_Pin4, LOW); //Set AStepper_Pin4 LOW = 0
  delay(2); //Time delay 2 msec

  digitalWrite(AStepper_Pin1,LOW); //Set AStepper_Pin1 LOW = 0
  digitalWrite(AStepper_Pin2,HIGH); //Set AStepper_Pin2 HIGH = 1
  digitalWrite(AStepper_Pin3,HIGH); //Set AStepper_Pin3 HIGH = 1
  digitalWrite(AStepper_Pin4, LOW); //Set AStepper_Pin4 LOW = 0
  delay(2); //Time delay 2 msec
}

void Clockwise_AStep_Motor () { // Function for Counterclockwise movement of
AStep_Motor
  digitalWrite(AStepper_Pin1, HIGH); //Set AStepper_Pin1 HIGH = 1
  digitalWrite(AStepper_Pin2, HIGH); //Set AStepper_Pin2 HIGH = 1
  digitalWrite(AStepper_Pin3, LOW); //Set AStepper_Pin1 LOW = 0
  digitalWrite(AStepper_Pin4, LOW); //Set AStepper_Pin1 LOW = 0
  delay(2); //Time delay 2 msec

  digitalWrite(AStepper_Pin1, HIGH); //Set AStepper_Pin1 HIGH = 1
  digitalWrite(AStepper_Pin2, LOW); //Set AStepper_Pin2 LOW = 0
  digitalWrite(AStepper_Pin3, LOW); //Set AStepper_Pin1 LOW = 0
  digitalWrite(AStepper_Pin4, HIGH); //Set AStepper_Pin1 HIGH = 1
  delay(2); //Time delay 2 msec

  digitalWrite(AStepper_Pin1, LOW); //Set AStepper_Pin1 LOW = 0
```

```
  digitalWrite(AStepper_Pin2, LOW); //Set AStepper_Pin2 LOW = 0
  digitalWrite(AStepper_Pin3, HIGH); //Set AStepper_Pin1 HIGH = 1
  digitalWrite(AStepper_Pin4, HIGH); //Set AStepper_Pin1 HIGH = 1
  delay(2); //Time delay 2 msec

  digitalWrite(AStepper_Pin1, LOW); //Set AStepper_Pin1 LOW = 0
  digitalWrite(AStepper_Pin2, HIGH); //Set AStepper_Pin2 HIGH = 1
  digitalWrite(AStepper_Pin3, HIGH); //Set AStepper_Pin1 HIGH = 1
  digitalWrite(AStepper_Pin4, LOW); //Set AStepper_Pin1 LOW = 0
  delay(2); //Time delay 2 msec
}

void Counterclockwise_EStep_Motor () { // Function for clockwise movement of
EStep_Motor
  digitalWrite(EStepper_Pin1,LOW); //Set EStepper_Pin1 LOW = 0
  digitalWrite(EStepper_Pin2,LOW); //Set EStepper_Pin2 LOW = 0
  digitalWrite(EStepper_Pin3,HIGH); //Set EStepper_Pin3 HIGH = 1
  digitalWrite(EStepper_Pin4, HIGH); //Set EStepper_Pin4 HIGH = 1
  delay(2); //Time delay 2 msec

  digitalWrite(EStepper_Pin1,HIGH); //Set EStepper_Pin1 HIGH = 1
  digitalWrite(EStepper_Pin2,LOW); //Set EStepper_Pin2 LOW = 0
  digitalWrite(EStepper_Pin3,LOW); //Set EStepper_Pin3 LOW = 0
  digitalWrite(EStepper_Pin4, HIGH); //Set EStepper_Pin4 HIGH = 1
  delay(2); //Time delay 2 msec

  digitalWrite(EStepper_Pin1,HIGH); //Set EStepper_Pin1 HIGH = 1
  digitalWrite(EStepper_Pin2,HIGH); //Set EStepper_Pin2 HIGH = 1
  digitalWrite(EStepper_Pin3,LOW); //Set EStepper_Pin3 LOW = 0
  digitalWrite(EStepper_Pin4, LOW); //Set EStepper_Pin4 LOW = 0
  delay(2); //Time delay 2 msec

  digitalWrite(EStepper_Pin1,LOW); //Set EStepper_Pin1 LOW = 0
  digitalWrite(EStepper_Pin2,HIGH); //Set EStepper_Pin2 HIGH = 1
  digitalWrite(EStepper_Pin3,HIGH); //Set EStepper_Pin3 HIGH = 1
  digitalWrite(EStepper_Pin4, LOW); //Set EStepper_Pin4 LOW = 0
```

```
  delay(2); //Time delay 2 msec
}

void Clockwise_EStep_Motor (){ // Function for Counterclockwise movement of
EStep_Motor
  digitalWrite(EStepper_Pin1, HIGH); //Set EStepper_Pin1 HIGH = 1
  digitalWrite(EStepper_Pin2, HIGH); //Set EStepper_Pin2 HIGH = 1
  digitalWrite(EStepper_Pin3, LOW); //Set EStepper_Pin1 LOW = 0
  digitalWrite(EStepper_Pin4, LOW); //Set EStepper_Pin1 LOW = 0
  delay(2); //Time delay 2 msec

  digitalWrite(EStepper_Pin1, HIGH); //Set EStepper_Pin1 HIGH = 1
  digitalWrite(EStepper_Pin2, LOW); //Set EStepper_Pin2 LOW = 0
  digitalWrite(EStepper_Pin3, LOW); //Set EStepper_Pin1 LOW = 0
  digitalWrite(EStepper_Pin4, HIGH); //Set EStepper_Pin1 HIGH = 1
  delay(2); //Time delay 2 msec

  digitalWrite(EStepper_Pin1, LOW); //Set EStepper_Pin1 LOW = 0
  digitalWrite(EStepper_Pin2, LOW); //Set EStepper_Pin2 LOW = 0
  digitalWrite(EStepper_Pin3, HIGH); //Set EStepper_Pin1 HIGH = 1
  digitalWrite(EStepper_Pin4, HIGH); //Set EStepper_Pin1 HIGH = 1
  delay(2); //Time delay 2 msec

  digitalWrite(EStepper_Pin1, LOW); //Set EStepper_Pin1 LOW = 0
  digitalWrite(EStepper_Pin2, HIGH); //Set EStepper_Pin2 HIGH = 1
  digitalWrite(EStepper_Pin3, HIGH); //Set EStepper_Pin1 HIGH = 1
  digitalWrite(EStepper_Pin4, LOW); //Set EStepper_Pin1 LOW = 0
  delay(2); //Time delay 2 msec
}

void eStep_Motor_Stop () { // Function to deactivate movement of eStep_Motor
  digitalWrite(eStepper_Pin1,LOW); //Set eStepper_Pin1 LOW = 0
  digitalWrite(eStepper_Pin2,LOW); //Set eStepper_Pin2 LOW = 0
  digitalWrite(eStepper_Pin3,LOW); //Set eStepper_Pin3 LOW = 0
  digitalWrite(eStepper_Pin4, LOW); //Set eStepper_Pin4 LOW = 0
  delay(2); //Time delay 2 msec
```

```
}

void BStep_Motor_Stop () { // Function to deactivate movement of BStep_Motor
  digitalWrite(BStepper_Pin1,LOW); //Set BStepper_Pin1 LOW = 0
  digitalWrite(BStepper_Pin2,LOW); //Set BStepper_Pin2 LOW = 0
  digitalWrite(BStepper_Pin3,LOW); //Set BStepper_Pin3 LOW = 0
  digitalWrite(BStepper_Pin4, LOW); //Set BStepper_Pin4 LOW = 0
  delay(2); //Time delay 2 msec
}

void GStep_Motor_Stop () { // Function to deactivate movement of GStep_Motor
  digitalWrite(GStepper_Pin1, LOW); //Set GStepper_Pin1 LOW = 0
  digitalWrite(GStepper_Pin2, LOW); //Set GStepper_Pin2 LOW = 0
  digitalWrite(GStepper_Pin3, LOW); //Set GStepper_Pin1 LOW = 0
  digitalWrite(GStepper_Pin4, LOW); //Set GStepper_Pin1 LOW = 0
  delay(2); //Time delay 2 msec
}

void DStep_Motor_Stop () { // Function to deactivate movement of DStep_Motor
  digitalWrite(DStepper_Pin1, LOW); //Set DStepper_Pin1 LOW = 0
  digitalWrite(DStepper_Pin2, LOW); //Set DStepper_Pin2 LOW = 0
  digitalWrite(DStepper_Pin3, LOW); //Set DStepper_Pin1 LOW = 0
  digitalWrite(DStepper_Pin4, LOW); //Set DStepper_Pin1 LOW = 0
  delay(2); //Time delay 2 msec
}

void AStep_Motor_Stop () { // Function to deactivate movement of AStep_Motor
  digitalWrite(AStepper_Pin1, LOW); //Set AStepper_Pin1 LOW = 0
  digitalWrite(AStepper_Pin2, LOW); //Set AStepper_Pin2 LOW = 0
  digitalWrite(AStepper_Pin3, LOW); //Set AStepper_Pin1 LOW = 0
  digitalWrite(AStepper_Pin4, LOW); //Set AStepper_Pin1 LOW = 0
  delay(2); //Time delay 2 msec
}

void EStep_Motor_Stop () {// Function to deactivate movement of EStep_Motor
  digitalWrite(EStepper_Pin1, LOW); //Set EStepper_Pin1 LOW = 0
```

```
  digitalWrite(EStepper_Pin2, LOW); //Set EStepper_Pin2 LOW = 0
  digitalWrite(EStepper_Pin3, LOW); //Set EStepper_Pin1 LOW = 0
  digitalWrite(EStepper_Pin4, LOW); //Set EStepper_Pin1 LOW = 0
  delay(2);
}

void Manual_Button_Read() //Function for button read at manual control
{
  ADCSRA = 0; //Clear ADCSRA register
  ADCSRB = 0; //Clear ADCSRB register
  ADMUX = 0; //Clear ADMUX register
  ADMUX |= (1<<REFS0)|(0<<REFS1); //Set bit REFS0 and clear bit REFS1 at ADMUX
register
  ADCSRB |= (1<<ACME); //Set bit ACME at ADCSRB register
  ADCSRA |=(1<<ADEN); //Set bit ADEN at ADCSRA register
  ADCSRA |= (1<<ADSC); //Set bit ADSC at ADCSRA register
  x = ADC; //Store ADC value to x variable
  delay(100); // Delay 100 msec
  Serial.println(x); //Print x value at Serial port

  if (x<60) //If right button has been pressed
  {
    manual_menu+=2; //Add 2 at current variable manual_menu and store the result at the
same variable
    if (manual_menu > 7) // If the value of the manual_menu variable is bigger than 7
    {
      manual_menu = 7; //Store number 7 at manual_menu variable
    }
    delay(50); //Delay 50 msec
  }
  else if(x<200) //If up button has been pressed
  {
    manual_menu--; //Reduce manual_menu variable by 1
    if (manual_menu == 0) //If manual_menu variable is equal to 0
    {
      manual_menu = 1; //Store number 1 at manual_menu variable
```

```
    }
    delay(50); //Delay 50 msec
  }
  else if(x<400) //If down button has been pressed
  {
    manual_menu++; //Increase manual_menu variable by 1
    if (manual_menu > 7) //If manual_menu variable is bigger than 7
    {
      manual_menu = 7; //Store number 7 at manual_menu variable
    }
    delay(50); //Delay 50 msec
  }
  else if(x<600) //If left button has been pressed
  {
    manual_menu-=2; //Reduce manual_menu variable by 2
    if (manual_menu < 1) //If manual_menu variable is less than 1
    {
      manual_menu = 1; //Store number 1 at manual_menu variable
    }
    delay(50); //Delay 50 msec
  }
  else if(x<800) //If select button has been pressed
  {
    Manual_Select = 1; //Set Manual_Select
  }
} //Manual_Button_Read()

void Manual_Step_Motor_Control() //Function for Manual keys control
{
  ADCSRA = 0; //Clear ADCSRA register
  ADCSRB = 0; //Clear ADCSRB register
  ADMUX = 0; //Clear ADMUX register
  ADMUX |= (1<<REFS0)|(0<<REFS1); //Set bit REFS0 and clear bit REFS1 at ADMUX
register
  ADCSRB |= (1<<ACME); //Set bit ACME at ADCSRB register
  ADCSRA |=(1<<ADEN); //Set bit ADEN at ADCSRA register
```

```
 ADCSRA |= (1<<ADSC); //Set bit ADSC at ADCSRA register
 x = ADC; //Store ADC value at x variable
 delay(100); //Delay 100 msec
 Serial.println(x); //Print x value at Serial port
 if (x<60) //If right button has been pressed
 {
  Turn_Right = 1; //Set Turn_Right Variable
 }
 else if(x<200) //If up button has been pressed
 {
  lcd.clear(); //Clear LCD screen
  lcd.setCursor(0,0); //Set cursor at 1st column and 1st line
  lcd.print(" Press Right or "); //Print message at LCD screen
  lcd.setCursor(0,1); //Set cursor at 1st column and 2nd line
  lcd.print("  Left Buttons  "); //Print message at LCD screen
  delay(3000); //Delay 3 sec
  lcd.clear(); //Clear LCD screen
  lcd.setCursor(0,0); //Set cursor at 1st column and 1st line
  lcd.print("  Press SELECT  "); //Print message at LCD screen
  lcd.setCursor(0,1); //Set cursor at 1st column and 2nd line
  lcd.print("   to go back   "); //Print message at LCD screen
 }
 else if(x<400) //If down button has been pressed
 {
  lcd.clear(); //Clear LCD  screen
  lcd.setCursor(0,0); //Set cursor at 1st column and 1st line
  lcd.print(" Press Right or "); //Print message at LCD screen
  lcd.setCursor(0,1); //Set cursor at 1st column and 1st line
  lcd.print("  Left Buttons  "); //Print message at LCD screen
  delay(3000); //Delay 3 sec
  lcd.clear(); //Clear LCD screen
  lcd.setCursor(0,0); //Set cursor at 1st column and 1st line
  lcd.print("  Press SELECT  "); //Print message at LCD screen
  lcd.setCursor(0,1); //Set cursor at 1st column and 1st line
  lcd.print("   to go back   "); //Print message at LCD screen
 }
```

```
  else if(x<600) //If left button has been pressed
  {
    Turn_Left = 1; //Set Turn_Left Variable
  }
  else if(x<800) //If select button has been pressed
  {
    Finish_Control = 1; //Set Finish_Control Variable to return at previous Menu
    eStep_Motor_Stop (); //Function to Deactivate e string Step Motor
    BStep_Motor_Stop (); //Function to Deactivate B string Step Motor
    GStep_Motor_Stop (); //Function to Deactivate G string Step Motor
    DStep_Motor_Stop (); //Function to Deactivate D string Step Motor
    AStep_Motor_Stop (); //Function to Deactivate A string Step Motor
    EStep_Motor_Stop (); //Function to Deactivate E string Step Motor
    lcd.clear(); //Clear LCD screen
  }
} //void Manual_Step_Motor_Control()
```