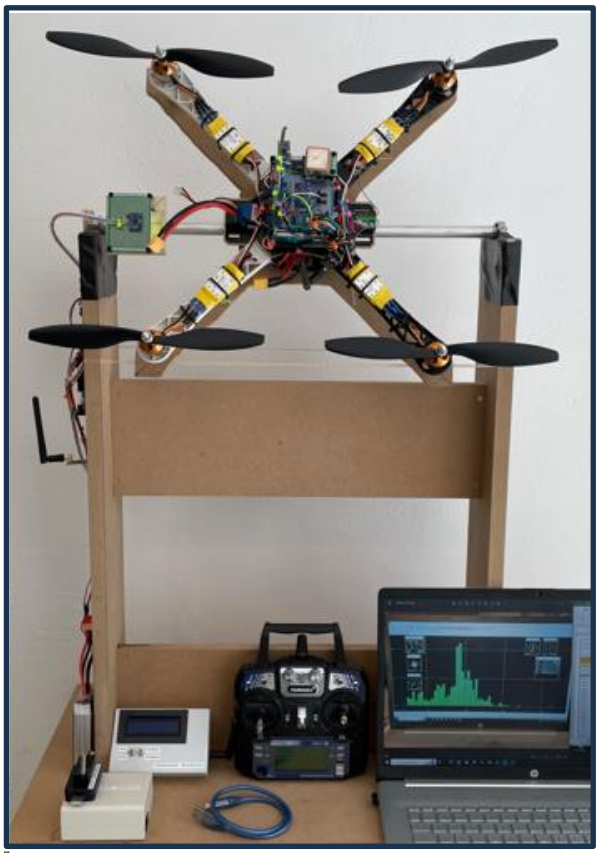
ΠΑΝΕΠΙΣΤΗΜΙΟ ΔΥΤΙΚΗΣ ΑΤΤΙΚΗΣ

ΣΧΟΛΗ ΜΗΧΑΝΙΚΩΝ

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΒΙΟΜΗΧΑΝΙΚΗΣ ΣΧΕΔΙΑΣΗΣ ΚΑΙ ΠΑΡΑΓΩΓΗΣ

ΘΕΜΑ ΔΙΠΛΩΜΑΤΙΚΗΣ ΕΡΓΑΣΙΑΣ

«Ανάπτυξη εκπαιδευτικού τετρακόπτερου με δυνατότητα τηλεμετρίας σε πραγματικό χρόνο»



ΟΝΟΜΑ ΦΟΙΤΗΤΗ: ΓΑΛΑΝΗΣ ΠΑΥΛΟΣ

ΕΠΙΒΛΕΠΩΝ ΚΑΘΗΓΗΤΗΣ: ΑΒΡΑΑΜ ΧΑΤΖΟΠΟΥΛΟΣ

ΑΙΓΑΛΕΩ, ΦΕΒΡΟΥΑΡΙΟΣ 2023

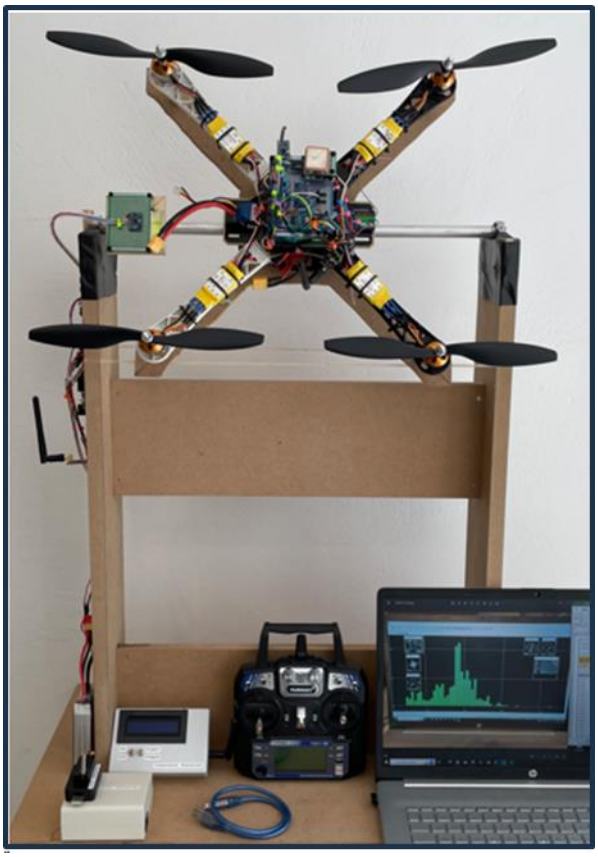
UNIVERSITY OF WEST ATTICA

SCHOOL OF ENGINNERING

DEPARTMENT OF INDUSTRIAL DESIGN
AND PRODUCTION ENGINNERING

DIPLOMA THESIS

«Development of an educational quadcopter with telemetry capability in real time»



STUDENT NAME: GALANIS PAUL

SUPERVISOR: AVRAAM CHATZOPOULOS

EGALEO, FEBRUARY 2023

Η παρούσα διπλωματική εργασία εγκρίθηκε ομόφωνα από την τριμελή εξεταστική επιτροπή, η οποία ορίστηκε από την Γ.Σ. του Τμήματος Μηχανικών Βιομηχανικής Σχεδίασης και Παραγωγής του Πανεπιστημίου Δυτικής Αττικής, σύμφωνα με το νόμο και τον εγκεκριμένο Οδηγό Σπουδών του τμήματος.

Επιβλέπων: Χατζόπουλος Αβραάμ
Λέκτορας

Επιτροπή Αξιολόγησης:

AVRAAM CHATZOPOULOS
Digitally signed by AVRAAM CHATZOPOULOS
Date: 2024.04.12 17:06:23 +03'00'

.....

Χατζόπουλος Αβραάμ
Λέκτορας

Digitally signed by Michail Papoutsidakis
Date: 2024.04.12 11:06:25 +03'00'

.....

Evangelos Papakitsos
Digitally signed by Evangelos Papakitsos
Date: 2024.04.10 20:46:19 +03'00'

.....

ΔΗΛΩΣΗ ΣΥΓΓΡΑΦΕΑ ΠΤΥΧΙΑΚΗΣ ΕΡΓΑΣΙΑΣ

Ο/η κάτωθι υπογεγραμμένος **Γαλάνης Παύλος** του **Λάμπρου**, με αριθμό μητρώου **8096619** φοιτητής του Πανεπιστημίου Δυτικής Αττικής της Σχολής Μηχανικών του Τμήματος Μηχανικών Βιομηχανικής Σχεδίασης και Παραγωγής, δηλώνω υπεύθυνα ότι:

«Είμαι συγγραφέας αυτής της πτυχιακής/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 πτυχίου μου».

Ο Δηλών

Παύλος Γαλάνης

ΕΥΧΑΡΙΣΤΙΕΣ

Με την παρούσα διπλωματική εργασία ολοκληρώνονται οι σπουδές μου στο μεταπτυχιακό πρόγραμμα σπουδών «ΜΗ ΕΠΑΝΔΡΩΜΕΝΑ ΑΥΤΟΝΟΜΑ ΚΑΙ ΤΗΛΕΚΑΤΕΥΘΥΝΟΜΕΝΑ ΣΥΣΤΗΜΑΤΑ» του Τμήματος Μηχανικών Βιομηχανικής Σχεδίασης Και Παραγωγής του Πανεπιστημίου Δυτικής Αττικής.

Θα ήθελα θερμά να ευχαριστήσω τη σύζυγό μου Αναστασία όπως επίσης και τα παιδιά μου Λαμπρινή και Χαρίκλεια για τη συμπαράσταση και την υπομονή τους.

ΠΕΡΙΛΗΨΗ

Η παρούσα διπλωματική αφορά τη σχεδίαση και ανάπτυξη ενός πρωτότυπου τετρακόπτερου για χρήση σε εκπαιδευτικές δραστηριότητες. Το τετρακόπτερο είναι χαμηλού κόστους και το πλαίσιο του είναι μια ρεπλίκα του DJ1 Flame Wheel 450 (F450), υλοποιήθηκε με τον μικροελεγκτή Arduino Nano και έχει σαν αδρανειακές μονάδες τις MPU-6050 και MPU-9250. Τα μοτέρ είναι τύπου brushless τα οποία ταιριάζουν με προπέλες μήκους 10 ιντσών και 4.5΄΄ βήμα. Η μπαταρία του είναι τύπου Lipo 3Cells, χωρητικότητας 2600 mAh. Το τετρακόπτερο είναι εξοπλισμένο με μονάδα GPS της οποίας τα δεδομένα στέλνονται ασύρματα, με την βοήθεια του πομποδέκτη HC-12, σε φορητό υπολογιστή ο οποίος απεικονίζει σε γραφικό περιβάλλον τα δεδομένα με το πρόγραμμα U –Center, της U-Blox . Επιπρόσθετα έχει ενσωματωθεί ο αισθητήρας της Bosch, BMP280, ο οποίος παρέχει δεδομένα υψομέτρου και θερμοκρασίας. Τέλος χρησιμοποιείται ο πομποδέκτης nRf24 ο οποίος αποστέλλει σε πραγματικό χρόνο χαρακτηριστικά της πτήσης σε αυτόνομο σταθμό βάσης αλλά και στον φορητό υπολογιστή. Στην βασική του διαμόρφωση έχει την δυνατότητα ανύψωσης φορτίου (payload) έως 700gr.

Για τους αρχικούς πειραματισμούς και την σταθερότητα του προγράμματος ελέγχου πτήσης, τις ρυθμίσεις του τετρακόπτερου και την ασφάλεια του εξοπλισμού αλλά και του χρήστη, το drone προσαρμόζεται σε ειδική ξύλινη βάση . Η βάση είναι αυτόνομη, έχει το δικό της αδρανειακό σύστημα μέτρησης και στέλνει ασύρματα δεδομένα για τις γωνίες του drone σε φορητό υπολογιστή. Τροφοδοτείται από εξωτερική πηγή (τροφοδοτικό ή μπαταρία).

ΛΕΞΕΙΣ ΚΛΕΙΔΙΑ

Τετρακόπτερο, Arduino, Τηλεμετρία, Μονάδα Μέτρησης Αδράνειας(IMU), PID, Εκπαιδευτική πλατφόρμα.

ABSTRACT

This thesis concerns the design and development of a prototype quadcopter for educational purposes. The quadcopter is low-cost, its frame is a replica of the DJ1 Flame Wheel 450 (F450), it is implemented with the Arduino Nano microcontroller and has the MPU-6050 and the MPU-9250 as inertial units. The motors are of brushless type, which match with 10-inch-long propellers, 4.5˝ pitch. Its battery is of the Lipo 3Cells type, with a capacity of 2600 mAh. The quadcopter is equipped with a GPS unit whose data are sent wirelessly, with the help of the HC-12 transceiver, to a laptop computer which displays the data in a graphical environment with the U-Blox program, U-Center. In addition, the Bosch BMP280 sensor has been implemented giving altitude and temperature data. Finally, the nRf24 transceiver is used to send real-time data of the flight to an autonomous base station and to the laptop computer. The Quadcopter in its base configuration can lift a load up to 700gr(payload).

The drone will be adapted to a special wooden base so as to be ensured during the initial experiments, the stability of the flight control program, the settings of the quadcopter as well as the safety of the equipment and the user. The base is autonomous, it has its own inertial measurement system and is sending wirelessly the data of the drone's inclination to a laptop computer. It is powered by an external source (power supply or battery).

KEYWORDS

Quadcopter, Arduino, Telemetry, Inertial Measurement Unit (IMU), PID, Educational platform.

ΠΙΝΑΚΑΣ ΠΕΡΙΕΧΟΜΕΝΩΝ

ΚΑΤΑΛΟΓΟΣ ΠΙΝΑΚΩΝ

ΚΑΤΑΛΟΓΟΣ ΕΙΚΟΝΩΝ

ΕΥΡΕΤΗΡΙΟ ΟΡΩΝ

Accelerometer	Επιταχυνσιόμετρο
ADC	Analog to Digital Converter
Altitude	Υψόμετρο
Arm	Ενεργοποίηση
ASIC	Application - Specific Integrated Circuit
AT Commands	Attention Commands
Bandwidth	Εύρος Ζώνης
Battery Level	Στάθμη Μπαταρίας
Bearing	Ρουλεμάν
Bps	Bits Per Second
Brushless Motor	Μοτέρ χωρίς ψήκτρες
Calibration	Βαθμονόμηση
Clock	Σήμα Χρονισμού
CoG	Center of Gravity
Data Rate	Ρυθμός Μετάδοσης Δεδομένων
Disarm	Απενεργοποίηση
DOF	Degrees of Freedom
ESC	Electronic Speed Controller
FIFO	First In First Out
FoV	Field Of View
GPS(Global Positioning System)	Παγκόσμιο Σύστημα Στιγματοθέτησης
Gyro	Γυροσκόπιο
Heading	Γωνία διεύθυνσης ως προς τον άξονα Z
Hover	Σταθερή αιώρηση σε συγκεκριμένο ύψος
IDE	Integrated Development Environment
Idle	Περιστροφή μοτέρ σε Throttle 10%
IIC	Inter-Integrated Circuit
IMU(Inertial Measurement Unit)	Αδρανειακό Σύστημα Μέτρησης
ISR	Interrupt Service Routine
Jitter	Διακύμανση
LCD	Liquid Crystal Display
LiPo Battery	Μπαταρία Πολυμερούς Ιόντων Λιθίου
Magnetic Declination	Μαγνητική Απόκλιση
MARG	Magnetic, Angular Rate and Gravity
Moment	Ροπή
Payload	Φορτίο

PID	Proportional–Integral–Derivative
Pitch	Γωνία διεύθυνσης ως προς τον άξονα Y
PWM	Pulse Width Modulation
Quadcopter	Τετρακόπτερο
Quaternion	Τετραδόνιο
Roll	Γωνία διεύθυνσης ως προς τον άξονα X
Rotation Matrix	Πίνακας περιστροφής
SPI	Serial Peripheral Interface
Telemetry	Τηλεμετρία
Temperature	Θερμοκρασία
Throttle	Ισχύς
Thrust	Ώθηση - Ώση
ToF	Time of Flight
UART	Universal Asynchronous Rx / Tx
Yaw	Γωνία διεύθυνσης ως προς τον άξονα Z

ΕΙΣΑΓΩΓΗ

Η παρούσα διπλωματική εργασία, που εκπονήθηκε στα πλαίσια αποφοίτησης του Μεταπτυχιακού Προγράμματος Σπουδών ΄΄Μη Επανδρωμένα Αυτόνομα και Τηλεκατευθυνόμενα Συστήματα΄΄ του τμήματος Μηχανικών Βιομηχανικής Σχεδίασης και Παραγωγής του πανεπιστημίου Δυτικής Αττικής, έχει ως αντικείμενο τον σχεδιασμό και την υλοποίηση ενός τετρακόπτερου για χρήση σε εκπαιδευτικές δραστηριότητες με δυνατότητα τηλεμετρίας σε πραγματικό χρόνο. Στοχεύει να αναλύσει τις αρχές λειτουργίας βασικών εξαρτημάτων που συνιστούν ένα τετρακόπτερο καθώς και το τρόπο λειτουργίας αυτού. Έτσι λοιπόν στο πρώτο κεφάλαιο παρουσιάζονται θεωρητικές αναλύσεις, γνώση που θα χρησιμοποιηθεί για την περαιτέρω κατανόηση των αρχών λειτουργίας ενώ στο δεύτερο μέρος αναλύονται όλα τα υλικά που χρησιμοποιήθηκαν για την ολοκλήρωση της κατασκευής. Στο τρίτο κεφάλαιο αναλύεται ο τρόπος λειτουργίας ενός Χ τύπου τετρακόπτερου ακολουθώντας μια σειριακή αντιμετώπιση των προβλημάτων που προκύπτουν. Αρχίζοντας από τον τρόπο ανάγνωσης και αναπαραγωγής του παλμού του τηλεχειρισμού, την οδήγηση των μοτέρ, το αδρανειακό σύστημα μέτρησης, την ασύρματη εκπομπή – λήψη δεδομένων και καταλήγοντας στην κατασκευή ενός σταθμού ελέγχου για να ελεγχθεί και να ρυθμιστεί με ασφάλεια το σκάφος .

Η κατασκευή εστιάζει ως επί το πλείστων στο ηλεκτρολογικό – ηλεκτρονικό κομμάτι . Για αυτόν τον λόγο έχει επιλεχθεί ένα έτοιμο πλαίσιο, μια ρεπλίκα του DJ1 Flame Wheel 450 (F450), ένα αρκετά ελαφρύ και πολύ-δοκιμασμένο πλαίσιο το οποίο θα υλοποιήσει ένα τετρακόπτερο τύπου Χ . Τα ηλεκτρονικά του τετρακόπτερου πλαισιώνουν δύο Arduino Nano ως μικροελεγκτές. Το ένα έχει τον μοναδικό ρόλο του Flight Controller ενώ το δεύτερο χρησιμοποιείται για να ελέγξει τους αισθητήρες της κατασκευής.

Ο βασικός μικροελεγκτής έχει τον ρόλο να αναγνωρίσει τον τηλεχειρισμό εισόδου, να μετρήσει το εύρος του παλμού και να τον αναπαράγει ώστε να οδηγήσει τα τέσσερα μοτέρ. Το αδρανειακό σύστημα μέτρησης είναι ο αισθητήρας MPU – 6050 σε συνδεσμολογία IIC. Η δοκιμασία ήταν να διαμορφωθεί ο κώδικας τόσο απλός ώστε να επιτευχθεί το μικρότερο δυνατό loop time . Έτσι επιτεύχθηκε ένας ρυθμός ανανέωσης των 200 Hz. Ο ρυθμός αυτός είναι ο ρυθμός ανανέωσης των Electronic Speed Controllers που τελικά οδηγούν τα μοτέρ. Τα μοτέρ που επιλέχθηκαν είναι τα 2212 1000KV Brushless Motors και ταίριαξαν με προπέλες 10 ιντσών, βήματος 4.5΄΄. Τα μοτέρ μετρήθηκαν και απέδωσαν περί τα 730 γρ. κατά μέσο όρο μέγιστο thrust (με τάση αναφοράς τα 12.2 V) με μέγιστη κατανάλωση ρεύματος περί τα 15 A. Η μπαταρία που επιλέχθηκε είναι τύπου Lipo 3 Cells με χωρητικότητα 2600mAh. Ο τηλεχειρισμός είναι ένας εξακάναλος με δυνατότητα frequency hopping στα 2.4 GHz.

Το δεύτερο Arduino Nano είναι ο μικροελεγκτής που ασχολείται με τα υπόλοιπα χαρακτηριστικά του τετρακόπτερου. Έτσι χρησιμοποιεί σαν αδρανειακό σύστημα μέτρησης το MPU-9250, μια εξελιγμένη έκδοση του MPU-6050 με ενσωματωμένη πυξίδα σε συνδεσμολογία IIC. Επίσης έχει ενσωματωθεί ο αισθητήρας της Bosch BMP280, παρέχοντας πληροφορίες για το υψόμετρο και την θερμοκρασία του περιβάλλοντος. Ο πομποδέκτης nRF24 αναλαμβάνει να εκπέμψει σε πραγματικό χρόνο πληροφορίες του τετρακόπτερου όπως:Throttle, Battery level, Temperature, Altitude, Heading, Pitch και Roll σε αυτόνομο δέκτη με οθόνη LCD 16X2.

Παράλληλα υπάρχει η δυνατότητα τα ίδια δεδομένα να απεικονιστούν και σε φορητό υπολογιστή με την χρήση του προγράμματος Excel και Excel Data Streamer. Τέλος έχει ενσωματωθεί μια μονάδα GPS της U-blox, η NEO M8N ή οποία σε συνδεσμολογία UART αποστέλλει σειριακά, με την χρήση του πομποδέκτη HC-12, τα δεδομένα στον φορητό υπολογιστή. Εκεί σε γραφικό περιβάλλον με την χρήση του προγράμματος της ίδιας εταιρίας, το U – Center απεικονίζονται πληθώρα χαρακτηριστικών. Τα υπόλοιπα ηλεκτρονικά – ηλεκτρολογικά εξαρτήματα που χρησιμοποιούνται εκτός από τους αισθητήρες είναι κυρίως ρυθμιστές DC τάσης και ποτενσιόμετρα. Οι συνδέσεις είναι απλά καλώδια σιλικόνης και τύπου Dupont.

Ολοκληρώνοντας προκύπτει ένα τετρακόπτερο βάρους περίπου 1000 γρ. με συνολικό μέγιστο Thrust περί τα 2800 γρ. και μέγιστη κατανάλωση ρεύματος περί τα 60 Α το οποίο θα κάνει hover στο 50% του Throttle. Σε αυτήν τη τιμή η κατανάλωση του ρεύματος είναι γύρω στα 13 Α οπότε η μπαταρία δίνει μια αυτονομία περί τα 10 λεπτά.

Για τους αρχικούς πειραματισμούς και την σταθερότητα του προγράμματος ελέγχου πτήσης, τις ρυθμίσεις του τετρακόπτερου και την ασφάλεια του εξοπλισμού αλλά και του χρήστη κατασκευάστηκε ειδική ξύλινη βάση. Το τετρακόπτερο ελέγχθηκε κυρίως για την σταθερότητα του κώδικα και για την ρύθμιση του PID για περισσότερες από 30 ώρες. Η βάση είναι αυτόνομη και έχει το δικό της αδρανειακό σύστημα που στηρίζεται και αυτό στο MPU-6050, στέλνει ασύρματα δεδομένα για τις γωνίες του drone σε φορητό υπολογιστή, που με την βοήθεια αρχικά του Minitab και του Excel Data streamer αργότερα, ρυθμίστηκαν οι παράμετροι PID. Η ξύλινη βάση που φιλοξενεί το τετρακόπτερο εφαρμόζει σε σωλήνα αλουμινίου η οποίος εδράζεται σε bearings κλειστού τύπου για όσο το δυνατό λιγότερες τριβές. Η βάση αυτή τροφοδοτείται από εξωτερική πηγή, τροφοδοτικό ή μπαταρία.

Η μεθοδολογία που χρησιμοποιήθηκε στηρίχτηκε στον τεμαχισμό του προβλήματος που πραγματεύεται η διπλωματική εργασία σε μικρά αυτόνομα προβλήματα. Κάθε ένα από αυτά μελετήθηκε και αναπτύχθηκε αρχικά σε πρόγραμμα εξομοίωσης ή σε πραγματικές συνθήκες με την χρήση breadboard με την προοπτική να ενσωματωθεί τελικά στην κατασκευή. Κατά αυτό το τρόπο τα παρακάτω επιμέρους προβλήματα προέκυψαν :

- Ανάγνωση και αναπαραγωγή σήματος τηλεχειρισμού
- Επιλογή μοτέρ και προπελών σε συνάρτηση με το βάρος του τετρακόπτερου
- Δυναμομέτρηση μοτέρ – προπελών και κατανάλωση ρεύματος, επιλογή μπαταρίας
- Αδρανειακό σύστημα μέτρησης
- PID
- Πομποδέκτες σε διάφορες συχνότητες
- Ηλεκτρονικές πυξίδες
- Time of flight αισθητήρες
- Αισθητήρες βαρομετρικής πίεσης
- GPS
- Σταθμός ελέγχου τετρακόπτερου

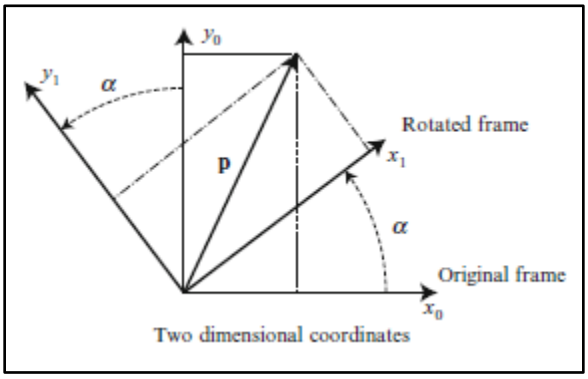
Οι πληροφορίες για την αντιμετώπιση των επιμέρους προβλημάτων προήλθαν κυρίως από αναζήτηση στο διαδίκτυο και από πληροφορίες από τα εγχειρίδια των κατασκευαστών των αισθητήρων αλλά επίσης και πολλές λύσεις στηρίχτηκαν σε προσωπικές εμπειρίες από την

εργασιακή μου ιδιότητα ως ηλεκτρονικός αεροσκαφών . Για την ολοκλήρωση της διπλωματικής τα παρακάτω προγράμματα χρησιμοποιήθηκαν.

1. ΘΕΩΡΗΤΙΚΟ ΥΠΟΒΑΘΡΟ

1.1 Κινηματική Ανάλυση

Ο στόχος αυτής της παραγράφου είναι να ορίσει έναν μετασχηματισμό πλαισίου συντεταγμένων (coordinate frame transformation) και να εξηγηθεί ο σχετικός μαθηματικός κανόνας. Ο πίνακας συνημίτονων ή direct cosine matrix (DCM) εισάγεται και παρουσιάζονται οι βασικές του ιδιότητες. Μια διαφορική περιστροφή καθορίζει το ρυθμό μεταβολής του DCM πίνακα. Παρουσιάζεται η θεμελιώδης ιδιότητα της απλής άθροισης των γωνιακών αλλαγών.



Εικόνα 1. Περιστροφή πλαισίου σε σύστημα δύο αξόνων. [1]

Μια απροσδιόριστη κίνηση ενός σώματος άκαμπτου, μπορεί να περιγραφεί με συνιστώσες δύο διαστάσεων. Έτσι αν θεωρήσουμε ένα διάνυσμα p που ορίζεται σε δύο ορθογώνια πλαίσια συντεταγμένων που περιστρέφονται κατά γωνία α όπως φαίνεται στην εικόνα 1, αυτό μπορεί να αναλυθεί στις συντεταγμένες :

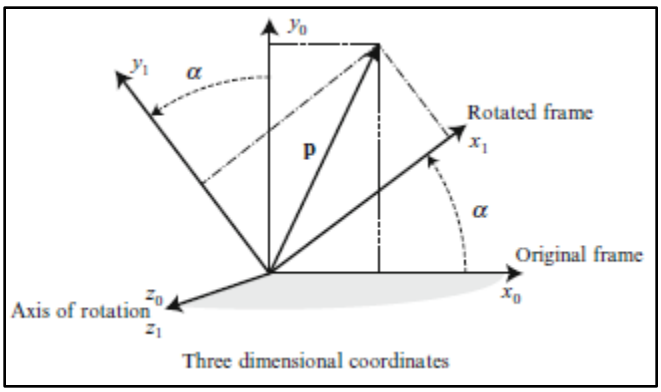
$$x_1 = x_0 \cos\alpha + y_0 \sin\alpha$$

$$y_1 = -x_0 \sin\alpha + y_0 \cos\alpha$$

όπου α είναι η γωνία περιστροφής του πλαισίου και 0 και 1, η αρχική και η τελική θέση αντίστοιχα. Αυτή η περιστροφή μπορεί να περιγραφεί με την χρήση rotation matrix ($R\_0\^1$) και προκύπτει ο παρακάτω μετασχηματισμός :

$$\begin{bmatrix} x_1 \\ y_1 \end{bmatrix} = R_0^1 * \begin{bmatrix} x_0 \\ y_0 \end{bmatrix} \text{ ή } R_0^1 = \begin{bmatrix} cos\alpha & \sin a \\ -\sin a & \cos a \end{bmatrix}$$

Αν ακολουθήσουμε την ίδια λογική και μεταφέρουμε την κίνηση ενός άκαμπτου σώματος σε ένα σύστημα τριών διαστάσεων όπως φαίνεται στην εικόνα 2 μπορούμε να αναλύσουμε το διάνυσμα p σε τρείς συνιστώσες x, y, z και να διαμορφώσουμε αντίστοιχα τους τρείς πίνακες περιστροφής για γωνία περιστροφής a και αρχική θέση 0 και τελική 1.



Εικόνα 2. Περιστροφή πλαισίου σε σύστημα τριών αξόνων. [1]

Για να προκύψουν οι τελικοί πίνακες θεωρούμε ότι κάθε φορά το πλαίσιο περιστρέφεται κατά ένα άξονα.

$$xR_0^1 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\alpha & \sin\alpha \\ 0 & -\sin\alpha & \cos\alpha \end{bmatrix}, \; yR_0^1 = \begin{bmatrix} \cos\alpha & 0 & -\sin\alpha \\ 0 & 1 & 0 \\ \sin\alpha & 0 & \cos\alpha \end{bmatrix}$$

$$zR_0^1 = \begin{bmatrix} \cos\alpha & \sin\alpha & 0 \\ -\sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Για μια συγκεκριμένη ακολουθία περιστροφών με αναφορά σε κάθε άξονα περιστροφής ο τελικός μετασχηματισμός μπορεί να περιγραφεί με έναν πίνακα περιστροφής ο οποίος αποτελείται από τρεις σε ακολουθία περιστροφές οι οποίες καλούνται γωνίες Euler. Τελικά για ένα διάνυσμα $p_0 = [x_0,y_0,z_0]^T$ στον χώρο των τριών διαστάσεων με αρχική θέση την 0 και τελική θέση την 1 η περιστροφή μπορεί να περιγραφεί με ένα πίνακα $R\_0\text{^}1$ αποτελούμενο από τρείς σε ακολουθία περιστροφές όπως φαίνεται παρακάτω:

$$\begin{bmatrix} x_1 \\ y_1 \\ z_1 \end{bmatrix} = R_0^1 * \begin{bmatrix} x_0 \\ y_0 \\ z_0 \end{bmatrix} \qquad \text{ή}$$

$$\begin{bmatrix} x_1 \\ y_1 \\ z_1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\varphi & \sin\varphi \\ 0 & -\sin\varphi & \cos\varphi \end{bmatrix} \begin{bmatrix} \cos\theta & 0 & -\sin\theta \\ 0 & 1 & 0 \\ \sin\theta & 0 & \cos\theta \end{bmatrix} \begin{bmatrix} \cos\psi & \sin\psi & 0 \\ -\sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} x_0 \\ y_0 \\ z_0 \end{bmatrix}$$

και τελικά προκύπτει

$$R_0^1 = \begin{bmatrix} cos\theta cos\psi & cos\theta sin\psi & -sin\theta \\ -cos\theta\ sin\psi + sin\varphi\ sin\theta\ cos\psi & cos\varphi\ cos\psi + sin\varphi\ sin\theta\ sin\psi & sin\varphi cos\theta \\ sin\varphi\ sin\psi + cos\varphi\ sin\theta\ cos\psi & -sin\varphi\ cos\psi + cos\varphi\ sin\theta\ sin\psi & cos\varphi cos\theta \end{bmatrix}$$

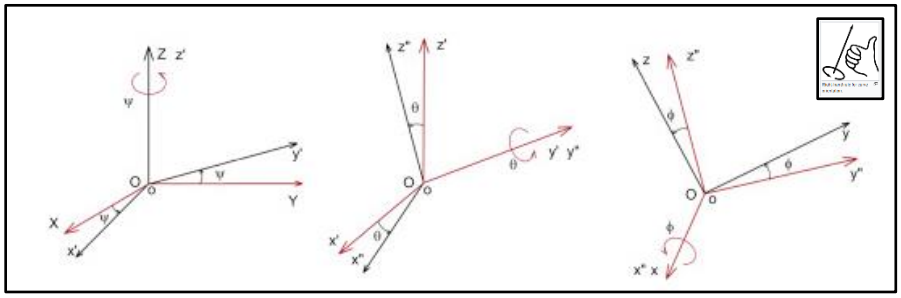
όπου φ η γωνία περιστροφής ως προς τον άξονα x, θ ως προς τον άξονα y και ψ ως προς τον άξονα z αντίστοιχα.

Συνοψίζοντας μπορούμε να πούμε ότι οι γωνίες Euler περιγράφουν τον προσανατολισμό κάποιου άκαμπτου σώματος με περιστροφή γύρω από τους άξονες x, y και z. Αυτές οι γωνίες σημειώνονται ως:

φ: roll angle (περιστροφή γύρω από τον άξονα x),
θ: pitch angle (περιστροφή γύρω από τον άξονα y)
ψ: yaw or heading angle (περιστροφή γύρω από τον άξονα z).

Υπάρχουν 12 δυνατοί συνδυασμοί περιστροφής γύρω από τους άξονες x, y και z . Ο πιο συχνά χρησιμοποιούμενος συνδυασμός που περιγράφει την κίνηση ενός άκαμπτου σώματος ή ενός μικρό-οχήματος είναι ο συνδυασμός 3–2–1, όπου ο προσανατολισμός κάποιου πλαισίου συντεταγμένων σύμφωνα με το πλαίσιο συντεταγμένων αναφοράς λαμβάνεται από την αρχική στάση όπου και τα δύο πλαίσια συντεταγμένων είναι ευθυγραμμισμένα και στη συνέχεια το πλαίσιο του σώματος περιστρέφεται με την ακόλουθη σειρά:

Yaw: rotation about the *z*-axis by $\psi(t)$ to get $x^I y^I z^I$ or $x^I y^I Z$
Pitch: rotation about the y^I axis by $\theta(t)$ to get $x^{II} y^{II} z^{II}$ or $x^{II} y^I z^{II}$
Roll: rotation about the y^I axis by $\varphi(t)$ to get xyz or $x^{II} yz$



Εικόνα 3. Περιστροφή 3-2-1 και κανόνας του δεξιού χεριού

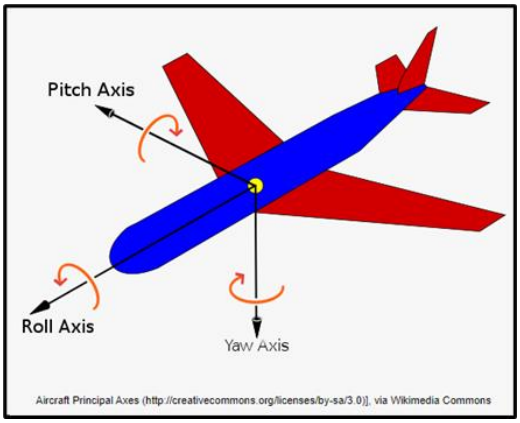
https://www.sciencedirect.com/topics/engineering/euler-angle

και ο rotation matrix που προκύπτει είναι ο παρακάτω :

$$\begin{bmatrix} x_1 \\ y_1 \\ z_1 \end{bmatrix} = \begin{bmatrix} \cos\psi & \sin\psi & 0 \\ -\sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\theta & 0 & -\sin\theta \\ 0 & 1 & 0 \\ \sin\theta & 0 & \cos\theta \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\varphi & \sin\varphi \\ 0 & -\sin\varphi & \cos\varphi \end{bmatrix} * \begin{bmatrix} x_0 \\ y_0 \\ z_0 \end{bmatrix}$$

Παράλληλα μια ακόμη παραδοχή χρειάζεται για να περιγράψουμε την κίνηση ενός μικρό-οχήματος στον χώρο των τριών διαστάσεων, εισάγοντας την συμβατική μέθοδος απεικόνισης:

Παρατηρούμε ότι ο άξονας z είναι παράλληλος με το διάνυσμα της βαρύτητας g. Κατά αυτήν τη μέθοδο αν το σκάφος περιστραφεί κατά $\pi/2$ ως προς τον άξονα x ή αλλιώς κάνει Roll κατά 90^0 το διάνυσμα του x θα ταυτιστεί με αυτό του z ή του διανύσματος της βαρύτητας g. Αυτό το γεγονός είναι ένα μειονέκτημα των γωνιών Euler, είναι ένα μαθηματικό πρόβλημα, συνήθως



Εικόνα 4. Συμβατική μέθοδος απεικόνισης . Το μοντέλο του αεροπλάνου.

οδηγεί σε λανθασμένα αποτελέσματα και είναι γνωστό ως Gimbal Lock. Ένας τρόπος να το ξεπεράσουμε είναι τα Quaternions για τα οποία θα μιλήσουμε στην επόμενη παράγραφο [1] [2].

1.2 Quaternions

Ένα Quaternion (Τετραδόνιο)είναι ένας τεσσάρων διαστάσεων μιγαδικός αριθμός που μπορεί να χρησιμοποιηθεί για να αναπαραστήσει τον προσανατολισμό ενός άκαμπτου σώματος ή ενός πλαισίου συντεταγμένων σε τρισδιάστατο χώρο. Ένα Quaternion q = ($\dot{q}$. $\vec{q}$) αναπαρίσταται από ένα πραγματικό μέρος $\dot{q} \in \mathbb{R}$ και ένα μέρος που περιγράφει το διάνυσμα στον χώρο $\vec{q} \in \mathbb{R}^3$. Το Quaternion εκφράζεται από την σχέση

$$q = \alpha + b\,\mathbf{i} + c\,\mathbf{j} + d\,\mathbf{k}$$

όπου a,b,c,d είναι πραγματικοί αριθμοί και **i, j, k** είναι τα σύμβολα που μπορούν να ερμηνευτούν ως μονάδες-διανύσματα που δείχνουν κατά μήκος των τριών αξόνων. Για τα **i, j, k** ισχύει

$$\mathbf{i^2 = j^2 = k^2 = i\,j\,k = -1}$$

Quaternion multiplication table

↓ × →	1	i	j	k
1	1	i	j	k
i	i	−1	k	−j
j	j	−k	−1	i
k	k	j	−i	−1

Πίνακας 1. Πίνακας πολλαπλασιασμού Quaternions

https://en.wikipedia.org/wiki/Quaternion

Ένα quaternion είναι αυτό που ικανοποιεί την σχέση :

$$q \cdot q = \dot{q}^2 + \vec{q} \cdot \vec{q} = 1$$

Το μέρος του Quaternion που περιγράφει το διάνυσμα $\vec{q}$ είναι παράλληλο στον άξονα περιστροφής και το πραγματικό μέρος του είναι αυτό που περιγράφει την γωνία περιστροφής. Έτσι ένας πίνακας περιστροφής R = Rot ($\bar{\theta}$, ν) μπορεί να περιγραφεί από μια μονάδα Quaternion :

$$q = (\cos(\frac{\bar{\theta}}{2}), \sin(\frac{\bar{\theta}}{2})\,\nu)\,,$$

Τα Quaternions συντίθεται μέσω πολλαπλασιασμού που αναπαρίσταται με το σύμβολο ο και ορίζονται όπως:

$$q_1 \, o \, q_2 = (\, \dot{q}_1 \dot{q}_2 - \vec{q}_1 . \vec{q}_2 \,, \dot{q}_1 \vec{q}_2 + \dot{q}_2 \vec{q}_1 - \vec{q}_1 \, x \, \vec{q}_2 \,)$$

Ένα διάνυσμα σε ένα τρισδιάστατο χώρο μπορεί να περιγραφεί ως ένα καθαρό quaternion (para quaternion), δηλαδή ένα quaternion χωρίς πραγματικό μέρος:

$$q = 0 + xi + yj + zk\,.$$

Μια περιστροφή περιγράφεται από ένα quaternion q_R που έχει μέτρο ίσο με ένα. Η περιστροφή από ένα μέρος Α σε ένα μέρος Β δίνεται από την σχέση:

$$q_B = q_R\, q_A\, q_R^*$$

Αυτή η εξίσωση θα καταλήξει σε ένα διάνυσμα που δεν έχει πραγματικό μέρος και περιγράφεται από την εξίσωση:

$$q_B = q_R\, q_A\, q_R^* = (q_0 + q_1 i + q_2 j + q_3 k)(xi+yj+zk)\,(q_0 - q_1 i - q_2 j - q_3 k)$$

$$= (x(q_0^2 + q_1^2 - q_2^2 - q_3^2) + 2y(q_1 q_2 - q_0 q_3) + 2z(q_0 q_2 + q_1 q_3)i +$$

$$(2x(q_1 q_2 + q_0 q_3) + y(q_0^2 - q_1^2 + q_2^2 - q_3^2) + 2z(q_2 q_3 - q_0 q_1)\, j +$$

$$(2x(q_1 q_3 - q_0 q_2) + 2y(q_0 q_1 + q_2 q_3) + z\,(q_0^2 - q_1^2 - q_2^2 + q_3^2)k$$

Επιπρόσθετα αυτή η εξίσωση μπορεί να περιγραφεί και με την βοήθεια ενός rotational matrix και καταλήγει ως:

$$M = \begin{bmatrix} q0^2 + q1^2 - q2^2 - q3^2 & 2(q1q2 - q0q3) & 2(q0q2 + q1q3) \\ 2(q1q2 + q0q3) & q0^2 - q1^2 + q2^2 - q3^2 & 2(q2q3 - q0q1) \\ 2(q1q3 - q0q2) & 2(q0q1 + q2q3) & q0^2 - q1^2 - q2^2 + q3^2 \end{bmatrix}$$

Ισχύει ότι $q_0^2 + q_1^2 + q_2^2 + q_3^2 = 1$, ο πίνακας απλοποιείται ως εξής:

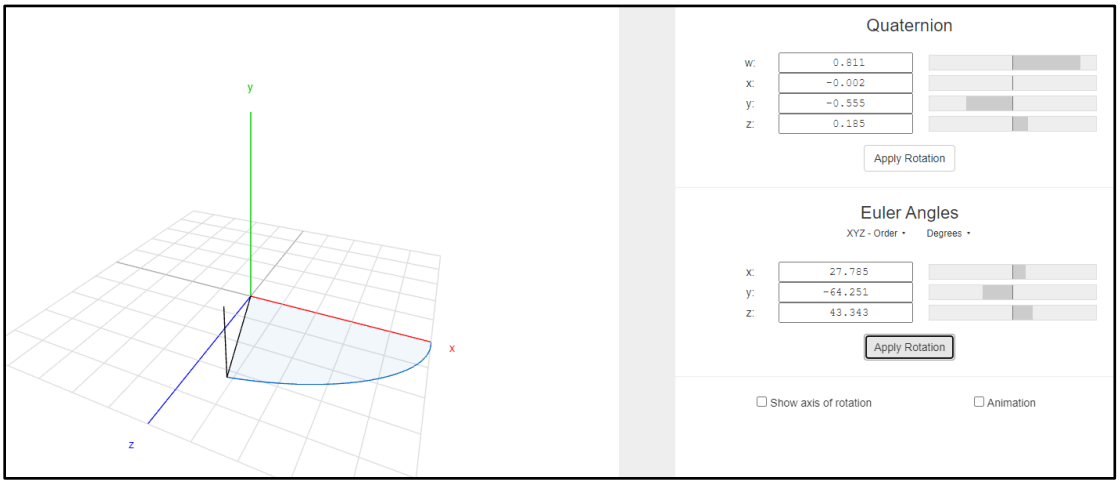
$$M = \begin{bmatrix} 2q0^2 + 2q1^2 - 1 & 2(q1q2 - q0q3) & 2(q0q2 + q1q3) \\ 2(q1q2 + q0q3) & 2q0^2 + 2q2^2 - 1 & 2(q2q3 - q0q1) \\ 2(q1q3 - q0q2) & 2(q0q1 + q2q3) & 2q0^2 + 2q3^2 - 1 \end{bmatrix}$$

Τέλος είναι δυνατό να υπολογίσουμε τις γωνίες Euler από τα quaternions σύμφωνα με τις παρακάτω σχέσεις:

$$\begin{matrix} \varphi \\ \theta = \\ \psi \end{matrix} \quad \begin{matrix} atan2\{\, 2(q1q2 + q0q3), [1 - 2(q0^2 + q1^2)]\} \\ sin^{-1}[2(q1q3 - q0q2)] \\ atan2\{2(q0q1 + q2q3), [[1 - 2(q1^2 + q2^2)]\} \end{matrix}$$

Τα Quaternions είναι συχνά ο επιλεγμένος τρόπος για να παραμετροποιήσει κάποιος την κίνηση ενός μικρού – οχήματος στον χώρο και αυτό γιατί απαιτεί λιγότερη επεξεργαστική μνήμη και χρόνο υπολογισμού από το να αποθηκεύεις και πολλαπλασιάζεις rotational matrices [3].

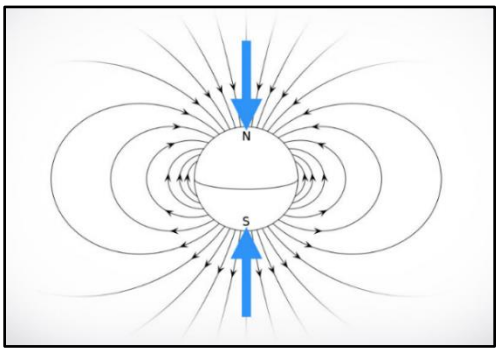
Ένας τρόπος να κατανοηθεί η σχέση μεταξύ Euler γωνιών και Quaternions είναι ο σύνδεσμος https://quaternions.online/ που αναπαριστά σε γραφικό περιβάλλον το αποτέλεσμα.



Εικόνα 5. Γραφική αναπαράσταση σχέσης Quaternions – Euler angles

1.3 Electronic Compass

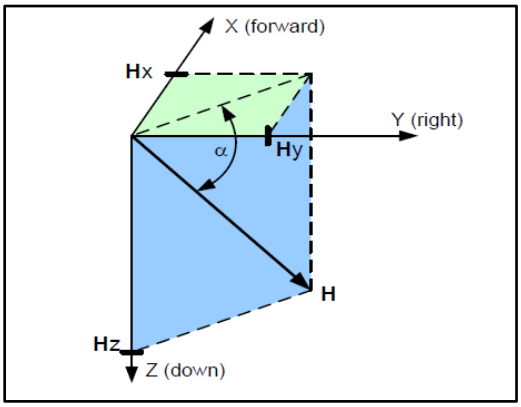
Οι ηλεκτρονικές πυξίδες βασίζονται στις μετρήσεις του μαγνητικού πεδίου της Γης. Η Γη θεωρείται ένα μαγνητικό δίπολο με πόλους που βρίσκονται κοντά στον γεωγραφικό Βόρειος Πόλο και Νότιος Πόλο. Παρατηρώντας το διάνυσμα του μαγνητικού πεδίου διακρίνουμε ότι είναι παράλληλο με την επιφάνεια κοντά στον ισημερινό. Ο ισημερινός είναι μια νοητή γραμμή γύρω από τη μέση της γης. Βρίσκεται στα μισά του δρόμου μεταξύ του βόρειου και του νότιου πόλου, σε γεωγραφικό πλάτος 0 μοίρες (Latitude 0^o).



Εικόνα 6. Το Μαγνητικό πεδίο της Γης

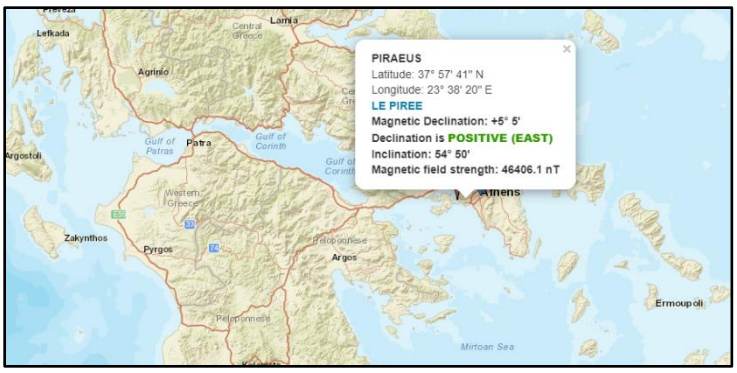
(https://www.boldmethod.com)

Στο Βόρειο ημισφαίριο, αυτό το διάνυσμα δείχνει προς τα κάτω προς τον Βόρειο πόλο.



Εικόνα 7. Γωνία κλίσης διανύσματος μαγνητικού πεδίου [5]

Η γωνία μεταξύ του διανύσματος του μαγνητικού πεδίου Η και του οριζόντιου επιπέδου ονομάζεται γωνία κλίσης α (inclination angle) και για κάθε τόπο είναι διαφορετική . Ένας τρόπος να βρούμε την γωνία αυτή είναι από την ιστοσελίδα https://www.magnetic-declination.com/ , όπως παρατηρούμε από την εικόνα 8 για παράδειγμα για την περιοχή του Πειραιά είναι + 5^o και 5 ΄.



Εικόνα 8. Magnetic Inclination . https://www.magnetic-declination.com/

Παρατηρώντας την εικόνα 7 μπορούμε να υπολογίσουμε το Heading ως το τόξο εφαπτομένης

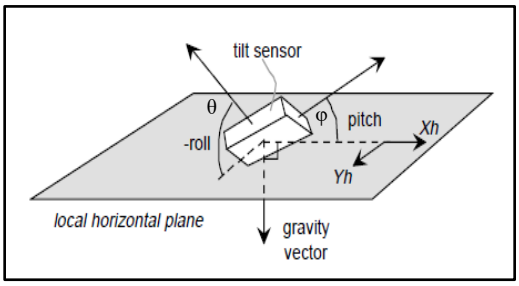
$$Heading = \arctan\left(\frac{H_y}{H_X}\right)$$

Ο υπολογισμός του Heading σύμφωνα με τον παραπάνω τύπο είναι σωστός αν και μόνο αν η πυξίδα είναι εντελώς οριζόντια. Σε κάθε άλλη περίπτωση τα αποτελέσματα είναι λανθασμένα. Αυτός είναι και ο λόγος που η συμβατικές πυξίδες στην αεροπλοΐα είναι εγκλεισμένες σε κλειστό κλωβό με υγρό αλκοόλη ή κηροζίνη, ώστε να μειώνονται οι τριβές, η πυξίδα να κινείται ελεύθερα και να είναι πάντα οριζόντια ανεξάρτητα από τις κλίσεις του σκάφους (mechanical gimbal).

Εικόνα 9. Stand By Magnetic Compass

https://www.boldmethod.com

Αυτό το μειονέκτημα των ηλεκτρονικών πυξίδων διορθώνεται με την χρήση ενός επιταχυνσιόμετρου (Accelerometer). Γνωρίζοντας κάθε στιγμή την γωνία pitch και roll διορθώνεται το Heading και προκύπτει η Tilt Compensated Electronic Compass. Με την βοήθεια των rotation matrices παρακάτω θα συνδυάσουμε το Tilt Compensated Heading με τις γωνίες pitch και roll.



Εικόνα 10. Περιστροφή πλαισίου στο χώρο[7]

Αν ορίσουμε θ την γωνία περιστροφής ως προς τον άξονα X (roll angle) και φ την γωνία περιστροφής ως προς τον άξονα Y (pitch angle) τότε οι παρακάτω πίνακες περιστροφής προκύπτουν:

$$Rot_R = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & \sin\theta \\ 0 & -\sin\theta & \cos\theta \end{bmatrix}$$

$$Rot_P = \begin{bmatrix} \cos\varphi & 0 & -\sin\varphi \\ 0 & 1 & 0 \\ \sin\varphi & 0 & \cos\varphi \end{bmatrix}$$

Η περιστροφή ως προς τον άξονα Z δεν λαμβάνεται υπόψιν αφού δεν αφορά τον υπολογισμό του Heading. Επίσης θεωρούμε ότι η περιστροφή γίνεται πρώτα στον άξονα X (roll) και μετά στον άξονα Y (pitch) και ότι η περιστροφή ακολουθεί τον κανόνα του δεξιού χεριού, τελικά ισχύει:

$$\begin{bmatrix} \boldsymbol{x_h} \\ \boldsymbol{y_h} \\ \boldsymbol{z_h} \end{bmatrix} = Rot_R * Rot_P * \begin{bmatrix} \boldsymbol{x} \\ \boldsymbol{y} \\ \boldsymbol{z} \end{bmatrix} \text{ ή}$$

$$\begin{bmatrix} \boldsymbol{x_h} \\ \boldsymbol{y_h} \\ \boldsymbol{z_h} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\varphi & \sin\varphi \\ 0 & -\sin\varphi & \cos\varphi \end{bmatrix} \begin{bmatrix} \cos\theta & 0 & -\sin\theta \\ 0 & 1 & 0 \\ \sin\theta & 0 & \cos\theta \end{bmatrix} * \begin{bmatrix} \boldsymbol{x} \\ \boldsymbol{y} \\ \boldsymbol{z} \end{bmatrix} \text{ ή}$$

$$\begin{bmatrix} \boldsymbol{x_h} \\ \boldsymbol{y_h} \\ \boldsymbol{z_h} \end{bmatrix} = \begin{bmatrix} \cos\varphi & 0 & -sin\varphi \\ sin\theta sin\varphi & cos\theta & sin\theta cos\varphi \\ cos\theta \sin\varphi & -sin\theta & cos\theta \cos\varphi \end{bmatrix} * \begin{bmatrix} \boldsymbol{x} \\ \boldsymbol{y} \\ \boldsymbol{z} \end{bmatrix} \text{ ή}$$

$$\boldsymbol{x_h} = xcos\varphi + ysin\theta sin\varphi + zcos\theta cos\varphi$$

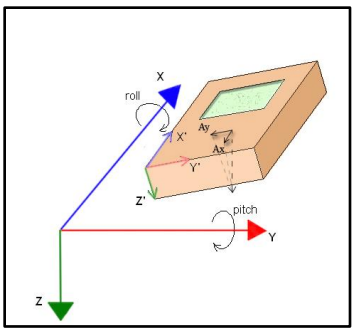
$$\boldsymbol{y_h} = ycos\theta - zsin\theta$$

όπου x, y, z τα δεδομένα από την πυξίδα και θ, φ οι κλίσεις της πυξίδας ως προς το σύστημα αναφοράς.

Όσο αφορά το επιταχυνσιόμετρο, οι τιμές A_x - A_y του, μπορούν να θεωρηθούν ως το ημίτονο των γωνιών φ και θ.

$$A_x = -\sin\varphi$$

$$A_y = \sin\theta$$



Εικόνα 11. Περιστροφή επιταχυνσιόμετρου στο χώρο [5]

Αυτή η υπόθεση θεωρείται σωστή όταν η κλίση είναι πολύ μικρή. Οι έξοδοι του επιταχυνσιόμετρου A_x - A_y μπορεί να περιγραφούν από το παρακάτω rotation matrix:

$$\begin{bmatrix} \boldsymbol{A_x} \\ \boldsymbol{A_y} \\ \boldsymbol{A_z} \end{bmatrix} = \begin{bmatrix} \cos\varphi & 0 & -sin\varphi \\ sin\theta sin\varphi & cos\theta & sin\theta cos\varphi \\ cos\theta \sin\varphi & -sin\theta & cos\theta \cos\varphi \end{bmatrix} * \begin{bmatrix} \boldsymbol{Raw_x} \\ \boldsymbol{Raw_y} \\ \boldsymbol{Raw_z} \end{bmatrix}$$

Στην περίπτωση που το επιταχυνσιόμετρο είναι σε ηρεμία είναι $\boldsymbol{Raw_x = Raw_y = 0}$ και $\boldsymbol{Raw_z = 1\ g}$

Προκύπτει $\begin{bmatrix} \boldsymbol{A_x} \\ \boldsymbol{A_y} \\ \boldsymbol{A_z} \end{bmatrix} = \begin{bmatrix} \cos\varphi & 0 & -sin\varphi \\ sin\theta sin\varphi & cos\theta & sin\theta cos\varphi \\ cos\theta \sin\varphi & -sin\theta & cos\theta \cos\varphi \end{bmatrix} * \begin{bmatrix} \boldsymbol{0} \\ \boldsymbol{0} \\ \boldsymbol{1} \end{bmatrix}$ ή

$A_x = -\sin\varphi$

$A_y = \sin\theta \cos\varphi$ ή

Pitch = $\varphi = asin\ (-A_x)$

Roll = $\theta = asin\ (\frac{A_y}{\cos\ (Pitch)})$

Συνοψίζοντας, ο συνδυασμός των παρακάτω σχέσεων:

$$\boldsymbol{x_h} = xcos\varphi + ysin\theta sin\varphi + zcos\theta cos\varphi$$

$$\boldsymbol{y_h} = ycos\theta - zsin\theta$$

$$\boldsymbol{\varphi} = asin\ (-A_x)$$

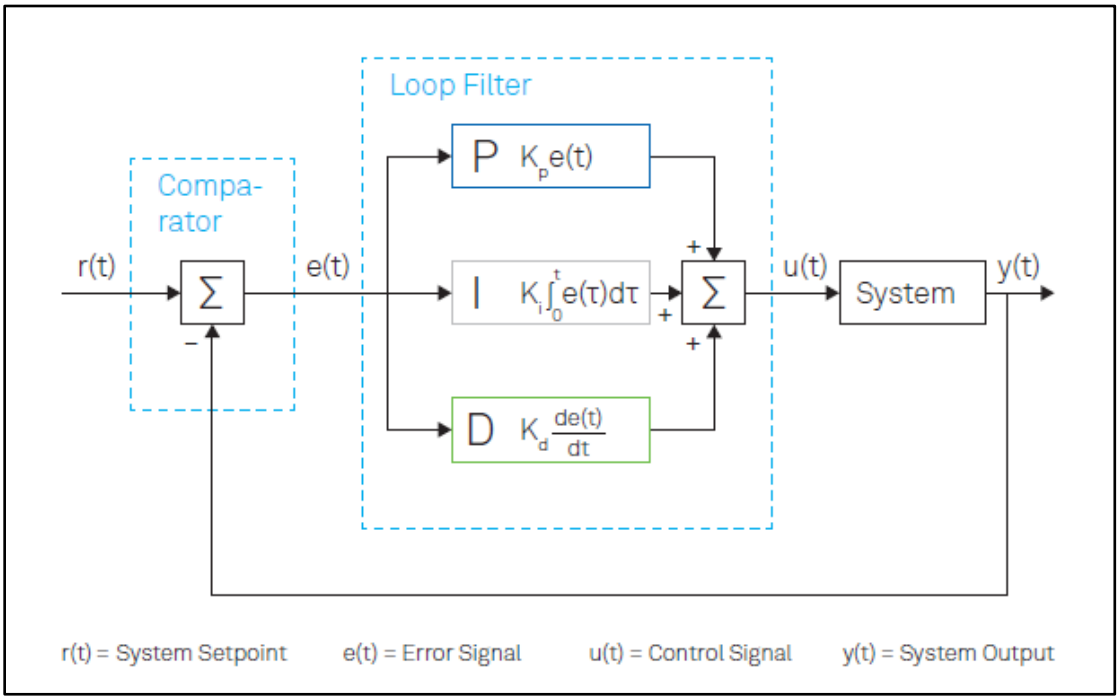
$$\boldsymbol{\theta} = asin\ (\frac{A_y}{\cos\ (Pitch)})$$

δίνει τα compensated x_h, y_h και τελικά από το τόξο εφαπτομένης τους το:

$$\boldsymbol{Tilt\ Compensated\ Heading} = \mathbf{arctan}(\frac{\boldsymbol{y_h}}{\boldsymbol{x_h}})$$ [4] [5] [6] [7] [8]

1.4 PID Controller

Ο έλεγχος σταθερότητας ενός Drone απαιτεί τις περισσότερες φορές την χρήση ενός ελεγκτή κλειστού κυκλώματος (closed loop controller) ο οποίος ονομάζεται PID controller. Ο PID controller ή αλλιώς Proportional – Integral – Derivative controller είναι ένας ελεγκτής που μετρά και προσαρμόζει την έξοδο ενός συστήματος σε ένα επιθυμητό σημείο (set point). Ο στόχος του ελεγκτή είναι να παράγει ένα σήμα ελέγχου το οποίο θα ελαχιστοποιήσει την διαφορά μεταξύ της τρέχουσας κατάστασης και του επιθυμητού σημείου. Στην εικόνα που ακολουθεί φαίνεται το μπλοκ διάγραμμα ενός τέτοιου συστήματος .

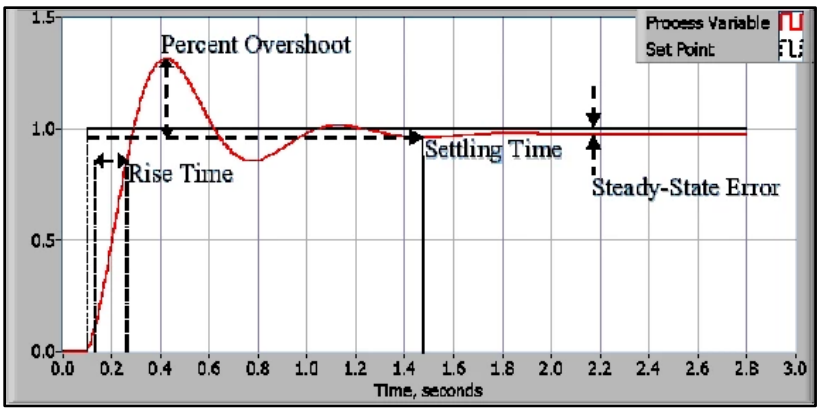


Εικόνα 12. Ελεγκτής κλειστού κυκλώματος ,PID closed loop controller [9]

Σε πρώτη φάση ή έξοδος y(t) οδηγείται στον συγκριτή ο οποίος συγκρίνει την έξοδο του συστήματος με την επιθυμητή τιμή και παράγει το λάθος e(t) (error) = r(t) – y(t). Στην συνέχεια αυτό το λάθος οδηγείται στο PID controller Loop Filter και παράγεται το σήμα ελέγχου u(t) οποίο τελικά οδηγείται στο σύστημα για να παράγει ξανά την νέα έξοδο κλείνοντας έτσι το loop. Αυτή η διαδικασία εκτελείται συνέχεια και έχει ως σκοπό να ελαχιστοποιήσει το λάθος (error). Ο PID controller αποτελείται από τρείς όρους όπως προαναφέρθηκε τον Proportional , τον Integral και τον Derivative. Αυτό μαθηματικά εκφράζεται από την σχέση:

$$u(t) = u_P(t) + u_I(t) + u_D(t) = K_P\, e(t) + K_I \int_0^t e(t)dt + K_D \frac{d}{dt} e(t)$$

όπου K_P, K_I και K_D είναι οι συντελεστές που αναφέρονται στο P, I και D αντίστοιχα.



Εικόνα 13. Χαρακτηριστικά απόκρισης PID controller

https://www.ni.com/en/shop/labview/pid-theory-explained.html

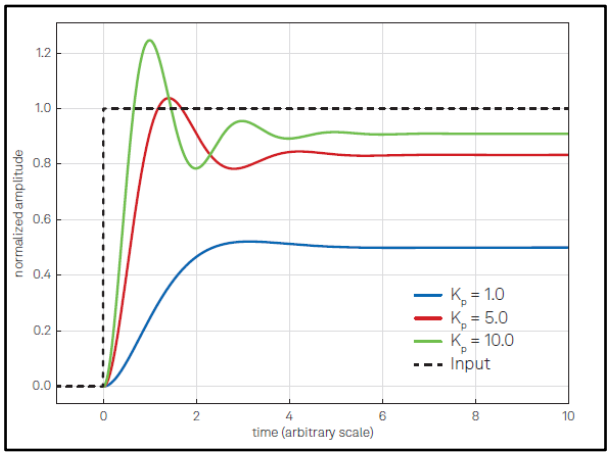
Στην εικόνα 13 φαίνεται η απόκριση ενός τυπικού PID συστήματος. Rise time είναι ο χρόνος που χρειάζεται το σύστημα για να μεταβεί από το 10% στο 90% της τελικής τιμής. Overshoot percent είναι το ποσό που η έξοδος του συστήματος υπερβαίνει την τελική τιμή, εκφραζόμενη ως ποσοστό της τελικής τιμής. Settling time ο χρόνος που απαιτείται ώστε η έξοδος του συστήματος να μεταβεί σε ένα ποσοστό της τελικής τιμής (συνήθως 5%). Steady – State error είναι η τελική διαφορά μεταξύ του επιθυμητού και του πραγματικού σημείου.

1.4.1 Όρος Proportional

Ο όρος Proportional που συμβολίζεται με το γράμμα P ενεργεί στο e(t) ή στο τρέχον σφάλμα μεταξύ του επιθυμητού και του πραγματικού σημείου εξόδου. Εφαρμόζει στο σύστημα μια διόρθωση που είναι ανάλογη του πλάτους του σφάλματος.

$$u\_P\ (t) = K\_P\ e(t)$$

Όσο μεγαλύτερο το σφάλμα τόσο μεγαλύτερη και η έξοδος του P όρου. Αφού η ύπαρξη του οφείλεται στο λάθος error γίνεται αντιληπτό ότι ο όρος αυτός, από μόνος του δεν μπορεί να μηδενίσει το λάθος.

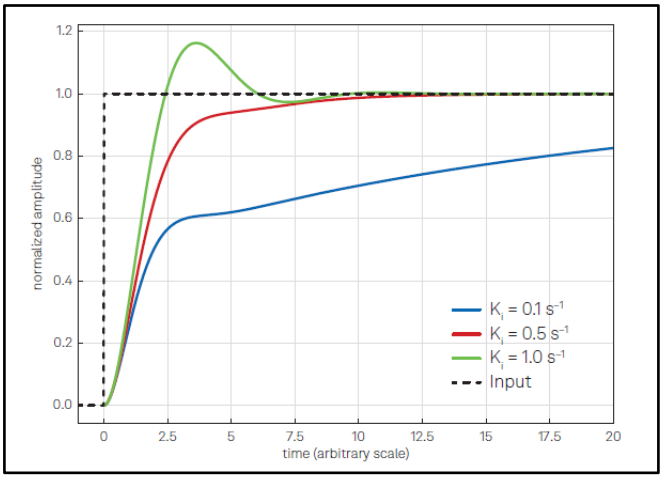


Εικόνα 14. Επίδραση Kp στην έξοδο του συστήματος [9]

Παρατηρούμε ότι καθώς αυξάνει η τιμή του K_P η έξοδος πλησιάζει το επιθυμητό σημείο. Επίσης για μεγάλη τιμή K_P υπάρχει περίπτωση το σύστημα να πέσει σε ταλάντωση.

1.4.2 Όρος Integral

Ο όρος Integral που συμβολίζεται με το γράμμα I ενεργεί στο e(t) πραγματοποιεί μια διόρθωση ανάλογη με το χρονικό ολοκλήρωμα του λάθους (error). Έχει χαρακτηριστικά ιστορικού, έτσι αν με την πάροδο του χρόνου το error παραμένει, η έξοδος του αυξάνεται και έχει την δυνατότητα σε αντίθεση με τον όρο P να μηδενίσει το λάθος.



Εικόνα 15. Επίδραση Ki στην έξοδο του συστήματος [9]

Εκφράζεται από την σχέση:

$$u_I(t) = K_I \int_0^t e(t)dt$$

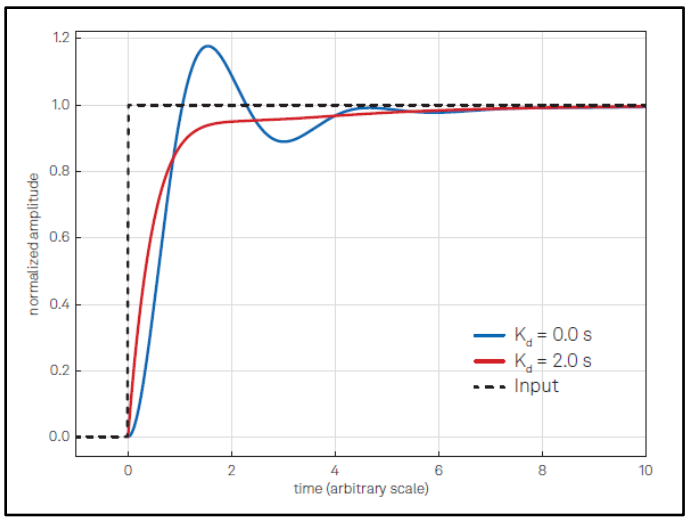
Αυξανόμενης της τιμής του ή έξοδος του όρου πλησιάζει στο επιθυμητό σημείο ενώ μεγάλη τιμή μπορεί να οδηγήσει σε overshoot και δημιουργία ταλαντώσεων γύρω από το σημείο αυτό.

1.4.3 Όρος Derivative

Ο όρος Derivative που συμβολίζεται με το γράμμα D παρέχει έναν έλεγχο πάνω στην ΄΄ τάση ΄΄ (tendency) λάθους, δηλαδή στην μελλοντική συμπεριφορά του συστήματος εφαρμόζοντας μια έξοδο ανάλογη με την χρονική παράγωγο του σφάλματος. Του δίνει την δυνατότητα να επιδράσει στην αλλαγή του λάθους με στόχο να το περιορίσει .

Εκφράζεται από την σχέση:

$$u_D(t) = K_D \frac{d}{dt} e(t)$$



Εικόνα 16. Επίδραση Kd στην έξοδο του συστήματος [9]

Καθώς αυξάνεται το σήμα εξόδου του όρου D το σύστημα τείνει να αντιδρά στις αλλαγές μειώνοντας το overshoot γύρο από το επιθυμητό σημείο[2].

Ο ελεγκτής PID συνίσταται από τρείς όρους , καθένας των οποίων έχει διαφορετική επίδραση στην συμπεριφορά ενός συστήματος. Οι τρείς όροι μπορούν είτε να μην χρησιμοποιηθούν όλοι , είτε να χρησιμοποιηθούν με διαφορετική σειρά . Η πιο απλή μορφή είναι αυτή του P controller ενώ υπάρχει περίπτωση να συναντήσουμε PD controller, PI controller ή τελικά PID controller. Για την περίπτωση του PID controller κάθε συντελεστής όρου φέρνει διαφορετικά αποτελέσματα στην ταχύτητα αποκατάστασης του συστήματος και στην ευστάθεια:

	Speed	Stability
K increases	increases	reduces
Ti increases	reduces	increases
Td increases	increases	increases

Πίνακας 2. Επίδραση PID συντελεστών στην Ταχύτητα και την Ευστάθεια

Advances in PID Control. Tan Kok Kiong, Wang Qing-Guo, Hang Chang

Η ρύθμιση του PID controller είναι μια πρόκληση. Πολλοί τρόποι υπάρχουν ώστε να τυποποιήσουν κατά κάποιο τρόπο αυτή την διαδικασία . Ο πιο γνωστός είναι αυτός των Ziegler-Nichols ή αλλιώς όπως λέγεται μέθοδος της συχνότητας. Σύμφωνα με αυτή τη μέθοδο :

- Μηδενίζουμε τους παράγοντες I και D.
- Αυξάνουμε το P gain ώστε το σύστημα να μεταβεί σε αυτοταλάντωση. Κατόπιν μετράμε την περίοδο της ταλάντωσης T_π
- Από την περίοδο της ταλάντωσης και το gain του P, K_π οι υπόλοιποι παράμετροι του ελεγκτή PID μπορούν να υπολογιστούν σύμφωνα με τον πίνακα 3.

Controller	K	Ti	Td
P	0,5Kπ	null	null
PI	0,4 Kπ	0,8 Tπ	null
PID	0,4 Kπ	0,5 To	0,25Tπ

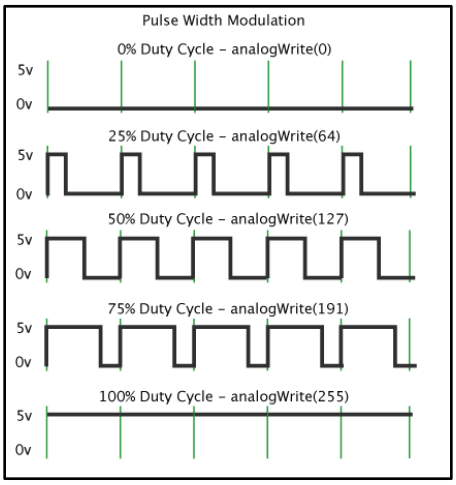
Πίνακας 3. Η μέθοδος των Ziegler-Nichols

Το μεγάλο πλεονέκτημα του τρόπου αυτού είναι η απλότητα της εφαρμογής , παρόλα αυτά όμως πολλές φορές οδηγεί σε έλλειψη σταθερότητας και οι τελικές τιμές των παραμέτρων ενδέχεται να είναι λίγο διαφορετικές [9] [10].

1.5 Pulse Width Modulation

Η Pulse Width Modulation, ή PWM, είναι μια τεχνική για τη λήψη αναλογικών πληροφοριών με ψηφιακά μέσα. Ο ψηφιακός έλεγχος χρησιμοποιείται για τη δημιουργία ενός τετραγωνικού σήματος, ενός σήματος εναλλαγής μεταξύ ενεργοποίησης και απενεργοποίησης. Αυτό το pattern on-off μπορεί να προσομοιώσει τις τάσεις μεταξύ του πλήρους Vcc (High) της πλακέτας και απενεργοποίησης (Low) αλλάζοντας το τμήμα του χρόνου που αφιερώνει το σήμα σε σχέση με το χρόνο που το σήμα σβήνει. Η διάρκεια του High time ονομάζεται πλάτος παλμού. Για να καταφέρουμε διάφορες αναλογικές τιμές, αλλάζουμε ή διαμορφώνουμε αυτό το πλάτος παλμού. Εάν επαναλάβουμε αυτό το pattern on-off αρκετά γρήγορα με ένα LED, για παράδειγμα, το αποτέλεσμα είναι σαν το σήμα να είναι μια σταθερή τάση μεταξύ 0 και Vcc που ελέγχει τη φωτεινότητα του LED.

Στο παρακάτω γράφημα, οι πράσινες γραμμές αντιπροσωπεύουν μια κανονική χρονική περίοδο. Αυτή η διάρκεια ή περίοδος είναι το αντίστροφο της συχνότητας PWM. Με άλλα λόγια, με τη συχνότητα PWM του Arduino στα 500 Hz περίπου, οι πράσινες γραμμές θα μετρούσαν 2 msec η καθεμία. Μια κλήση στην analogWrite() είναι σε κλίμακα 0 - 255, έτσι η analogWrite(255) ζητά έναν κύκλο λειτουργίας(duty cycle) 100% (πάντα ενεργοποιημένο) και για παράδειγμα το analogWrite(127) είναι ένας κύκλος λειτουργίας 50% (στο μισό χρόνο) για παράδειγμα.



Εικόνα 17. Pulse width Modulation (www.arduino.cc - Timothy Hirzel)

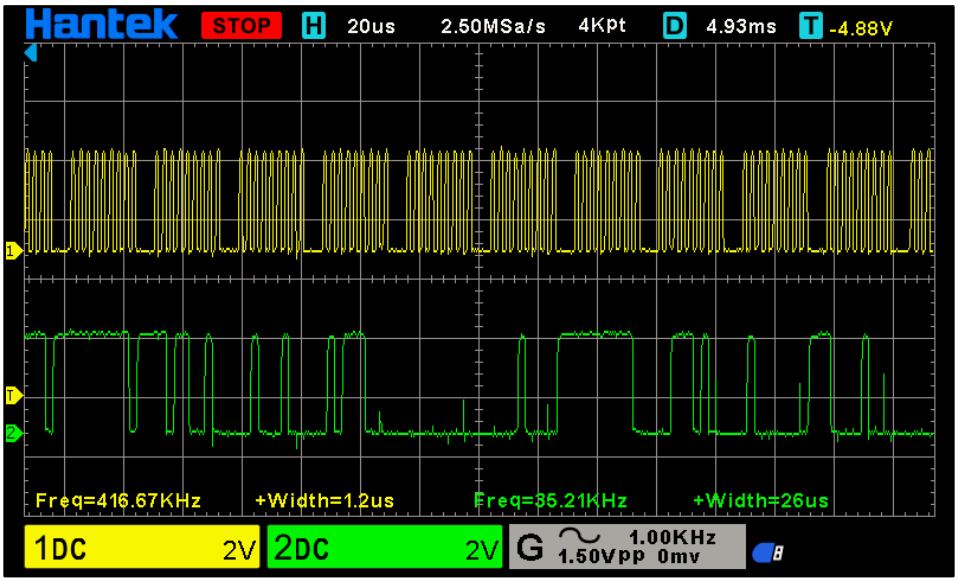
Στο Arduino Uno το PWM είναι διαθέσιμο μόνο σε επιλεγμένες εξόδους. Συμβολίζονται με ένα πρόσημο (~), αυτές είναι οι D3,D5,D6,D9,D10,D11 με τις D5,D6 να έχουν μέγιστο refresh rate 980 Hz ενώ οι υπόλοιπες να έχουν 490 Hz [11].

1.6 I2C protocol

Το πρωτόκολλο I2C (Inter Integrated Circuit) περιλαμβάνει τη χρήση δύο γραμμών για την αποστολή και τη λήψη δεδομένων: ένα pin σειριακού ρολογιού (SCL) που η πλακέτα ελεγκτή Arduino δημιουργεί τον παλμό σε τακτά χρονικά διαστήματα και ένα pin σειριακών δεδομένων (SDA) μέσω της οποίας αποστέλλονται δεδομένα μεταξύ των δύο συσκευών. Καθώς η γραμμή του clock αλλάζει από χαμηλή σε υψηλή (γνωστή ως ανερχόμενη άκρη του παλμού του clock – rising), ένα μόνο κομμάτι πληροφοριών - που θα σχηματίσει διαδοχικά τη διεύθυνση μιας συγκεκριμένης συσκευής και μια εντολή ή δεδομένα - μεταφέρεται από τον πίνακα στον τη συσκευή I2C πάνω από τη γραμμή SDA. Όταν αυτές οι πληροφορίες αποστέλλονται - bit μετά το bit -, η συσκευή που καλείται εκτελεί το αίτημα και μεταδίδει τα δεδομένα της πίσω - εάν απαιτείται - στην πλακέτα μέσω της ίδιας γραμμής χρησιμοποιώντας το σήμα clock που εξακολουθεί να παράγεται από τον ελεγκτή στο SCL ως χρονισμό.

Επειδή το πρωτόκολλο I2C επιτρέπει σε κάθε ενεργοποιημένη συσκευή να έχει τη δική της μοναδική διεύθυνση, και καθώς τόσο ο ελεγκτής όσο και οι περιφερειακές συσκευές επικοινωνούν εναλλάξ σε μία γραμμή, είναι δυνατό η πλακέτα Arduino να επικοινωνεί (με τη σειρά της) με πολλές συσκευές ή άλλες πλακέτες, ενώ χρησιμοποιείτε μόνο δύο pin του μικροελεγκτή σας.

Η προεπιλεγμένη ταχύτητα clock I2C είναι 100KHz και η μέγιστη ταχύτητα clock είναι 400KHz. Η πάνω κυματομορφή (κίτρινη) απεικονίζει το clock (SCL) ενώ η κάτω (πράσινη) τα δεδομένα (SDA) [12].



Εικόνα 18. Το πρωτόκολλο επικοινωνίας IIC

1.7 Interrupts

Το interrupt είναι ένα σήμα προς τον επεξεργαστή που εκπέμπεται από το Hardware ή το Software και υποδεικνύει ένα συμβάν που χρειάζεται άμεση προσοχή. Κάθε φορά που εμφανίζεται ένα interrupt, ο ελεγκτής ολοκληρώνει την εκτέλεση της τρέχουσας εντολής και ξεκινά την εκτέλεση μιας ρουτίνας υπηρεσίας διακοπής (ISR, Interrupt Service Routine) ή του Διαχειριστή Διακοπής (Interrupt Handler). Το ISR λέει στον επεξεργαστή ή στον ελεγκτή τι να κάνει όταν συμβεί η διακοπή. Υπάρχουν δύο ειδών Interrupts, τα Hardware Interrupts και τα Software Interrupts. Τα Hardware interrupts διαχειρίζονται με την INT και την PCINT.

1.7.1 Hardware Interrupts

INT

Ένα Hardware Interrupt είναι ένα ηλεκτρονικό σήμα ειδοποίησης που αποστέλλεται στον επεξεργαστή από μια εξωτερική συσκευή, όπως ένας ελεγκτής δίσκου ή μια εξωτερική περιφερειακή συσκευή. Για παράδειγμα, όταν πατάμε ένα πλήκτρο στο πληκτρολόγιο ή μετακινούμε το ποντίκι, προκαλούν Hardware Interrupts που αναγκάζουν τον επεξεργαστή να διαβάσει το πάτημα πλήκτρων ή τη θέση του ποντικιού.

Στο Arduino Uno, που είναι ο μικροελεγκτής που χρησιμοποιούμε υπάρχουν δύο Hardware Interrupts και αυτά απαντούν στα ψηφιακά pin 2,3. Η σύνταξη της εντολής που ενεργοποιεί ένα hardware interrupt είναι η ακόλουθη:

```
attachInterrupt(digitalPinToInterrupt(pin), ISR, mode)
```

όπου:

digitalPinToInterrupt(pin) : είναι το ψηφιακό pin που έχει αφιερωθεί για να εξυπηρετήσει το Interrupt. Στην περίπτωση μας είναι τα 2,3.

ISR : Interrupt Service Routine. Τα ISR είναι ειδικά είδη functions που έχουν κάποιους μοναδικούς περιορισμούς που δεν έχουν οι περισσότερες άλλες συναρτήσεις. Ένα ISR δεν μπορεί να έχει παραμέτρους και δεν πρέπει να επιστρέφει τίποτα.

Γενικά, ένα ISR πρέπει να είναι όσο το δυνατόν πιο σύντομο και γρήγορο. Εάν το πρόγραμμα χρησιμοποιεί πολλαπλούς ISR, μόνο ένα μπορεί να εκτελεστεί κάθε φορά, άλλα interrupts θα εκτελεστούν αφού τελειώσει το τρέχων ,με σειρά που εξαρτάται από την προτεραιότητα που έχουν. Η millis() βασίζεται σε interrupts για τη μέτρηση, επομένως δεν θα αυξάνεται ποτέ μέσα σε ένα ISR. Δεδομένου ότι η delay() απαιτεί interrupts για να λειτουργήσει, ούτε αυτή θα λειτουργήσει εάν καλείται μέσα σε ένα ISR. Η micros() λειτουργεί αρχικά, αλλά θα αρχίσει να συμπεριφέρεται ακανόνιστα μετά από 1-2 msec. Η delayMicroseconds() δεν χρησιμοποιεί κανένα μετρητή, επομένως θα λειτουργεί κανονικά.

Συνήθως οι καθολικές μεταβλητές (global variables)χρησιμοποιούνται για τη μετάδοση δεδομένων μεταξύ ενός ISR και του κύριου προγράμματος. Για να ενημερώνονται σωστά οι

μεταβλητές που μοιράζονται μεταξύ ενός ISR και του κύριου προγράμματος, αυτές πρέπει να δηλωθούν ως volatile.

mode : Καθορίζει πότε πρέπει να σκανδαλιστεί (triggered) ένα interrupt. Υπάρχουν 4 είδη mode που παρουσιάζονται παρακάτω :

LOW : το interrupt σκανδαλίζεται όταν το Pin είναι low,

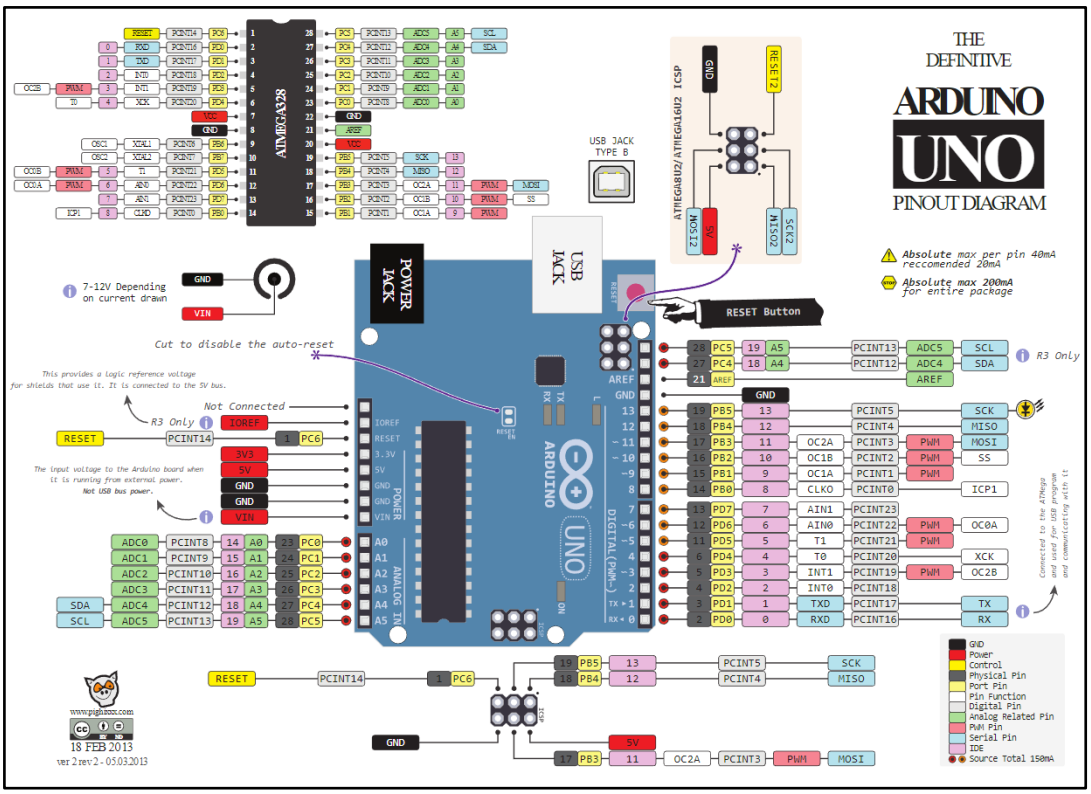
CHANGE : το interrupt σκανδαλίζεται όταν το Pin αλλάζει κατάσταση,

RISING : το interrupt σκανδαλίζεται όταν το Pin πηγαίνει από low σε high,

FALLING : το interrupt σκανδαλίζεται όταν το Pin πηγαίνει από high σε low.

1.7.2 PCINT

Τα interrupts PCINT δεν επενεργούν μόνο σε ένα Pin , αλλά σε μια ομάδα Pin πιο γνωστή ως port. Το Arduino UNO χρησιμοποιεί τον μικροελεγκτή ATmega328 ο οποίος έχει 3 θύρες, τη θύρα B, C και D. Έχουμε αρκετούς registers που ελέγχουν τα interrupts αυτών των θυρών. Το PCINT έχει επίσης ορισμένα μειονεκτήματα σε σύγκριση με το συνηθισμένο INT.SR και το κύριο πρόγραμμα. Έτσι, ενώ όπως προαναφέρθηκε τα INT μπορούν να σκανδαλιστούν (triggered) με LOW,CHANGE,RISING,FALLING τα PCINT σκανδαλίζονται μόνο με την κατάσταση CHANGE, είναι και ένας λόγος που είναι ελαφρώς πιο αργά στην απόκριση.



Εικόνα 19. The definitive Arduino Uno pinout www.pighixxx.com

Στην εικόνα 19 διακρίνονται οι τρεις θύρες (ports) , PB (IDE Digital Pin 8-13), PC (IDE Analog Pin A0 – A5) και PD (IDE Digital Pin 0 – 7).

Η παρακάτω εικόνα είναι ένα απόκομμα από το Datasheet του ATMEGA328p του επεξεργαστή που χρησιμοποιεί ο μικροελεγκτής Arduino Uno. Όπως φαίνεται αν κάποιος θέλει να χρησιμοποιήσει για Interrupt το physical Pin 13, IDE Digital Pin 7, θα πρέπει να κάνει την παρακάτω δήλωση στο void setup του IDE

PCIR |= B00000100 ;

όπως φαίνεται δίνοντας ΄΄1΄΄ στο bit 2 , ενεργοποιούμε το PCIE2 που είναι υπεύθυνο για τα physical pins 2-13 ή IDE digital pins 0-7 ή PCINT 16-23.

12.2.4 PCICR – Pin Change Interrupt Control Register

Bit	7	6	5	4	3	2	1	0	
(0x68)	–	–	–	–	–	PCIE2	PCIE1	PCIE0	PCICR
Read/Write	R	R	R	R	R	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7..3 - Res: Reserved Bits**

These bits are unused bits in the Atmel® ATmega328P, and will always read as zero.

- **Bit 2 - PCIE2: Pin Change Interrupt Enable 2**

When the PCIE2 bit is set (one) and the I-bit in the status register (SREG) is set (one), pin change interrupt 2 is enabled. Any change on any enabled PCINT23..16 pin will cause an interrupt. The corresponding interrupt of pin change interrupt request is executed from the PCI2 interrupt vector. PCINT23..16 pins are enabled individually by the PCMSK2 register.

- **Bit 1 - PCIE1: Pin Change Interrupt Enable 1**

When the PCIE1 bit is set (one) and the I-bit in the status register (SREG) is set (one), pin change interrupt 1 is enabled. Any change on any enabled PCINT14..8 pin will cause an interrupt. The corresponding interrupt of pin change interrupt request is executed from the PCI1 interrupt vector. PCINT14..8 pins are enabled individually by the PCMSK1 register.

- **Bit 0 - PCIE0: Pin Change Interrupt Enable 0**

When the PCIE0 bit is set (one) and the I-bit in the status register (SREG) is set (one), pin change interrupt 0 is enabled. Any change on any enabled PCINT7..0 pin will cause an interrupt. The corresponding interrupt of pin change interrupt request is executed from the PCI0 interrupt vector. PCINT7..0 pins are enabled individually by the PCMSK0 register.

Εικόνα 20. Pin Change Interrupt Control Register, www.arduino.cc

Στην συνέχεια πρέπει να δηλωθεί ποιο pin από την PCIE2 θα είναι αυτό που θα σκανδαλίσει το ISR. Έτσι όπως φαίνεται στην εικόνα που ακολουθεί, θα πρέπει να γίνει η παρακάτω δήλωση στο void setup:

PCMSK2 |= B10000000 ;

12.2.6 PCMSK2 – Pin Change Mask Register 2

Bit	7	6	5	4	3	2	1	0	
(0x6D)	PCINT23	PCINT22	PCINT21	PCINT20	PCINT19	PCINT18	PCINT17	PCINT16	PCMSK2
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7..0 – PCINT23..16: Pin Change Enable Mask 23..16**

Each PCINT23..16-bit selects whether pin change interrupt is enabled on the corresponding I/O pin. If PCINT23..16 is set and the PCIE2 bit in PCICR is set, pin change interrupt is enabled on the corresponding I/O pin. If PCINT23..16 is cleared, pin change interrupt on the corresponding I/O pin is disabled.

Εικόνα 21. Pin Change Mask Register, www.arduino.cc

Τέλος όσο αφορά τα PCINT θα πρέπει στην ISR να αναφερθεί το vector το οποίο αντιστοιχεί στον pin ενδιαφέροντος , έτσι η παρακάτω δήλωση θα πρέπει να γίνει:

ISR (PCINT2_vect)

όπου PCINT0_vect, IDE digital pins 8-13

PCINT1_vect, IDE analog pins A0-A5

PCINT2_vect, IDE digital pins 0-7 [13]

1.7.3 Software Interrupts

Προκύπτουν ως απόκριση σε μια οδηγία που αποστέλλεται στο software. Ο μόνος τύπος interrupt που υποστηρίζει η "γλώσσα Arduino" είναι η συνάρτηση attachInterrupt().

Σε κάθε περίπτωση όταν το πρόγραμμα εκτελεί μια ρουτίνα η οποία δεν πρέπει να διακοπεί για κανένα λόγο η "γλώσσα Arduino" μας δίνει την δυνατότητα να απενεργοποιήσουμε τα interrupts:

```
void setup() {}

void loop() {
  noInterrupts();
  // critical, time-sensitive code here
  interrupts();
  // other code here
}
```

Εικόνα 22. Εντολή noInterrupts – time sensitive loop

1.8 Analog to Digital Converter

Οι ελεγκτές ATmega που χρησιμοποιούνται για το Arduino περιέχουν έναν ενσωματωμένο μετατροπέα 6 καναλιών (A0 – A5) αναλογικό σε ψηφιακό (A/D) μετατροπέα. Ο μετατροπέας έχει ανάλυση 10 bit, επιστρέφοντας ακέραιους αριθμούς από το 0 έως το 1023 (2^10). Ενώ η κύρια λειτουργία των αναλογικών ακίδων για τους περισσότερους χρήστες Arduino είναι η ανάγνωση αναλογικών αισθητήρων, τα αναλογικά pin διαθέτουν επίσης όλες τις λειτουργίες των ακροδεκτών εισόδου/εξόδου γενικής χρήσης (GPIO - το ίδιο με τις ψηφιακές ακίδες 0 - 13).

1.9 Weight and Balance

Τα πτητικά χαρακτηριστικά ενός τετρακόπτερου εξαρτώνται αποκλειστικά από το βάρος και την κατανομή αυτού στο σκάφος (Weight and Balance). Το συνολικό βάρος και το κέντρο βάρους διαδραματίζουν σημαντικό ρόλο στην αξιοπλοΐα του τετρακόπτερου σχετικά με τις επιδόσεις του, την ευστάθεια και τον έλεγχο του. Για παράδειγμα βάρος πολύ έξω από το επιτρεπτό όριο του κέντρου βάρους θα έχει σαν αποτέλεσμα την διαρκή κλίση είτε pitch είτε roll με υπαρκτή την πιθανότητα το thrust των μοτέρ να μην μπορούν να την διορθώσουν και να χαθεί ο έλεγχος του σκάφους. Γενικά πολύ βάρος ή/και το κέντρο βάρος έξω από τα προκαθορισμένα όρια οδηγεί αναμφισβήτητα σε μείωση απόδοσης (performance degradation) ή ικανότητας ανύψωσης φορτίου (payload reduction). Έτσι υπό αυτές τις συνθήκες έχουμε:

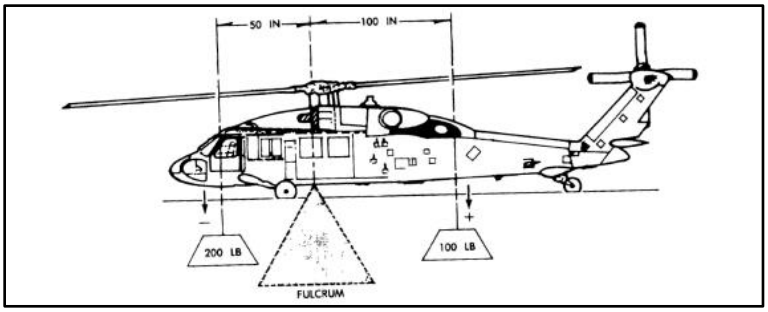
- Αστάθεια Pitch
- Αστάθεια Roll
- Μειωμένη απόδοση hover
- Μειωμένη απόδοση ελιγμών
- Μειωμένη απόδοση ταχύτητας πτήσης
- Μειωμένη απόδοση ρυθμού ανόδου
- Δομικά θέματα
- Αυξημένη κατανάλωση ενέργειας
- Μειωμένη διάρκεια πτήσης

Ένα τετρακόπτερο λέγεται ότι βρίσκεται σε ισορροπία ή ισορροπεί όταν όλα του τα βάρη είναι κατανεμημένα έτσι ώστε το κέντρο βάρος του να είναι μέσα στα προδιαγραμμένα όρια. Αυτά ορίζονται ως το μέγιστο μπροστά όριο, μέγιστο πίσω, μέγιστο αριστερά και μέγιστο δεξιά.

Για να κατανοηθεί η έννοια του CG πρέπει να εισαχθούν οι παρακάτω όροι:

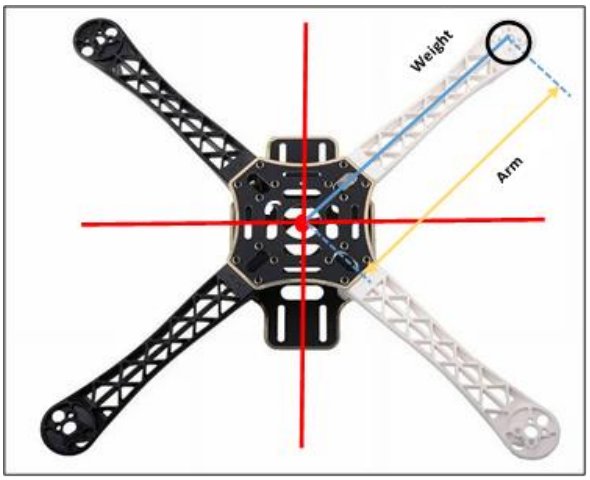
- Weight: το βάρος του υπό εγκατάσταση υλικού
- Arm: η απόσταση του υλικού αυτού από το CG
- Moment: το γινόμενο Weight X Arm

Δύο υλικά που έχουν το ίδιο Moment και βρίσκονται πάνω στον ίδιο άξονα αντιδιαμετρικά δεν αλλάζουν το κέντρο βάρους CG ενός σκάφους [14].



Εικόνα 23. Weight & Balance moments [14]

Moment1 = 50 in X 200lb
Moment2 = 100 in X 100lb
Moment1 = Moment2



Εικόνα 24. F450 πλαίσιο , Weight – Arm - CG

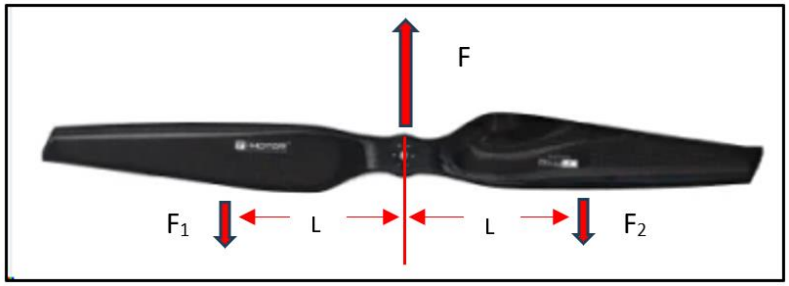
Πολλές φορές λόγοι σχεδιαστικοί, κατασκευαστικοί ή τυχόν τροποποιήσεις μπορούν να μεταφέρουν το CG εκτός της προδιαγεγραμμένης περιοχής. Για τον λόγο αυτό απαιτείται η χρήση Ballasts (έρμα). Είναι μόνιμες ή προσωρινές εγκαταστάσεις βαρών σε συγκεκριμένο Arm έτσι ώστε να ρυθμίσουν ή να μετατοπίσουν το CG στο επιθυμητό σημείο. Συνήθως βρίσκονται μακριά από το CG ώστε να επιτύχουν το μέγιστο δυνατό Moment με το ελάχιστο δυνατό βάρος μειώνοντας την επίδραση στο συνολικό βάρος του σκάφους. Μπορεί να είναι από διάφορα υλικά όπως μέταλλο, λάστιχο, άμμο κ.α.

Εικόνα 25. Weight & Balance , Ballasts

1.10 Propeller Balancing

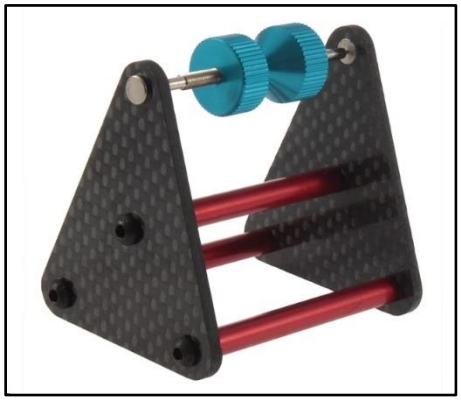
Η σωστή εξισορρόπηση της προπέλας (propeller balancing) είναι σημαντική για την έγκαιρη διόρθωση των προβλημάτων κραδασμών, συμβάλλοντας στην παράταση της διάρκειας ζωής των εξαρτημάτων όπως ο κινητήρας (bearings, rotation shaft) ή το πλαίσιο (frame) του τετρακόπτερου. Υπάρχουν δύο κύριοι τύποι ζυγοστάθμισης προπέλας: στατική και δυναμική.

Η στατική εξισορρόπηση προπέλας έχει να κάνει με το να φέρει την προπέλα σε ισορροπία με τον εαυτό της, έτσι ώστε το ένα blade να μην είναι βαρύτερο ή ελαφρύτερο από τις άλλο.



Εικόνα 26. Propeller Balancing - Static

Μια σωστά ζυγισμένη προπέλα εκτός των άλλων δεν εισάγει θόρυβο στο επιταχυνσιόμετρο του IMU (Inertial Measurement Unit) που όπως είναι γνωστό είναι ευαίσθητο στους κραδασμούς. Αυτό έχει σαν αποτέλεσμα την ομαλή πτήση του τετρακόπτερου.



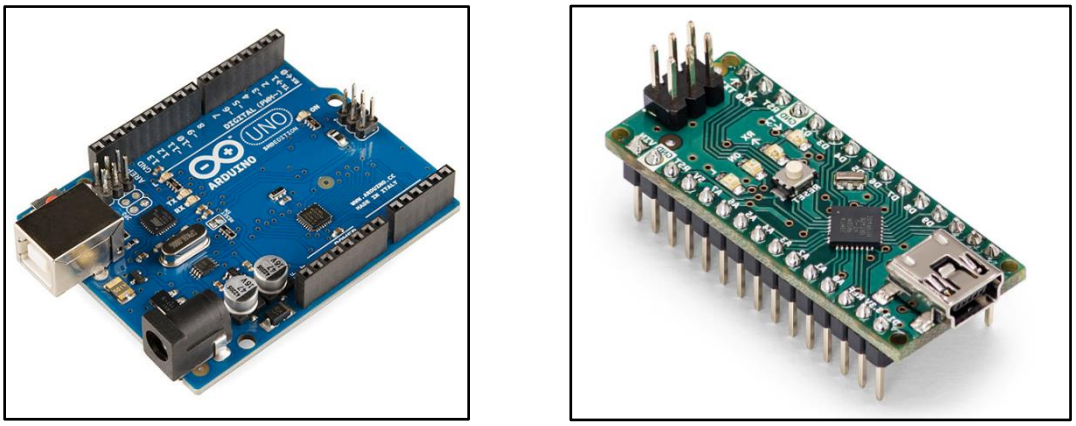
Εικόνα 27. Propeller Balancer

Η δυναμική εξισορρόπηση προπέλας έχει να κάνει με την διαδικασία που εκτελείται κατά την διάρκεια λειτουργείας του σκάφους. Πραγματοποιείται με την εγκατάσταση επιταχυνσιομέτρων τα οποία μετρούν το μέγεθος των κραδασμών σε IPS (inches per second) .

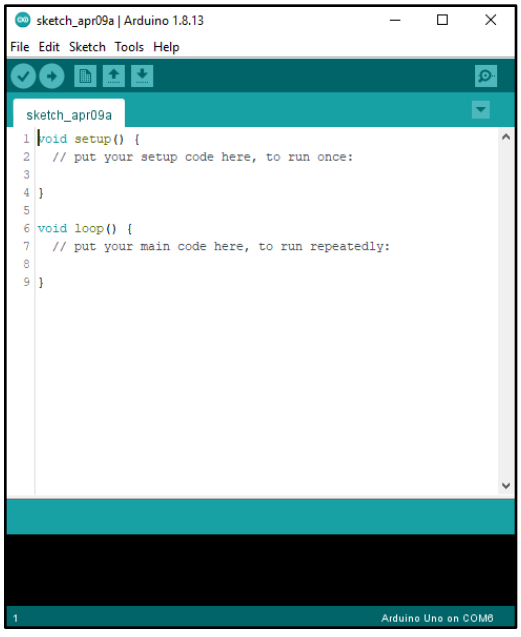
2. ΕΠΙΣΚΟΠΗΣΗ ΥΛΙΚΟΥ ΜΕΡΟΥΣ (HARDWARE)

2.1 Arduino

Το Arduino Uno είναι μια πλακέτα μικροελεγκτή open source που βασίζεται στον μικροελεγκτή Microchip ATmega328P και αναπτύχθηκε από την Arduino.cc και κυκλοφόρησε αρχικά το 2010. Η πλακέτα είναι εξοπλισμένη με σετ ψηφιακών και αναλογικών pin εισόδου/εξόδου (I/O). Το Arduino Uno έχει 14 ψηφιακά pin εισόδου/εξόδου (έξι με δυνατότητα εξόδου PWM), 6 αναλογικά pin εισόδου/εξόδου και μπορεί να προγραμματιστεί με το Arduino IDE (Integrated Development Environment), μέσω ενός καλωδίου USB τύπου. Μπορεί να τροφοδοτηθεί από ένα καλώδιο USB ή μια υποδοχή mini jack που δέχεται τάσεις μεταξύ 7 και 20 Volt.



Εικόνα 28. Arduino Uno, Nano



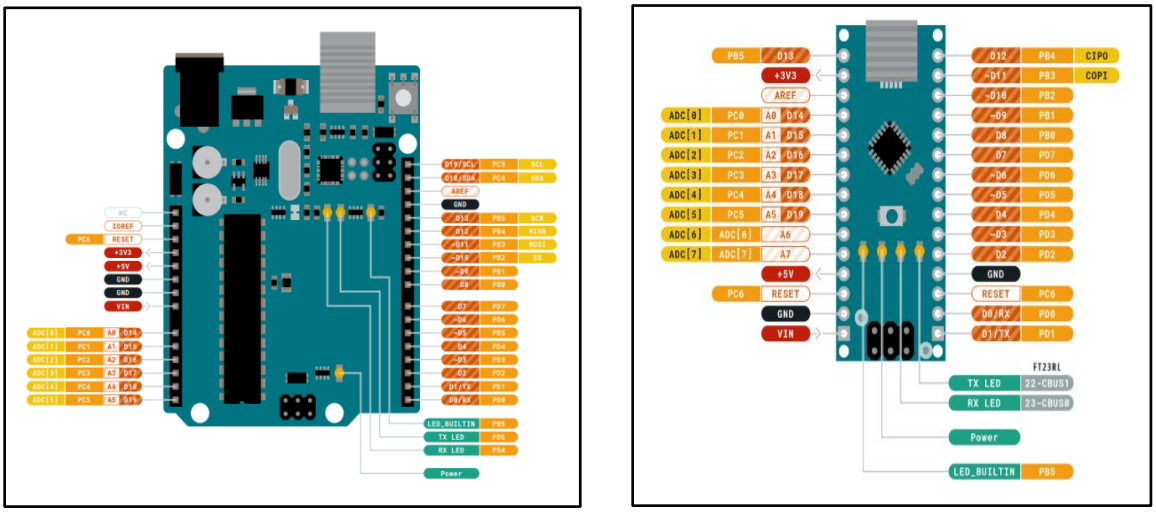
Το περιβάλλον προγραμματισμού του Arduino. Το λογισμικό open source Arduino (IDE) διευκολύνει τη σύνταξη κώδικα και τη φόρτωσή του στον πίνακα. Αυτό το λογισμικό μπορεί να χρησιμοποιηθεί με οποιαδήποτε πλακέτα Arduino.

Εικόνα 29. Arduino IDE

Η γλώσσα προγραμματισμού Arduino IDE είναι ένα framework που βασίζεται στη γλώσσα C++. Το IDE μεταγλωττίζει τον κώδικα C++ στη γλώσσα assembly, η οποία χρησιμοποιείται από το chip της Atmel που είναι τοποθετημένο στην πλακέτα Arduino Uno. Η γλώσσα Arduino βασίζεται σε μια απλοποιημένη έκδοση της γλώσσας C και C++ που την καθιστά ευκολότερη και πιο προσαρμόσιμη για αρχάριους.

2.1.1 Arduino Uno / Nano Pinout

Στις εικόνες που ακολουθούν φαίνεται το Pinout και περιγράφονται όλες οι είσοδοι/έξοδοι του Arduino Uno / Nano [15] [16] :



Εικόνα 30. Arduino Uno - Nano Pinout [15],[16]

Αναλογικό Μέρος			
n	Function	Type	Description
1	NC	NC	Not connected
2	IOREF	IOREF	Reference for digital logic - connected to 5V
3	Reset	Reset	Reset
4	+3V3	Power	+3V3 Power rail
5	+5V	Power	+5V Power rail
6	GND	Power	Ground
7	GND	Power	Ground
8	VIN	Power	Voltage Input
9	A0	Analog/GPIO	Analog input 0 /GPIO
10	A1	Analog/GPIO	Analog input 1 /GPIO
11	A2	Analog/GPIO	Analog input 2 /GPIO
12	A3	Analog/GPIO	Analog input 3 /GPIO
13	A4/SDA	Analog Input/I2C	Analog input 4/I2C Data line
14	A5/SCL	Analog Input/I2C	Analog input 5/I2C Clock line

Πίνακας 4. Αναλογικές είσοδοι / έξοδοι

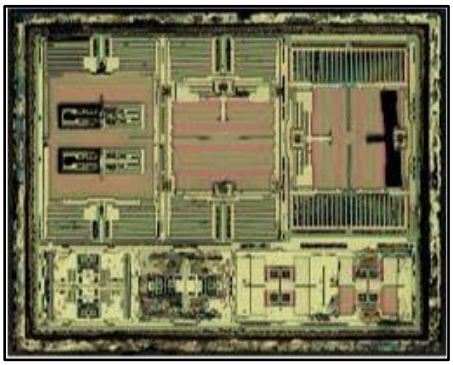
Ψηφιακό Μέρος			
n	Function	Type	Description
1	D0	Digital/GPIO	Digital pin 0/GPIO
2	D1	Digital/GPIO	Digital pin 1/GPIO
3	D2	Digital/GPIO	Digital pin 2/GPIO
4	D3	Digital/GPIO	Digital pin 3/GPIO
5	D4	Digital/GPIO	Digital pin 4/GPIO
6	D5	Digital/GPIO	Digital pin 5/GPIO
7	D6	Digital/GPIO	Digital pin 6/GPIO
8	D7	Digital/GPIO	Digital pin 7/GPIO
9	D8	Digital/GPIO	Digital pin 8/GPIO
10	D9	Digital/GPIO	Digital pin 9/GPIO
11	SS	Digital	SPI Chip select
12	MOSI	Digital	SPI1 Main Out Secondary In
13	MISO	Digital	SPI Main In Secondary Out
14	SCK	Digital	SPI serial clock output
15	GND	Power	Ground
16	AREF	Digital	Analog reference voltage
17	A4/SD4	Analog Input/I2C	Analog input 4/I2C Data line(duplicated)
18	A5/SD5	Analog Input/I2C	Analog input 5/I2C Clock line(duplicated)

Πίνακας 5. Ψηφιακές είσοδοι /έξοδοι

2.2 MPU 6050

Το MPU-6050 είναι η ενσωματωμένη συσκευή MotionTracking 6 αξόνων που συνδυάζει ένα γυροσκόπιο 3 αξόνων, ένα επιταχυνσιόμετρο 3 αξόνων και έναν ψηφιακό επεξεργαστή κίνησης™ (DMP) όλα σε μια μικρή συσκευασία 4x4x0,9 mm. Βασισμένο στην τεχνολογία I2C, δέχεται απευθείας εισόδους από μια εξωτερική πυξίδα 3 αξόνων για να παρέχει μια πλήρη έξοδο MotionFusion™ 9 αξόνων. Η συσκευή MPU-6050 MotionTracking, με την ενσωμάτωση 6 αξόνων, το on-board MotionFusion™ και το υλικό- λογισμικό βαθμονόμησης χρόνου εκτέλεσης, επιτρέπει στους κατασκευαστές να εξαλείψουν την περίπλοκη ενσωμάτωση διακριτών συσκευών, την πιστοποίηση και το επίπεδο συστήματος, διασφαλίζοντας τη βέλτιστη απόδοση. Το MPU-6050 έχει επίσης σχεδιαστεί για διασύνδεση με πολλούς μη αδρανειακούς ψηφιακούς αισθητήρες, όπως αισθητήρες πίεσης, στη βοηθητική θύρα I2C.

Το MPU-6050 διαθέτει τρεις μετατροπείς αναλογικού σε ψηφιακό 16-bit (ADC) για την ψηφιοποίηση των εξόδων του γυροσκοπίου (gyro) και τρεις ADC 16-bit για την ψηφιοποίηση των εξόδων του επιταχυνσιόμετρου(accelerometer). Για την ακριβή παρακολούθηση γρήγορων και αργών κινήσεων, τα εξαρτήματα διαθέτουν εύρος πλήρους κλίμακας γυροσκόπιο προγραμματιζόμενο από το χρήστη ±250, ±500, ±1000 και ±2000°/sec (dps) και ένα επιταχυνσιόμετρο πλήρους κλίμακας προγραμματιζόμενο από το χρήστη με εύρος ±2g, ±4g, ±8g και ±16g.

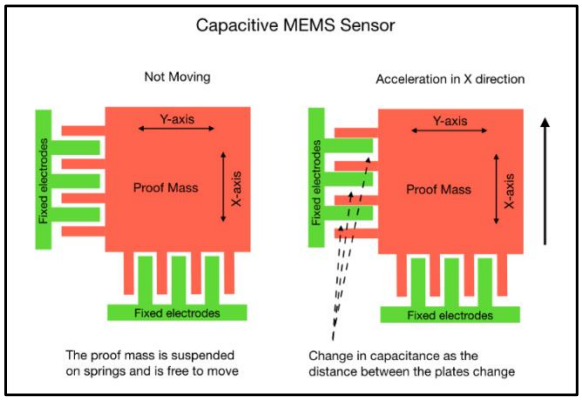


Εικόνα 31. Το μηχανικό μέρος του αισθητήρα, MPU-6050

https://mjwhite8119.github.io/Robots/mpu6050

Μια προσωρινή μνήμη FIFO 1024 Byte στο chip βοηθά στη μείωση της κατανάλωσης ενέργειας του συστήματος, επιτρέποντας στον επεξεργαστή του συστήματος να διαβάζει τα δεδομένα του αισθητήρα (burst) και στη συνέχεια να εισέρχεται σε λειτουργία χαμηλής κατανάλωσης καθώς η MPU συλλέγει περισσότερα δεδομένα. Με όλα τα απαραίτητα στοιχεία επεξεργασίας που απαιτούνται για την υποστήριξη πολλών περιπτώσεων χρήσης που βασίζονται στην κίνηση, το MPU-60X0 επιτρέπει μοναδικά εφαρμογές MotionInterface χαμηλής κατανάλωσης σε φορητές εφαρμογές με μειωμένες απαιτήσεις επεξεργασίας για τον επεξεργαστή συστήματος. Παρέχοντας μια ενσωματωμένη έξοδο MotionFusion, το DMP στο MPU-6050 αποφορτίζει τις απαιτητικές διεργασίες υπολογισμού MotionProcessing από τον κεντρικό επεξεργαστή του συστήματος, ελαχιστοποιώντας την ανάγκη για συχνή μέτρηση της εξόδου του αισθητήρα κίνησης. Η επικοινωνία με όλους τους καταχωρητές της συσκευής πραγματοποιείται χρησιμοποιώντας είτε I2C στα 400kHz.

Αξιοποιώντας την πατέντα της Nasiri-Fabrication, η οποία ενσωματώνει wafers MEMS (Micro Electro Mechanical Systems) με συνοδευτικά ηλεκτρονικά CMOS μέσω συγκόλλησης σε επίπεδο wafer, η InvenSense έχει μειώσει το μέγεθος συσκευασίας MPU-6050 σε επαναστατικό αποτύπωμα 4x4x0,9 mm (QFN), ενώ παρέχοντας την υψηλότερη απόδοση, το χαμηλότερο θόρυβο και τη χαμηλότερη συσκευασία ημιαγωγών που απαιτείται για φορητές ηλεκτρονικές συσκευές ευρείας κατανάλωσης.



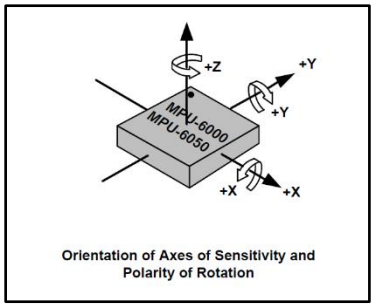
Εικόνα 32. MEMS (Micro Electro Mechanical Systems), MPU-6050 [18]

Το εξάρτημα διαθέτει στιβαρή ανοχή κραδασμών 10.000 g και έχει προγραμματιζόμενα φίλτρα χαμηλής διέλευσης (**D**igital **L**ow **P**ass **F**ilter)για τα γυροσκόπια, τα επιταχυνσιόμετρα και τον αισθητήρα θερμοκρασίας στο τσιπ.

Για ευελιξία τροφοδοσίας, το MPU-6050 λειτουργεί από εύρος τάσης τροφοδοσίας VDD 2,375V-3,46V. Επιπλέον, το MPU-6050 παρέχει ένα pin αναφοράς VLOGIC (επιπλέον του αναλογικού pin τροφοδοσίας: VDD), η οποία ορίζει τα λογικά επίπεδα της διεπαφής I2C. Η τάση VLOGIC μπορεί να είναι 1,8V±5% ή VDD.

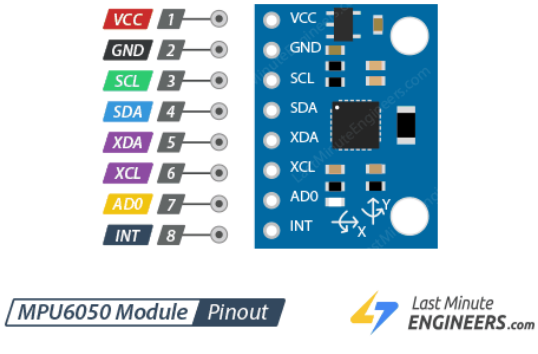
Το MPU-6050 περιέχει έναν καταχωρητή FIFO 1024 byte που είναι προσβάσιμος μέσω της σειριακής διεπαφής. Ο καταχωρητής διαμόρφωσης FIFO καθορίζει ποια δεδομένα εγγράφονται στο FIFO. Οι πιθανές επιλογές περιλαμβάνουν δεδομένα γυροσκοπίου, δεδομένα επιταχυνσιόμετρου, μετρήσεις θερμοκρασίας, μετρήσεις βοηθητικού αισθητήρα και είσοδο FSYNC. Ένας μετρητής FIFO παρακολουθεί πόσα byte έγκυρων δεδομένων περιέχονται στο FIFO. Ο καταχωρητής FIFO υποστηρίζει ριπές αναγνώσεις (burst reads). Η συνάρτηση διακοπής (INT) μπορεί να χρησιμοποιηθεί για τον προσδιορισμό του πότε είναι διαθέσιμα νέα δεδομένα.

Το παρακάτω διάγραμμα δείχνει τον προσανατολισμό των αξόνων ευαισθησίας και την πολικότητα περιστροφής. Αξιοσημείωτο είναι το pin αναγνωριστικό (•) στο σχήμα [17] [18].



Εικόνα 33. Προσανατολισμός αξόνων περιστροφής, MPU-6050 [17]

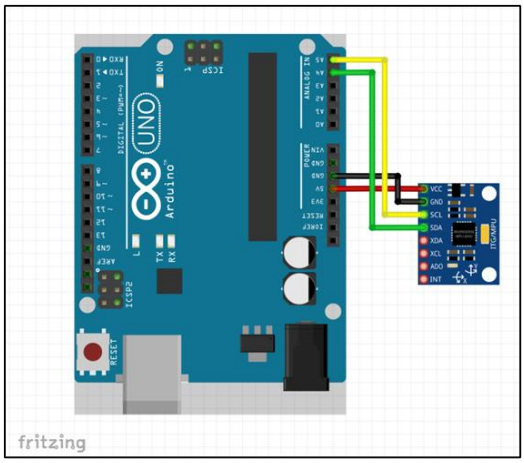
Το MPU-6050 είναι διαθέσιμο σε πολλές διαφορετικές διαμορφώσεις και σε πολλές διαφορετικές μονάδες . Ακολουθεί μια απεικόνιση των pin του module που χρησιμοποιούμε:



Εικόνα 34. Το MPU- 6050 Module / Pinout

https://lastminuteengineers.com/mpu6050-accel-gyro-arduino-tutorial/

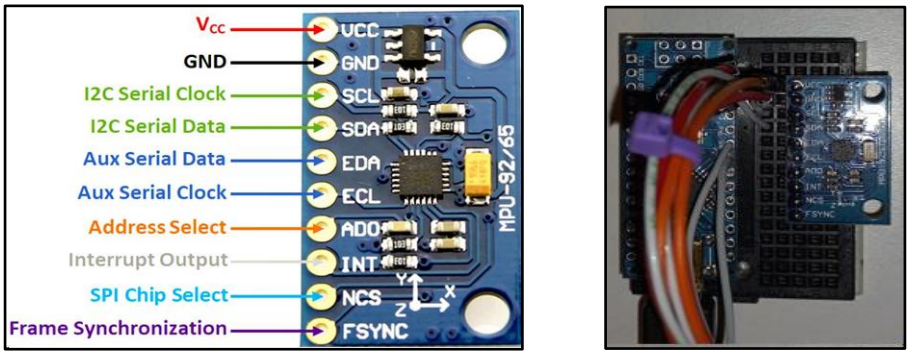
Το MPU-6050 συνδέεται με το Arduino όπως φαίνεται στο σχήμα που ακολουθεί, όπου SCL στην θέση A5 και SDA στην θέση A4:



Εικόνα 35. Συνδεσμολογία Arduino Uno με το module MPU 6050 , 6 DOF

2.3 MPU 9250

Το MPU-9250 της εταιρίας TDK , είναι το δεύτερης γενιάς προϊόν, 9 βαθμών ελευθερίας IMU (Inertial Measurement Unit). Η απόδοση όσο αφορά το γυροσκόπιο του είναι κατά 3 φορές καλλίτερη σε σχέση με τον θόρυβο ενώ η απόδοση της ενσωματωμένης πυξίδας είναι πάνω από 4 φορές καλύτερη από τις ανταγωνισμό. Το MPU-9250 συνδυάζει δύο τσιπ: το MPU-6500, το οποίο περιέχει ένα γυροσκόπιο 3 αξόνων, ένα επιταχυνσιόμετρο 3 αξόνων και έναν ενσωματωμένο Digital Motion Processor™ και το AK8963, την κορυφαία ψηφιακή πυξίδα 3 αξόνων. Το MPU-9250 υποστηρίζει επίσης το MotionFusion της InvenSense.



Εικόνα 36. Το MPU-9250

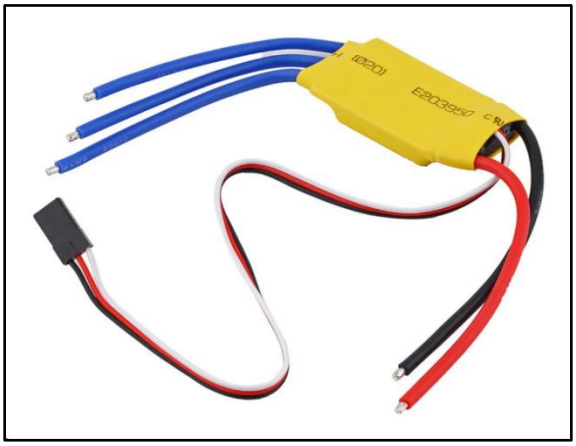
https://components101.com/sensors/MPU9250-9-dof-mems-sensor-module-datasheet-pinout-features-working

Οι συνδέσεις του με το Arduino σε συνδεσμολογία IIC είναι ακριβώς οι ίδιες με MPU-6050.

2.4 ESC (Electronic speed Controller)

Ένας electronic speed controller (ESC) είναι ένα ηλεκτρονικό κύκλωμα που ελέγχει και ρυθμίζει την ταχύτητα ενός ηλεκτροκινητήρα. Παρέχει επίσης αναστροφή του κινητήρα και δυναμική πέδηση.

Ένας ESC ακολουθεί ένα σήμα αναφοράς ταχύτητας (που προέρχεται από μοχλό γκαζιού, joystick ή άλλη χειροκίνητη είσοδο) και μεταβάλλει τον ρυθμό μεταγωγής ενός δικτύου τρανζίστορ εφέ πεδίου (FETs). Ρυθμίζοντας τον κύκλο λειτουργίας (duty cycle) ή τη συχνότητα μεταγωγής(switching frequency) των τρανζίστορ, αλλάζει η ταχύτητα του κινητήρα.



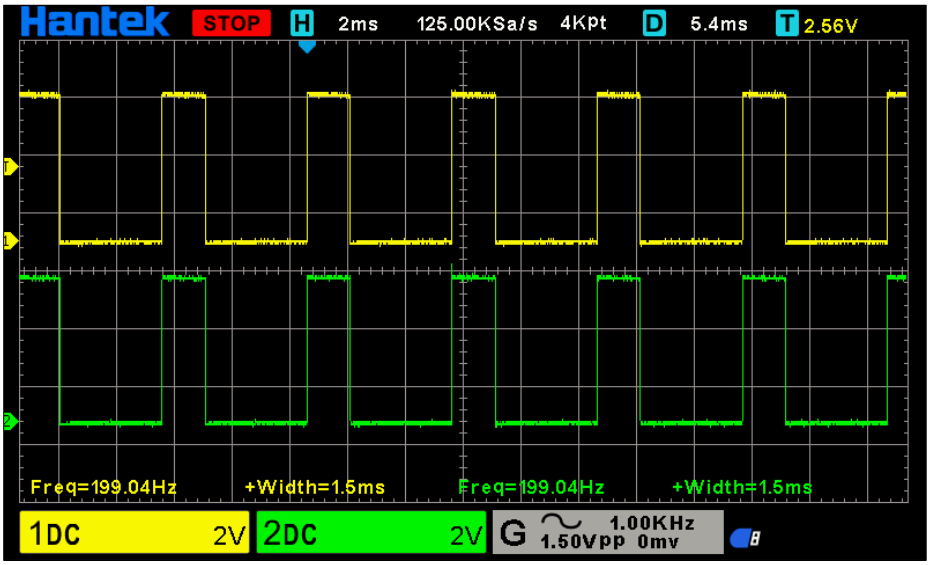
Εικόνα 37. Ο ESC (Electronic Speed Controller) που χρησιμοποιείται στην κατασκευή.

Απαιτούνται διαφορετικοί τύποι ελέγχων ταχύτητας για κινητήρες συνεχούς ρεύματος brushed και κινητήρες συνεχούς ρεύματος brushless. Ένας brushed κινητήρας μπορεί να ελέγχει την ταχύτητά του μεταβάλλοντας την τάση στον οπλισμό του. (Βιομηχανικά, οι κινητήρες με περιελίξεις ηλεκτρομαγνητικού πεδίου αντί για μόνιμους μαγνήτες μπορούν επίσης να ρυθμίσουν την ταχύτητά τους ρυθμίζοντας την ένταση του ρεύματος του πεδίου του κινητήρα.) Ένας κινητήρας χωρίς ψήκτρες (brushless) απαιτεί διαφορετική αρχή λειτουργίας. Η ταχύτητα του κινητήρα μεταβάλλεται ρυθμίζοντας το χρονισμό των παλμών ρεύματος που παρέχονται στις διάφορες περιελίξεις του κινητήρα.

Τα συστήματα ESC χωρίς ψήκτρες (brushless) βασικά δημιουργούν τριφασική ισχύ εναλλασσόμενου ρεύματος, όπως μια μονάδα μεταβλητής συχνότητας, για να λειτουργούν κινητήρες χωρίς ψήκτρες. Οι κινητήρες χωρίς ψήκτρες είναι δημοφιλείς στον κόσμο της ράδιο-ελεγχόμενης αεροπορίας (RC) λόγω της απόδοσης, της ισχύος, της μακροζωίας και του μικρού βάρους τους σε σύγκριση με τους παραδοσιακούς κινητήρες brushed. Οι ελεγκτές κινητήρων συνεχούς ρεύματος brushless είναι πολύ πιο περίπλοκοι από τους ελεγκτές κινητήρα brushed.

Η σωστή φάση του ρεύματος που τροφοδοτείται στον κινητήρα ποικίλλει ανάλογα με την περιστροφή του κινητήρα, η οποία πρέπει να λαμβάνεται υπόψη από το ESC: Συνήθως, το back EMF από τις περιελίξεις του κινητήρα χρησιμοποιείται για την ανίχνευση αυτής της περιστροφής, αλλά υπάρχουν παραλλαγές που χρησιμοποιούν ξεχωριστό μαγνητικούς (εφέ Hall) αισθητήρες ή

οπτικούς ανιχνευτές. Τα προγραμματιζόμενα από υπολογιστή χειριστήρια ταχύτητας έχουν γενικά επιλογές που καθορίζονται από το χρήστη που επιτρέπουν τον καθορισμό ορίων αποκοπής χαμηλής τάσης, χρονισμού, επιτάχυνσης, πέδησης και κατεύθυνσης περιστροφής. Η αντιστροφή της κατεύθυνσης του κινητήρα μπορεί επίσης να επιτευχθεί με εναλλαγή οποιωνδήποτε δύο από των τριών καλωδίων από το ESC στον κινητήρα [19].



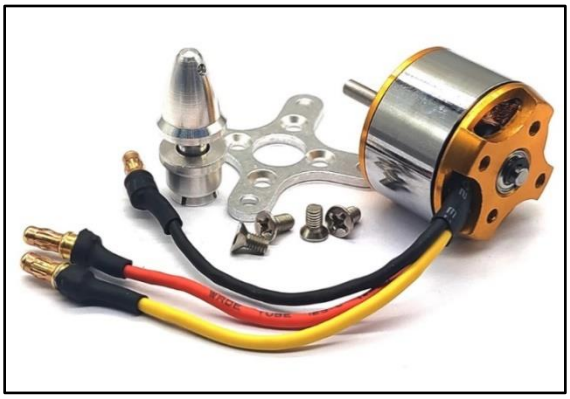
Εικόνα 38. Οι κυματομορφές που οδηγούν τους ESC's. 1500 μsec και 200 Hz refresh rate .

2.5 Brushless Motors

Ένας ηλεκτροκινητήρας συνεχούς ρεύματος χωρίς ψήκτρες (BLDC, **brushless DC electric motor**), γνωστός και ως, ηλεκτρονικά εναλλάκτης κινητήρας, είναι ένας ΄΄σύγχρονος΄΄ κινητήρας που χρησιμοποιεί τροφοδοτικό συνεχούς ρεύματος (DC). Χρησιμοποιεί έναν ηλεκτρονικό ελεγκτή για τη μετάβαση των ρευμάτων συνεχούς ρεύματος στις περιελίξεις του κινητήρα που παράγουν μαγνητικά πεδία που περιστρέφονται αποτελεσματικά στο χώρο και τα οποία ακολουθεί ο ρότορας μόνιμου μαγνήτη. Ο ελεγκτής προσαρμόζει τη φάση και το πλάτος των παλμών συνεχούς ρεύματος για να ελέγχει την ταχύτητα και τη ροπή του κινητήρα. Αυτό το σύστημα ελέγχου είναι μια εναλλακτική λύση στον μηχανικό μεταγωγέα (brushes) που χρησιμοποιείται σε πολλούς συμβατικούς ηλεκτρικούς κινητήρες.

Τα πλεονεκτήματα ενός κινητήρα χωρίς ψήκτρες έναντι των brushed κινητήρων είναι η υψηλή αναλογία ισχύος προς βάρος, η υψηλή ταχύτητα, ο σχεδόν στιγμιαίος έλεγχος της ταχύτητας (rpm) και της ροπής, η υψηλή απόδοση και η χαμηλή συντήρηση. Χωρίς περιελίξεις στον ρότορα, δεν υπόκεινται σε φυγόκεντρες δυνάμεις και επειδή οι περιελίξεις υποστηρίζονται από το περίβλημα, μπορούν να ψύχονται με αγωγιμότητα, χωρίς να απαιτείται ροή αέρα μέσα στον κινητήρα για ψύξη.

Οι κινητήρες χωρίς ψήκτρες έχουν γίνει μια δημοφιλής επιλογή κινητήρα για μοντέλα αεροσκαφών, συμπεριλαμβανομένων ελικοπτέρων και drones. Οι ευνοϊκές αναλογίες ισχύος προς βάρος καθώς και η μεγάλη γκάμα διαθέσιμων μεγεθών έχουν φέρει επανάσταση στην αγορά μοντέλων πτήσης με ηλεκτρική ενέργεια, εκτοπίζοντας ουσιαστικά όλους τους ηλεκτρικούς brushed κινητήρες, εκτός από τα φθηνά αεροσκάφη χαμηλής ισχύος συχνά κατηγορίας παιχνιδιών. Η αυξημένη αναλογία ισχύος προς βάρος των σύγχρονων μπαταριών και των κινητήρων χωρίς ψήκτρες επιτρέπει στα μοντέλα να ανεβαίνουν κάθετα, αντί να ανεβαίνουν σταδιακά. Ο χαμηλός θόρυβος και η έλλειψη μάζας σε σύγκριση με τους κινητήρες εσωτερικής καύσης είναι ένας άλλος λόγος για τη δημοτικότητά τους.



Εικόνα 39. Brushless Motor

Το χαρακτηριστικό Kv του κινητήρα παρέχει έναν τρόπο περιγραφής της σχέσης μεταξύ της μέγιστης τάσης και της ταχύτητας περιστροφής σε έναν κινητήρα χωρίς ψήκτρες σε κατάσταση χωρίς φορτίο. Η μονάδα για Kv είναι RPM/V και μπορεί να υπολογιστεί διαιρώντας την ταχύτητα περιστροφής του κινητήρα χωρίς φορτίο με την εφαρμοζόμενη τάση. Στην πραγματικότητα, η τάση που θα πρέπει να λάβουμε υπόψη είναι στην πραγματικότητα η πίσω ηλεκτροκινητική δύναμη (back EMF) και όχι η εφαρμοζόμενη τάση. Σε έναν κινητήρα χωρίς ψήκτρες, η πίσω ηλεκτροκινητική δύναμη (Back EMF) είναι μια τάση που εμφανίζεται στην αντίθετη κατεύθυνση του ρεύματος που παρέχεται από την πηγή ισχύος, που προκαλείται από την κίνηση των πηνίων μέσω του μαγνητικού πεδίου στον κινητήρα. Αυτό μπορεί να εξηγηθεί από το νόμο του Faraday, ο οποίος δείχνει ότι ένα πηνίο που κινείται μέσα σε ένα μαγνητικό πεδίο δημιουργεί μια ροή ηλεκτρονίων μέσα στο πηνίο, που ονομάζεται τάση ή emf. Όταν ο κινητήρας περιστρέφεται, η Back EMF που δημιουργείται είναι ανάλογη με την ταχύτητα του ρότορα: καθώς αυξάνονται οι στροφές ανά λεπτό, η Back EMF αυξάνεται επίσης. Με πλήρες Throttle χωρίς φορτίο (αγνοώντας την αδράνεια του ίδιου του κινητήρα), μπορούμε να περιγράψουμε τη σχέση με αυτόν τον τύπο:

$$\omega_{rpm} = K_v \cdot Back\ EMF$$

where ω_{rpm} = rotation speed

Εικόνα 40. Γωνιακή ταχύτητα κινητήρα σε συνάρτηση με το Kv

Η Kv τιμή παρέχει μια εκτίμηση του πόσες περιστροφές θα υποστεί ένας κινητήρας για κάθε βολτ που εφαρμόζεται σε αυτόν. Αυτή η τιμή μπορεί να είναι χρήσιμη για τη σύγκριση κινητήρων που έχουν το ίδιο μέγεθος φυσικά, αλλά έχουν διαφορετικά χαρακτηριστικά απόδοσης λόγω της εσωτερικής τους λειτουργίας. Κατά γενικό κανόνα, όσο αυξάνεται ο αριθμός των περιελίξεων στα πηνία, το Kv του κινητήρα μειώνεται. Μηχανικά μιλώντας, οι κινητήρες χαμηλού Kv έχουν μεγαλύτερο αριθμό περιελίξεων ενός λεπτότερου σύρματος και το λεπτό καλώδιο μεταφέρει περισσότερα βολτ σε χαμηλότερο ρεύμα. Οι κινητήρες υψηλού Kv έχουν λιγότερες περιελίξεις αλλά παχύτερο σύρμα που μπορεί να μεταφέρει υψηλότερο ρεύμα με λιγότερα βολτ [20] [21].

2.6 Lipo Battery

Μια μπαταρία πολυμερούς λιθίου ή αλλιώς μια μπαταρία πολυμερούς ιόντων λιθίου (συντομογραφία LiPo, LIP, Li-poly, lithium-poly και άλλες), είναι μια επαναφορτιζόμενη μπαταρία τεχνολογίας ιόντων λιθίου που χρησιμοποιεί έναν ηλεκτρολύτη πολυμερούς αντί για έναν υγρό ηλεκτρολύτη. Τα ημιστερεά πολυμερή (gel) υψηλής αγωγιμότητας σχηματίζουν αυτόν τον ηλεκτρολύτη. Οι μπαταρίες LiPo παρέχουν υψηλότερη ειδική ενέργεια από άλλους τύπους μπαταριών λιθίου και χρησιμοποιούνται σε εφαρμογές όπου το βάρος είναι κρίσιμο χαρακτηριστικό, όπως στην περίπτωση μας για την τροφοδοσία του τετρακόπτερου.

Εικόνα 41. LiPo μπαταρία

Η τάση μιας LiPo εξαρτάται από τη χημεία της και κυμαίνεται από περίπου 4,2 V (πλήρως φορτισμένη) έως περίπου 2,7–3,0 V (πλήρως αποφορτισμένη), όπου η ονομαστική τάση είναι 3,6 ή 3,7 V. Οι ακριβείς ονομασίες τάσης θα πρέπει να καθορίζονται στα data sheets του προϊόντος, με την προϋπόθεση ότι οι κυψέλες πρέπει να προστατεύονται από ένα ηλεκτρονικό κύκλωμα που δεν θα τους επιτρέπει να υπερφορτίζονται ή να υπερεκφορτίζονται κατά τη χρήση.

Τα πακέτα μπαταριών LiPo (3 Cell), με κυψέλες συνδεδεμένα σε σειρά και παράλληλα, διαθέτουν ξεχωριστά pin-outs για κάθε στοιχείο. Ένας εξειδικευμένος φορτιστής μπορεί να παρακολουθεί τη φόρτιση ανά κυψέλη, έτσι ώστε όλες οι κυψέλες να φέρουν την ίδια κατάσταση φόρτισης (SOC).

Το C rating για μια μπαταρία LiPo αναφέρεται στη χωρητικότητα ενέργειας που μπορεί να εκ φορτίσει με ασφάλεια η μπαταρία, αντιπροσωπευόμενη ως πολλαπλάσιο της συνολικής χωρητικότητάς της. Μια μπαταρία που έχει υψηλότερη βαθμολογία C αποδίδει περισσότερη ενέργεια και αυτό σημαίνει υψηλότερη απόδοση. Η μεγάλη διαφορά μεταξύ μπαταριών με διαφορετικές ονομασίες C είναι το γεγονός ότι οι μπαταρίες υψηλότερης βαθμολογίας C υποφέρουν λιγότερο από πτώση τάσης υπό φορτίο. Αυτή η ικανότητα διατήρησης υψηλότερης και πιο σταθερής τάσης είναι αυτό που συνήθως δημιουργεί την υψηλή απόδοση που απολαμβάνουμε. Αυτό είναι ιδιαίτερα σημαντικό όταν έχετε πολλούς κινητήρες που περιστρέφουν πολλούς ρότορες.

Μια μπαταρία 2000 mAh, 10 C , θα παρέχει με ασφάλεια 20 αμπέρ συνεχώς (10 φορές 2000 mAh ισούται με 20 αμπέρ). Μία μπαταρία 20 C της ίδιας χωρητικότητας θα μας τροφοδοτήσει με ασφάλεια 40 αμπέρ (2000 mAh x 20 = 40 A) από αυτό συνεχώς. Αυτή είναι η ίδια χωρητικότητα και η διπλάσια αντλία ενισχυτή. Υπάρχουν μπαταρίες με ονομαστική τιμή C, 30 C, 45 C, ακόμη και 70 C που θα αποδίδουν (στο παράδειγμά μας 2000 mAh) 60, 90 και 140 αμπέρ. Αυτή η σημαντική επιλογή της μέγιστης παροχή του ρεύματος μας επιτρέπει να λειτουργούμε μεγαλύτερους (ή περισσότερους) κινητήρες για αυξημένη χρονική διάρκεια ή / και δυνατότητες ανύψωσης μεγαλύτερου ωφέλιμου φορτίου (payload) [22].

2.7 Τηλεκατεύθυνση

Η FLYSKY FS-i6 είναι μια πολύ καλή και υψηλής ποιότητας εξακάναλη ψηφιακή τηλεκατεύθυνση, εξοπλισμένη με τον FS-iA6B δέκτη μεγάλης εμβέλειας. Διαθέτει όλες τις απαραίτητες ρυθμίσεις για κάθε μοντέλο σκάφους, αεροπλάνου ή ελικοπτέρου. Είναι ελαφριά και εργονομικά σχεδιασμένη ώστε να μην κουράζει τον χρήστη. Το σύστημα εκπομπής σήματος (AFHDS) εξασφαλίζει μεγάλη εμβέλεια, σταθερή και χωρίς παρεμβολές επικοινωνία πομπού–δέκτη. Έχει τηλεμετρία που μας ενημερώνει για το ποσοστό λάθους στην επικοινωνία με τον δέκτη.

Εικόνα 42. Flysky FS-i6, εξακάναλος τηλεχειρισμός στα 2.4 GHz

Η περιοχή λειτουργίας του είναι στα 2.4 GHz (2.4055 – 2.4750 GHz), με ισχύς εξόδου < 20dbm , έχει 140 κανάλια με BW 500 KHz ενώ χρησιμοποιεί τεχνολογία Frequency Hopping . Η διαμόρφωση που χρησιμοποιεί είναι η Gaussian FSK [23].

2.8 DC Voltage Step-Down Switching Regulators

Δύο step-down, DC-DC converters χρησιμοποιούνται για να αποφορτίσουν το Arduino Nano από την τροφοδοσία του δέκτη του τηλεχειρισμού και των υπολοίπων περιφερειακών. Ο πρώτος τροφοδοτείται από την μπαταρία και ρυθμίζει τα 12,6 V σε 4,8 V. Το module που χρησιμοποιείται έχει ως ολοκληρωμένο το LM2596. Μπορεί να τροφοδοτήσει το κύκλωμα με μέγιστο ρεύμα 3.0 Ampere [24].

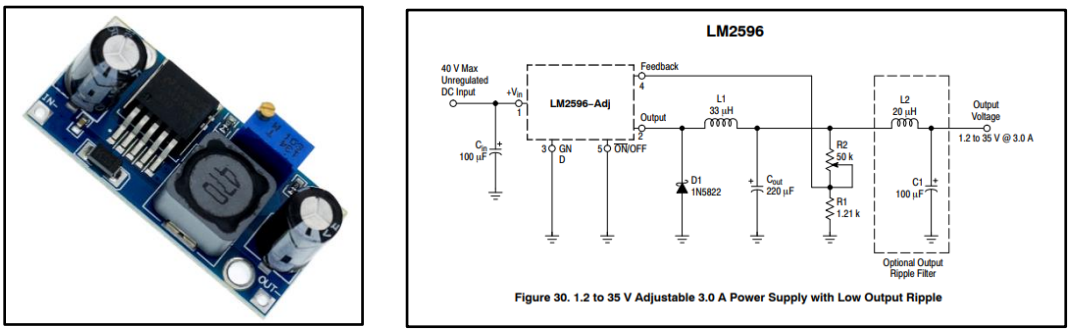
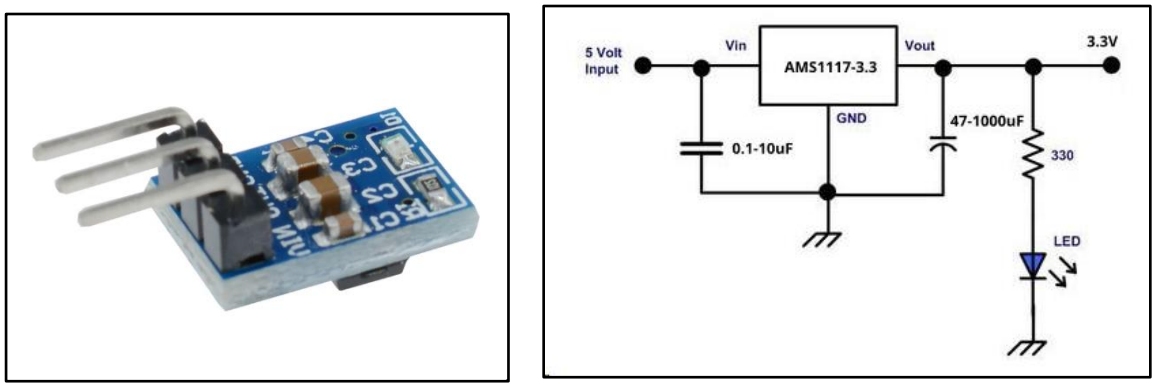


Figure 30. 1.2 to 35 V Adjustable 3.0 A Power Supply with Low Output Ripple

Εικόνα 43. DC-DC step down converter – LM2596 [24]

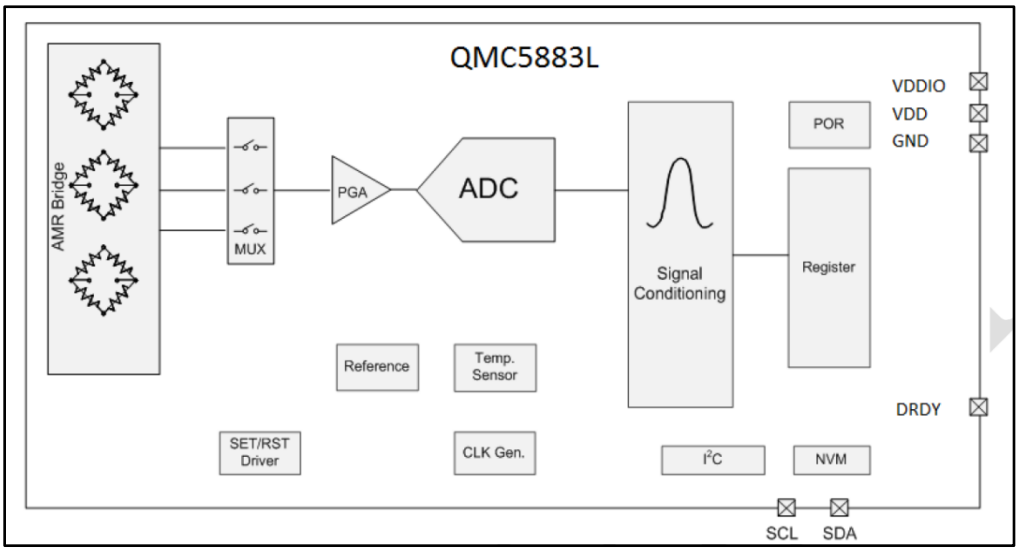
Ο δεύτερος converter αναλαμβάνει την παροχή 3,3 V DC τάσης για τα περιφερειακά κυκλώματα. Βασίζεται στο ολοκληρωμένο AMS1117 και έχει μέγιστο ρεύμα εξόδου περί τα 800 mA [25].



Εικόνα 44. DC-DC step down converter – AMS1117 [25]

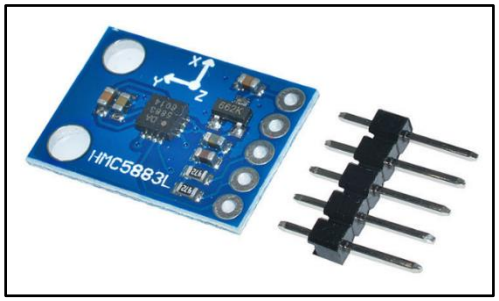
2.9 QMC5883L Compass

Η πυξίδα που εξετάζουμε στην εφαρμογή μας είναι το chip QMC5883L. Το chip αυτό στηρίζεται στην τεχνολογία AMR της Honeywell. Ο ειδικά σχεδιασμένος 16-bit ADC ASIC (Application - Specific Integrated Circuit) προσφέρει αρκετά πλεονεκτήματα όπως χαμηλό θόρυβο, υψηλή ακρίβεια, χαμηλή κατανάλωση ενέργειας, offset cancellation και αντιστάθμιση θερμοκρασίας. Το chip της QMC5883L προσφέρει ακρίβεια στο heading 1° έως 2°, ενώ συνδέεται εύκολα στον μικροελεγκτή με την διεπαφή IIC.



Εικόνα 45. Μπλοκ Διάγραμμα πυξίδας [26]

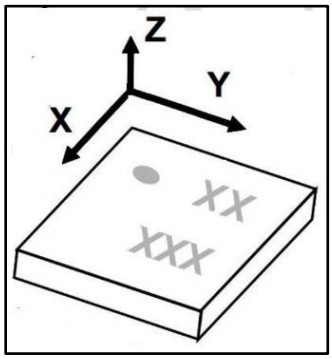
Το κύκλωμα του αισθητήρα αποτελείται από τριών αξόνων αισθητήρες και ειδικά κυκλώματα για τη μέτρηση μαγνητικών πεδίων. Όταν τροφοδοτείται με τάση ο αισθητήρας μετατρέπει οποιοδήποτε προσπίπτον μαγνητικό πεδίο στις διευθύνσεις του ευαίσθητου άξονα σε διαφορική τάση εξόδου. Το ASIC τότε ενισχύει και επεξεργάζεται το σήμα για να έχει ψηφιακή έξοδο. Η συσκευή συνδέεται σε έναν σειριακό δίαυλο διεπαφής ως εξαρτημένη συσκευή υπό τον έλεγχο μιας κύριας συσκευής, όπως ο μικροελεγκτής. Ο έλεγχος αυτής της συσκευής πραγματοποιείται μέσω I²C. Ο ρυθμός μετάδοσης δεδομένων με τον μικροελεγκτή είναι 100KHz ή 400KHz. Η προεπιλεγμένη διεύθυνση I2C είναι 0D: 0001101



Εικόνα 46. GY-273 Module

Στην εφαρμογή μας η πυξίδα χρησιμοποιείται σε μορφή module με την ονομασία GY-273. Όπου Vcc η τροφοδοσία του (3V3), GND η γείωση, SDA και SCL τα δύο pin της σειριακής διεπαφής IIC και τέλος το DRDY, data ready interrupt pin.

Στην εικόνα 3 φαίνονται οι άξονες προσανατολισμού που δίνουν θετική έξοδο στον ADC. Πρέπει να ληφθεί υπόψη γιατί στην πορεία όταν θα γίνει αντιστάθμιση (tilt compensation), αφορά τον προσανατολισμό του module.



Εικόνα 47. Προσανατολισμός chip QMC5883L [26]

Από το manual του κατασκευαστή όπως φαίνεται παρακάτω κατά την διάρκεια της αρχικοποίησης θα πρέπει να δηλωθούν οι παρακάτω παράμετροι. Αυτό συμβαίνει με την παρακάτω εντολή:

setMode(0x01,0x0C,0x10,0X00);

όπου Mode: 01, Continuous

Output Data Rate: 0C, 200Hz

Full Scale: 10, 8G

Over Sample Ratio: 00, 512

Addr	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
09H	OSR[1:0]		RNG[1:0]		ODR[1:0]		MODE[1:0]	
Reg.	**Definition**		**00**		**01**		**10**	**11**
Mode	Mode Control		Standby		Continuous		Reserve	Reserve
ODR	Output Data Rate		10Hz		50Hz		100Hz	200Hz
RNG	Full Scale		2G		8G		Reserve	Reserve
OSR	Over Sample Ratio		512		256		128	64

Εικόνα 48. QMC5883L Setup Registers. [26]

Για να διαβάσουμε τα δεδομένα από το chip οι παρακάτω εντολές χρησιμοποιούνται:

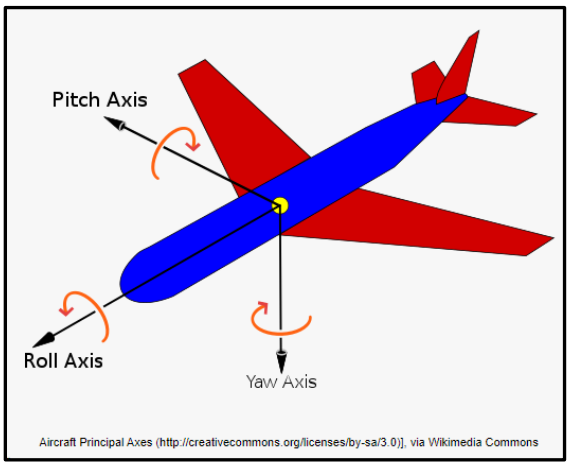
```
Wire.requestFrom(0x0D, (byte)6);

vRaw [0] = (int)(int16_t) (Wire.read() |
Wire.read() << 8);

vRaw [1] = (int)(int16_t) (Wire.read() |
Wire.read() << 8);

vRaw [2] = (int)(int16_t) (Wire.read() |
Wire.read() << 8);
```

Addr.	7	6	5	4	3	2	1	0
00H	Data Output X LSB Register XOUT[7:0]							
01H	Data Output X MSB Register XOUT[15:8]							
02H	Data Output Y LSB Register YOUT[7:0]							
03H	Data Output Y MSB Register YOUT[15:8]							
04H	Data Output Z LSB Register ZOUT[7:0]							
05H	Data Output Z MSB Register ZOUT[15:8]							

Εικόνα 49. QMC5883L Data registers [26]

Όπου, με αρχική διεύθυνση την 00H ζητάει για τις επόμενες 6 διευθύνσεις , 8 bit data. Οι έξοδοι των τιμών αφορούν τα X,Y,Z.

Η συμβατική μέθοδος του αεροπλάνου χρησιμοποιείται για να καθοριστεί το Heading της πυξίδας.



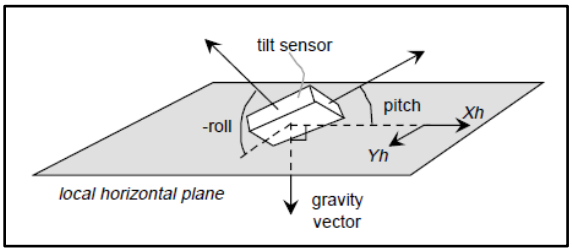
Εικόνα 50. Συμβατική μέθοδος απεικόνισης

Αν λοιπόν το αεροπλάνο ήταν σε οριζόντια θέση για τον άξονα Y και σε οριζόντια θέση για τον άξονα X, τότε αν μια πυξίδα βρισκόταν στο πιλοτήριο το Heading της θα υπολογιζόταν από την σχέση:

$$\text{Heading} = \text{Arctan}\,(Yh/Xh) \qquad \text{(rad)}$$

όπου Yh και Xh τα μαγνητικά πεδία της Γης για τους δύο αυτούς άξονες. Στην δικιά μας περίπτωση τα Yh,Xh είναι οι έξοδοι των καταχωρητών στις μεταβλητές vRaw[0], vRaw[1]. Καθώς το αεροπλάνο κινείται κατά τον άξονα Z, το Heading της πυξίδας αλλάζει από 0 έως 360 μοίρες.

Για την κατάσταση όμως που το αεροπλάνο δεν είναι level τα δεδομένα είναι διαφορετικά. Οι γωνίες pitch και roll όπως επίσης και τα μαγνητικά πεδία και των τριών αξόνων πρέπει να χρησιμοποιηθούν για να υπολογιστεί το νέο Heading.



Εικόνα 51. Tilt Sensor [7]

Στην δική μας περίπτωση οι γωνίες pitch και roll δίνονται από το gyro/accelerometer MPU6050. Οι εξισώσεις που θα μας δώσουν το νέο Heading είναι οι ακόλουθες που προέρχονται από τις σχέσεις της σελίδας 31 και έχουν προσαρμοστεί ανάλογα :

$$Xh = X*\cos(\phi) + Y*\sin(\theta)*\sin(\phi) + Z*\cos(\theta)*\sin(\phi) \quad \text{(rad)}$$
$$Yh = Y*\cos(\theta) - Z*\sin(\theta) \quad \text{(rad)}$$

όπου X,Y,Z οι τιμές την έντασης του μαγνητικού πεδίου και φ, θ το pitch και το roll από το MPU6050. Στις γραμμές που ακολουθούν θα μετατρέψουμε τις εξισώσεις αυτές στα δικά μας δεδομένα ώστε να καταλήξουν σε αυτές που χρησιμοποιούνται στον κώδικα προγραμματισμού. Έτσι :

Xhorizontal = raw_X*cos(pitch)+raw_X*sin(roll)*sin(pitch)+raw_Z*cos(roll)*sin(pitch) (rad)
Yhorizontal = raw_Y*cos(roll)-raw_Z*sin(roll) (rad)

Για να συμβαδίζουμε με τον συμβατικό τρόπο προσανατολισμού δύο αλλαγές πρέπει να γίνουν, αφενός πρέπει η πυξίδα να περιστραφεί κατά 180 μοίρες ώστε ο άξονες Z να ταυτιστεί με το διάνυσμα της βαρύτητας και αφετέρου το pitch που παρέχεται από το MPU 6050 πρέπει να αλλάξει πρόσημο . Έτσι μπορούμε να γράψουμε :

raw_Z = - rawZ και
pitch = - ypr [1]

Τελικά τα Xhorizontal και Yhorizontal δίνονται από τις σχέσεις :

Xhorizontal = (float)raw_X * cos (-ypr [1]) + (float)raw_Y * sin (ypr [2]) * sin (-ypr [1]) - (float)raw_Z * cos (ypr [2]) * sin (-ypr [1])

Yhorizontal = (float)raw_Y * cos (ypr [2]) + (float)raw_Z * sin (ypr [2])

όπου : raw_X, raw_Y, raw_Z οι τιμές raw από το GY-273 και ypr[1], ypr[2], οι τιμές pitch και roll από το MPU6050 σε rad [26].

Με την χρήση της γνωστής εξίσωσης του Heading παίρνουμε την νέα τιμή. Παράλληλα πρέπει να αναφερθούμε στο γνωστό magnetic declination. Εύκολα από το site

https://www.magnetic-declination.com/Greece/Athens/955650.html βρίσκουμε ότι για την περιοχή της Αθήνας η τιμή είναι +5.4 μοίρες, οι οποίες μετατρέπονται σε rad και είναι +0.0942477796 που προστίθεται στην τελική μορφή της εξίσωσης . Τελικά προκύπτει :

Heading = (atan2(Yhorizontal, Xhorizontal)+0.0942477796) * 180.0/ PI ;

Κάνοντας αναγωγή στο σύστημα πολικών συνταγμένων και προσθέτοντας στις αρνητικές τιμές, 360 μοίρες, παίρνουμε τελικά το Heading που για οποιαδήποτε τιμή pitch και roll έχει απόκλιση το πολύ +_2 μοίρες.

2.10 VL53L1X Time of Flight Sensor

Ο VL53L1X είναι ένας τελευταίας τεχνολογίας Time-of-Flight (ToF), της οικογένειας ST FlightSense™. Είναι ο πιο γρήγορος αισθητήρας ToF με ακριβή εμβέλεια έως 4 m και συχνότητα ανανέωσης μέτρησης έως 50 Hz

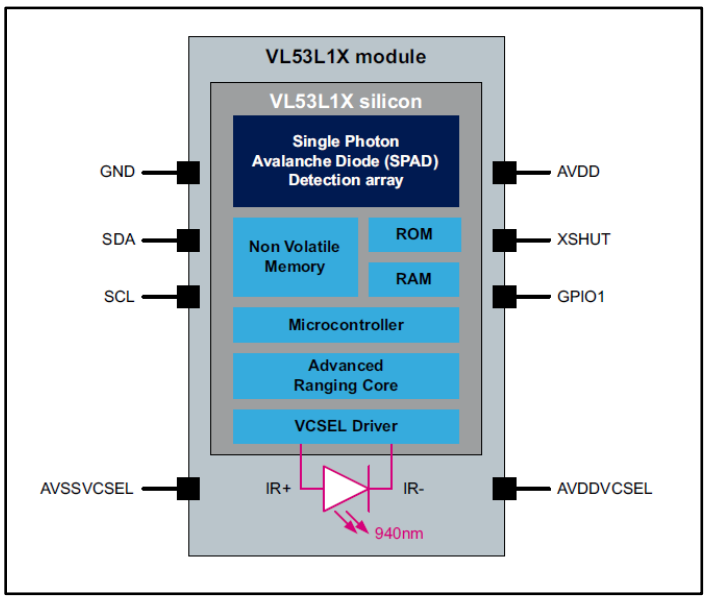
Στεγασμένο σε μια μινιατούρα, ενσωματώνει μια συστοιχία λήψης SPAD, έναν πομπό λέιζερ Class1 940 nm, μη ορατό στο ανθρώπινο μάτι, φυσικά φίλτρα υπέρυθρων και οπτικά για την επίτευξη της καλύτερης απόδοσης εμβέλειας σε διάφορες συνθήκες φωτισμού περιβάλλοντος.

Σε αντίθεση με τους συμβατικούς αισθητήρες υπέρυθρων, το VL53L1X χρησιμοποιεί την τεχνολογία ToF τελευταίας γενιάς της ST που επιτρέπει τη μέτρηση απόλυτης απόστασης ανεξάρτητα από το χρώμα και την ανάκλαση στόχου.

Είναι επίσης δυνατός ο προγραμματισμός του μεγέθους του ROI στη συστοιχία λήψης, επιτρέποντας τη μείωση του αισθητήρα FoV. Το Field of View του αισθητήρα είναι 27 μοίρες και στις δύο διαστάσεις X,Y.

Εάν η απόσταση εμβέλειας δεν μπορεί να μετρηθεί (στην περίπτωση που δεν υπάρχει στόχος ή θόρυβος ή ασθενές σήμα), δημιουργείται μια αντίστοιχη κατάσταση εύρους (Range Status) που μπορεί να διαβαστεί από τον μικροελεγκτή. Το πρόγραμμα οδήγησης λογισμικού VL53L1X παρέχει λειτουργίες στον χρήστη για την ανάγνωση των αποτελεσμάτων εξόδου μετά τη μέτρηση. Οι κύριες τιμές που αναφέρονται είναι:

- Απόσταση εμβέλειας σε mm (Ranging distance in mm)
- Ρυθμός σήματος επιστροφής (Return signal rate)
- Ρυθμός σήματος περιβάλλοντος (Ambient signal rate)
- Κατάσταση εύρους (Range Status).



Εικόνα 52. Μπλόκ διάγραμμα VL53L1X [27]

Το VL53L1X έχει τρεις λειτουργίες μέτρησης απόστασης (Distance Measurement): μικρή, μεσαία και μεγάλη (Short, Medium,Long). Η λειτουργία μέγιστης απόστασης επιτρέπει μετρήσεις έως την εμβέλεια των 4 m. Όμως , αυτή η μέγιστη απόσταση εμβέλειας είναι ευαίσθητη στο φως του περιβάλλοντος. Αντιθέτως η λειτουργία μικρής απόστασης είναι αυτή με την μεγαλύτερη ανοσία στο φως του περιβάλλοντος, αλλά η μέγιστη απόσταση εμβέλειάς της περιορίζεται συνήθως στα 1,35 m.

Distance mode	Max. distance in dark (cm)	Max. distance under strong ambient light (cm)
Short	136	135
Medium	290	76
Long	360	73

Εικόνα 53. VL53L1X Distance Modes [27]

Με τον όρο Timing Budget ορίζεται ο χρόνος που απαιτείται για να εκτελεστεί μια μέτρηση. Η default τιμή είναι τα 33msec και είναι κατάλληλη για όλο το εύρος των μετρήσεων, ενώ κυμαίνεται από 8msec έως 200 msec. Αξίζει να αναφερθεί ότι όσο μεγαλώνει αυτός ο χρόνος τόσο καλύτερες είναι οι μετρήσεις, ωστόσο αυξάνεται η κατανάλωση ενέργειας και ίσως γίνει επικίνδυνο για το ανθρώπινο μάτι. Ακολουθεί ένα παράδειγμα δήλωσης των δύο προαναφερθέντων ιδιοτήτων από τον κώδικα του project:

```
tof_sensor.setDistanceMode(VL53L1X::Medium)
tof_sensor.setMeasurementTimingBudget(33000)
```

Το VL53L1X επικοινωνεί με τον μικροελεγκτή με την σειριακά διεπαφή IIC στην συχνότητα των 400Hz. Η διεύθυνση του είναι η 0x52.

Ο πίνακας 6 που ακολουθεί είναι πολύ χρήσιμος καθώς δίνει αναλυτικά τις τιμές που επιστρέφει η Range Status.

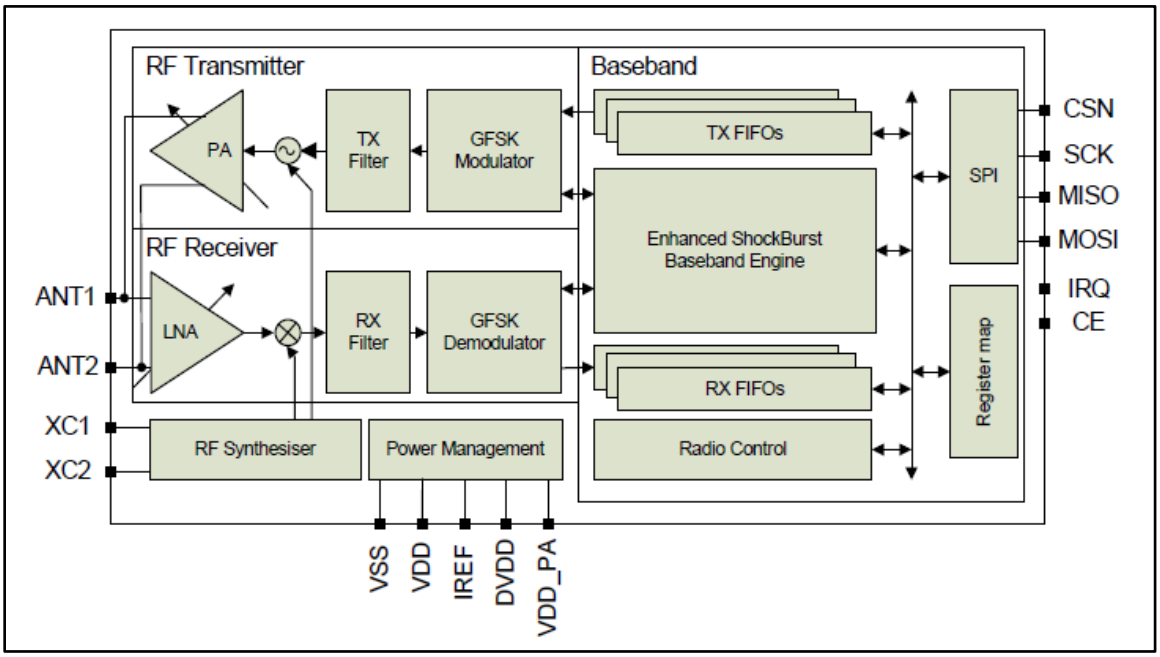
Value	RangeStatus String	Comment
0	VL53LX_RANGESTATUS_RANGE_VALID	Ranging measurement is valid
1	VL53LX_RANGESTATUS_SIGMA_FAIL	Raised if sigma estimator check is above the internal defined threshold. Sigma estimator gives a qualitative information about the signal.
2	VL53LX_RANGESTATUS_SIGNAL_FAIL	Raised when signal is too low to detect a target.
4	VL53LX_RANGESTATUS_OUTOFBOUNDS_FAIL	Raised when range result is out of bounds
5	VL53LX_RANGESTATUS_HARDWARE_FAIL	Raised in case of HW or VCSEL failure
6	VL53LX_RANGESTATUS_RANGE_VALID_NO_WRAP_CHECK_FAIL	No wraparound check has been done (this is the very first range)
7	VL53LX_RANGESTATUS_WRAP_TARGET_FAIL	Wraparound occurred
8	VL53LX_RANGESTATUS_PROCESSING_FAIL	Internal processing error
10	VL53LX_RANGESTATUS_SYNCRONISATION_INT	Raised one time after init, ranging value has to be ignored
11	VL53LX_RANGESTATUS_RANGE_VALID_MERGED_PULSE	Ranging is OK, but distance reported is the result of multiple targets merged together.
12	VL53LX_RANGESTATUS_TARGET_PRESENT_LACK_OF_SIGNAL	Indicate that there is a target, but signal is too low to report ranging
14	VL53LX_RANGESTATUS_RANGE_INVALID	Ranging data is negative and has to be ignored
255	VL53LX_RANGESTATUS_NONE	Target not detected, without no warning or errors raised

Πίνακας 6. VL53L1X Range Status [27]

Στο πρόγραμμα χρησιμοποιείται αρκετές φορές η συνάρτηση tof_sensor.ranging_data.range_status, η οποία όταν γυρίσει τιμή 0, υποδηλώνει ότι η μέτρηση είναι έγκυρη [27].

2.11 Το nRF24L01

Το nRF24L01 είναι ένα chip πομποδέκτη στα 2.4 GHz με μία ενσωματωμένη μηχανή πρωτοκόλλου Baseband (Enhanced ShockBurst™) της Nordic Semiconductor. Είναι κατάλληλος για χαμηλής ισχύος εφαρμογές και είναι σχεδιασμένος να λειτουργεί στην μπάντα των 2.400 GHz έως 2.4835 GHz. Για να σχεδιάσει κάποιος έναν πομποδέκτη χρειάζεται μόνο ένα μικροελεγκτή και μερικά παθητικά εξαρτήματα, που στην περίπτωση μας είναι το Arduino Uno/Nano. Το chip επικοινωνεί με τον μικροελεγκτή μέσω του πρωτοκόλλου SPI (Serial Peripheral Interface). Ο πομποδέκτης χρησιμοποιεί την διαμόρφωση GFSK (Gaussian FSK) και ο χρήστης είναι στην θέση να επιλέξει κανάλι συχνότητας, ισχύς εξόδου και data rate. Το nRF24L01 υποστηρίζει ρυθμό μετάδοσης δεδομένων 250 kbps, 1 Mbps και 2Mbps.



Εικόνα 54. Μπλοκ διάγραμμα nRF24L01 [28]

Το chip τροφοδοτείται με 3.3 Volt και η μέγιστη ισχύς εξόδου είναι στα 0 dbm. Η μέγιστη ευαισθησία λήψης επιτυγχάνεται στα 250 Kbps όπως φαίνεται στην εικόνα που ακολουθεί μέσα από το manual του κατασκευαστή.

SPI RF-SETUP (RF_PWR)	RF output power	DC current consumption
11	0dBm	11.3mA
10	-6dBm	9.0mA
01	-12dBm	7.5mA
00	-18dBm	7.0mA

Πίνακας 7. Ισχύς εξόδου – κατανάλωση, nRF24L01 [28]

Datarate	Symbol	Parameter (condition)	Notes	Min.	Typ.	Max.	Units
	RX_{max}	Maximum received signal at <0.1% BER			0		dBm
2Mbps	RX_{SENS}	Sensitivity (0.1%BER) @2Mbps			-82		dBm
1Mbps	RX_{SENS}	Sensitivity (0.1%BER) @1Mbps			-85		dBm
250kbps	RX_{SENS}	Sensitivity (0.1%BER) @250kbps			-94		dBm

Πίνακας 8. Ευαισθησία λήψης σε συνάρτηση με Bitrate, nRF24L01 [28]

Το πρωτόκολλο SPI δίνει την δυνατότητα να ελεγχθούνε όλες οι λειτουργίες του nRF24L01 με τα έξι παρακάτω ψηφιακά σήματα:

• IRQ (this signal is active low and controlled by three maskable interrupt sources)

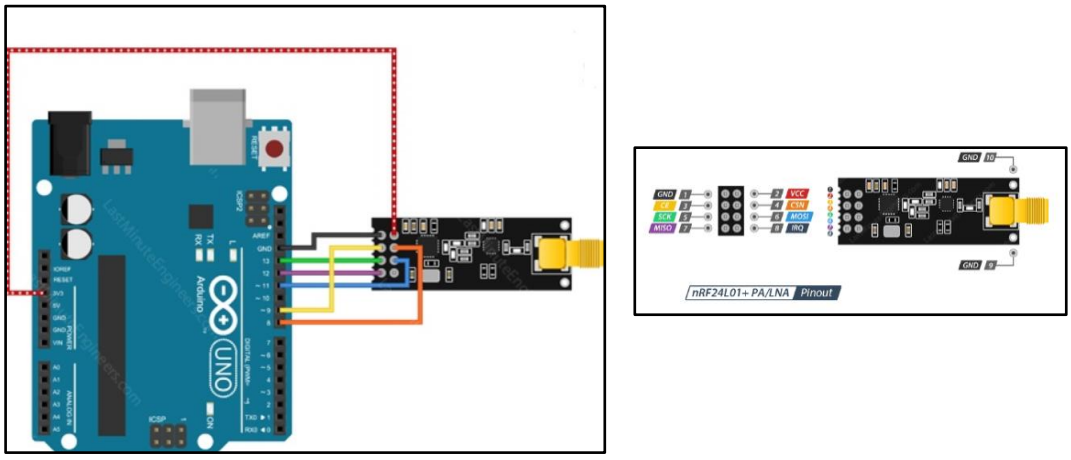
• CE (this signal is active high and used to activate the chip in RX or TX mode, Chip Enable)

• CSN (SPI signal, Chip Select Not)

• SCK (SPI signal, Serial Clock)

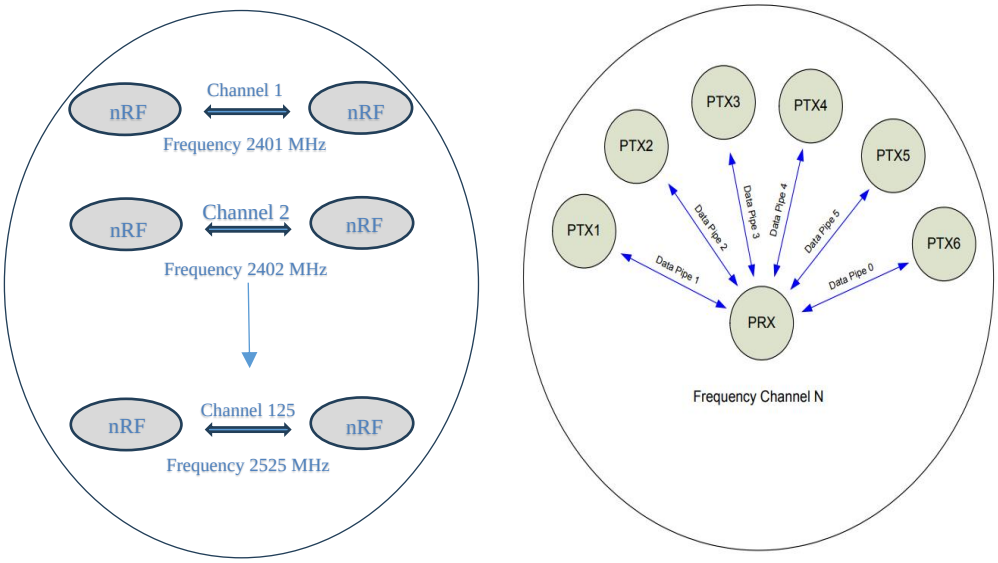
• MOSI (SPI signal, Master Output Slave Input)

• MISO (SPI signal, Master Input Slave Output)



Εικόνα 55. Arduino και nRF24L01 PLA

Το module μπορεί να χρησιμοποιεί 125 διαφορετικά κανάλια που δίνει τη δυνατότητα να εγκατασταθεί ένα δίκτυο 125 μόντεμ που λειτουργούν ανεξάρτητα σε ένα μέρος. Κάθε κανάλι μπορεί να έχει έως και 6 διευθύνσεις ή κάθε μονάδα μπορεί να επικοινωνεί με έως και 6 άλλες μονάδες ταυτόχρονα. Στη εικόνα που ακολουθεί παριστάνεται το εν δυνάμει δίκτυο.



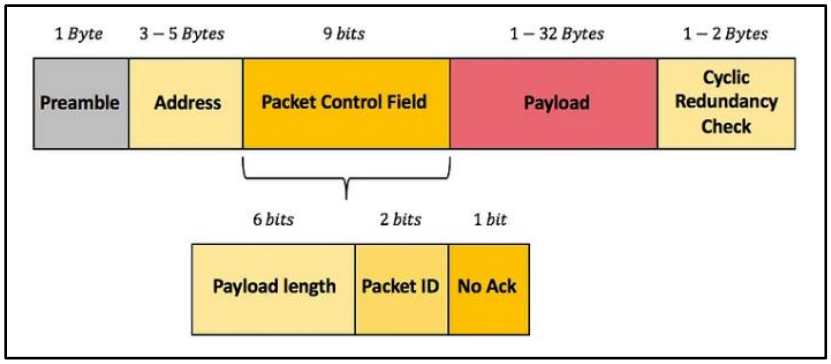
Εικόνα 56. Δίκτυο nRF [28]

Το RF κανάλι ορίζει την κεντρική συχνότητα του καναλιού. Το κανάλι καταλαμβάνει περίπου 1 MHz bandwidth για τα bitrate 250 Kbps και 1Mbps και περίπου 2 MHz για το bitrate των 2 Mbit. Η συχνότητα του RF καναλιού ορίζεται στον καταχωρητή RF_CH σύμφωνα με την σχέση :

$$F_0 = 2400 + RF\_CH \text{ [MHz]}$$

Έτσι για παράδειγμα αν επιλέξουμε να δώσουμε την τιμή 21 στο RF_CH θα έχουμε link στα 2421 MHz. Το module nRF24L01 PA/LNA (Power Amplifier/ Low Noise Amplifier) δύναται να εγκαταστήσει δίκτυο σε απόσταση έως και 1000 μέτρα στα 250 Kbps ευαισθησία -94 dbm και μέγιστη κατανάλωση 11.3 mA.

Το nRF24L01+ μπορεί να στείλει έως και 32 byte δεδομένων σε ένα πακέτο. Αυτό το πακέτο ορίζεται ως το Payload. Για να σταλθεί ένα πακέτο, ο επιθυμητός αριθμός byte προς αποστολή πρέπει να γραφτεί σε ένα buffer. Η δομή αυτού του πακέτου είναι μέρος του Enhanced ShockBurst™ και περιγράφεται στην εικόνα που ακολουθεί :



Εικόνα 57. Enhanced ShockBurstTM [28]

Παρατηρούμε ότι το Payload μπορεί να ποικίλλει 1 έως 32 byte. Επίσης, παρέχει σε κάθε απεσταλμένο πακέτο ένα αναγνωριστικό πακέτου (PID), το οποίο επιτρέπει στη συσκευή λήψης να καθορίσει εάν ένα μήνυμα είναι νέο ή αν έχει αναμεταδοθεί (και επομένως μπορεί να αγνοηθεί).

Παράλληλα, κάθε μήνυμα μπορεί να ζητήσει να σταλεί μια επιβεβαίωση (acknowledgement, ACK)όταν ληφθεί από άλλη συσκευή. Αυτό το μήνυμα επιβεβαίωσης μπορεί να περιέχει ένα προ φορτωμένο ωφέλιμο φορτίο, που σημαίνει ότι ένα σύστημα αμφίδρομων επικοινωνιών μπορεί να δημιουργηθεί εξ ολοκλήρου χρησιμοποιώντας μια κύρια συσκευή που μεταδίδει αιτήματα δεδομένων σε κάθε εξαρτημένη συσκευή διαδοχικά. Το nRF24L01+ το κάνει αυτό αποθηκεύοντας ένα προ φορτωμένο μήνυμα στο buffer μετάδοσης First-In-First-Out (FIFO) και στέλνοντας αυτό το μήνυμα στη διεύθυνση του Data pipe αμέσως μόλις ληφθεί ένα μήνυμα με αίτημα Payload.

Το CRC , Cyclic Redundancy Check , είναι ένας μηχανισμός αναγνώρισης λαθών και είναι απαραίτητος σύμφωνα με το Enhanced ShockBurst^{TM} . Είναι 1 ή 2 bytes και υπολογίζεται πάνω στην διεύθυνση, το Packet Control Field και το Payload.

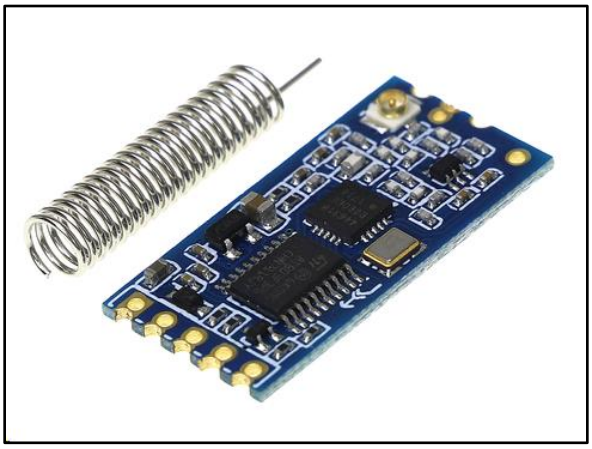
Το πολυώνυμο για ένα byte CRC είναι $X^8 + X^2 + X + 1$ με αρχική διεύθυνση 0xFF.

Το πολυώνυμο για δύο bytes CRC είναι $X^{16} + X^{12} + X^5 + 1$ με αρχική διεύθυνση 0xFFFF.

Σύμφωνα με το πρωτόκολλο αν το CRC αποτύχει , το πακέτο δεν γίνεται δεκτό [28].

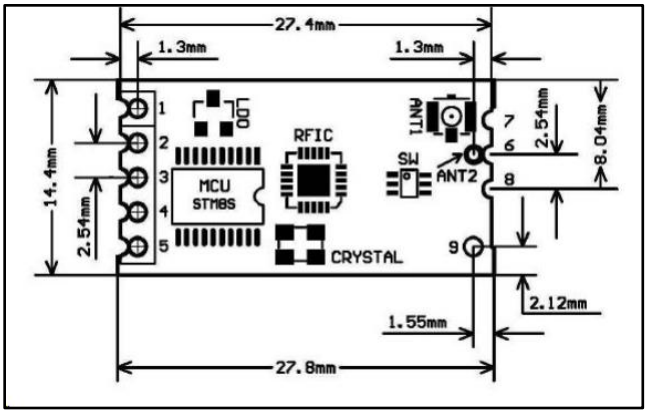
2.12 HC-12 Half Duplex Wireless Transceiver

Το module HC-12 είναι ένας half - duplex δέκτης , ισχύος 20 dbm ή 100mW ο οποίος έχει συνδυαστεί με έναν δέκτη με ευαισθησία -117 dbm σε ρυθμό δεδομένων 5 Kbps. Το module έχει ολοκληρωθεί βασικά με τον μικροελεγκτή STM8S003F3 και τον πομποδέκτη Si4463. Αν εφαρμοστεί κεραία τύπου ελατηρίου ο πομποδέκτης είναι ικανός να πραγματοποιήσει τηλεπικοινωνιακό δίκτυο απόστασης ως και 1 Km.



Εικόνα 58. HC-12 Module

Το εύρος της συχνότητας λειτουργίας είναι μεταξύ των συχνοτήτων 433.4 - 473.0 MHz. Συνολικά υπάρχουν 100 κανάλια λειτουργίας με απόσταση 400 KHz το καθένα. Το Module διαθέτει UART (Universal Asynchronous Receiver / Transmitter) διεπαφή και ελέγχεται πλήρως με AT Commands (Attention Commands) με την χρήση κάποιου προγράμματος σειριακής επικοινωνίας.



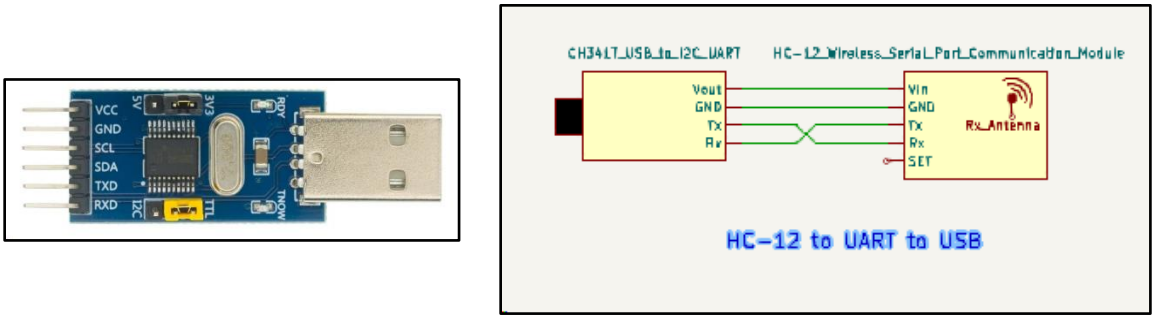
Εικόνα 59. HC-12 Pinout [29]

Στην εικόνα 54 το pinout του HC-12 module, όπου :

- Pin 1 , Vcc, Τάση τροφοδοσίας DC , εύρους 3,2 - 5,5 Volt
- Pin 2, Ground, Γείωση

- Pin 3, RXD, UART θύρα εισόδου
- Pin 4, TXD, UART θύρα εξόδου
- Pin 5, SET, pin ελέγχου
- ANT2, ANT, 433 MHz spring antenna
- ANT1, IPEX20279-001E-03 antenna socket

Το HC-12 για να συνδεθεί με την USB του υπολογιστή και να ρυθμιστεί, πρέπει να χρησιμοποιήσει έναν UART σε USB μετατροπέα. Στην περίπτωση μας αυτός εικονίζεται παρακάτω :



Εικόνα 60. CH341T/UART σε USB και συνδεσμολογία HC-12 και UART to USB

Το HC -12 για να εισέλθει σε κατάσταση παραμετροποίησης πρέπει να τεθεί σε κατάσταση LOW το pin 5. Αυτό πραγματοποιείται όταν ΄΄ δώσουμε΄΄ Ground στο συγκεκριμένο pin. Στη συνέχεια με την βοήθεια ενός Serial Communicator το module θα δεχθεί τις AT εντολές . Όλες οι δυνατές εντολές παρουσιάζονται παρακάτω.

AT+Bxxxx : Επιλογή Bitrate

AT+Cxxx : Επιλογή καναλιού επικοινωνίας

AT+FUx : Επιλογή τρόπου εκπομπής

AT+Px : Επιλογή ισχύος εκπομπής

AT+Rx : Όλοι οι παράμετροι του module

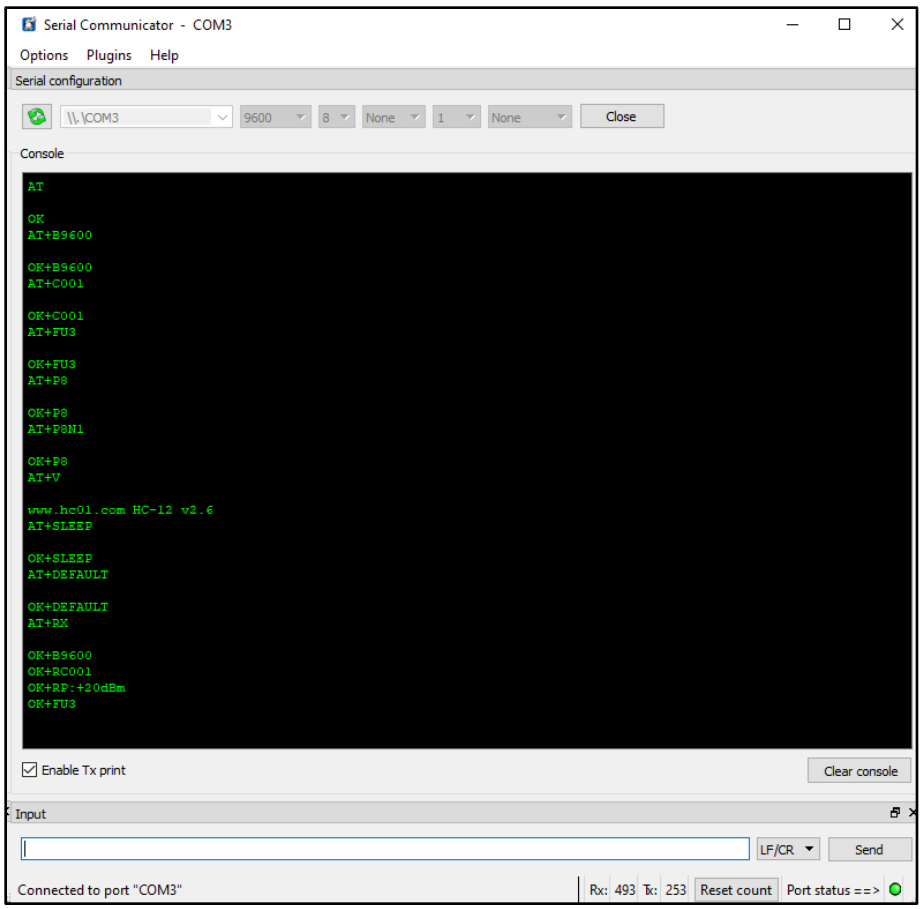
AT+Uxxx : Data bit, Check bit, Stop bit

AT+V : έκδοση Firmware

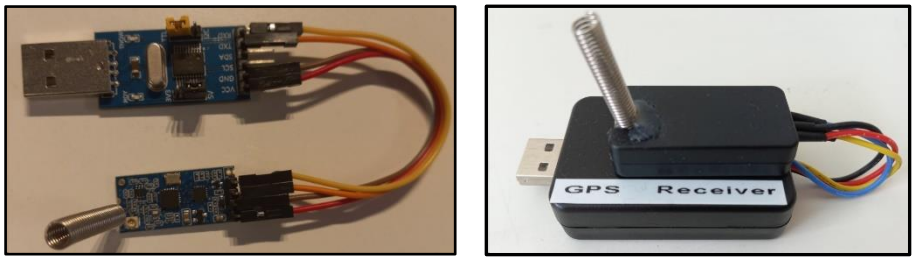
AT+SLEEP : Sleep Mode

AT+DEFAULT : Factory Defaults

Στην εικόνα 56 είναι συγκεντρωμένες όλες οι AT εντολές με τις απαντήσεις τους . Η σύνδεση είναι στην COM3 με ταχύτητα 9600 bps , 8N1 – Data Bits , Check Bits, Stop Bit. Η εκπομπή στα 20 dbm (100 mW), στο κανάλι 1, 433,4 MHz με τρόπο εκπομπής FU3 [29].



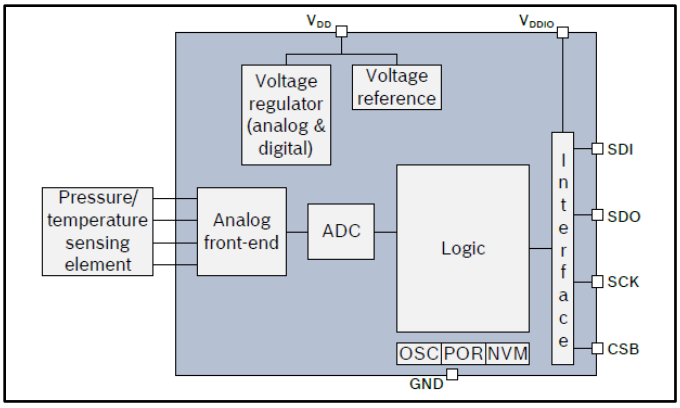
Εικόνα 61. Serial Communicator CH341T - HC-12



Εικόνα 62. HC-12 to UART to USB

2.13 BMP 280 Temperature and Pressure Sensor

Ο αισθητήρας BMP 280 είναι ένας αισθητήρας πίεσης σε συνδυασμό με ένα κύκλωμα ASIC (Application Specific Integrated Circuit) που πραγματοποιεί A/D (Analog to Digital) μετατροπές και παρέχει τα δεδομένα της μετατροπής όπως επίσης και συγκεκριμένα δεδομένα μέσω μιας ψηφιακής διεπαφής.



Εικόνα 63. Μπλοκ Διάγραμμα BMP280 [30]

Ένα σετ ρυθμίσεων προσαρμόζει τον αισθητήρα στις απαιτήσεις της εφαρμογής. Οι ρυθμίσεις είναι ένας συνδυασμός μετρήσεων πίεσης και μετρήσεων θερμοκρασίας ο οποίος χαρακτηρίζεται από τον ρυθμό της δειγματοληψίας με τον μέγιστο να είναι 16 φορές (x16) .

Oversampling setting	Pressure oversampling	Typical pressure resolution	Recommended temperature oversampling
Pressure measurement skipped	Skipped (output set to 0x80000)	–	As needed
Ultra low power	×1	16 bit / 2.62 Pa	×1
Low power	×2	17 bit / 1.31 Pa	×1
Standard resolution	×4	18 bit / 0.66 Pa	×1
High resolution	×8	19 bit / 0.33 Pa	×1
Ultra high resolution	×16	20 bit / 0.16 Pa	×2

Πίνακας 9. Ανάλυση πίεσης σε συνάρτηση με oversampling, BMP280 [30]

osrs_t[2:0]	Temperature oversampling	Typical temperature resolution
000	Skipped (output set to 0x80000)	–
001	×1	16 bit / 0.0050 °C
010	×2	17 bit / 0.0025 °C
011	×4	18 bit / 0.0012 °C
100	×8	19 bit / 0.0006 °C
101, 110, 111	×16	20 bit / 0.0003 °C

Πίνακας 10. Ανάλυση θερμοκρασίας σε συνάρτηση με oversampling, BMP280 [30]

Η πίεση του περιβάλλοντος είναι ένα μέγεθος ευμετάβλητο , για το λόγο αυτό χρησιμοποιείται ένα φίλτρο ώστε να περιορίσει τις μεταβολές και να παράγει ο αισθητήρας ένα αποτέλεσμα χωρίς έντονες διακυμάνσεις. Το φίλτρο αυτό ονομάζεται IIR και ακολουθεί την συνάρτηση:

$$data\ Filtered\ = \frac{(data\_filtered\_old\ )*(\ filter\_coefficient - 1)+data\_ADC}{filter\_coefficient}$$

Κάθε επόμενη μέτρηση θα λάβει υπόψη την προηγούμενη, ενώ στην συνάρτηση ενεργεί και η τιμή του φίλτρου με διακύμανση από 2 έως 16 όπως φαίνεται στον πίνακα 11:

Filter coefficient	Samples to reach ≥75 % of step response
Filter off	1
2	2
4	5
8	11
16	22

Πίνακας 11. Data filtering, BMP280 [30]

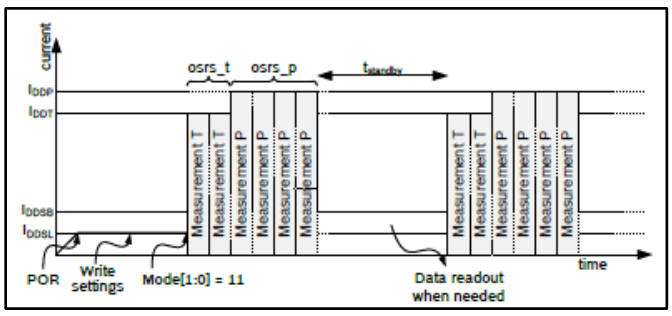
Ο κατασκευαστής για να μεγιστοποιήσει την απόδοση του αισθητήρα προτείνει τις παρακάτω ρυθμίσεις, ανάλογα με την εφαρμογή.

Use case	Mode	Over-sampling setting	osrs_p	osrs_t	IIR filter coeff. (see 3.3.3)	I_{DD} [µA] (see 3.7)	ODR [Hz] (see 3.8.2)	RMS Noise [cm] (see 3.5)
handheld device low-power (e.g. Android)	Normal	Ultra high resolution	×16	×2	4	247	10.0	4.0
handheld device dynamic (e.g. Android)	Normal	Standard resolution	×4	×1	16	577	83.3	2.4
Weather monitoring (lowest power)	Forced	Ultra low power	×1	×1	Off	0.14	1/60	26.4
Elevator / floor change detection	Normal	Standard resolution	×4	×1	4	50.9	7.3	6.4
Drop detection	Normal	Low power	×2	×1	Off	509	125	20.8
Indoor navigation	Normal	Ultra high resolution	×16	×2	16	650	26.3	1.6

Πίνακας 12. Προτεινόμενες ρυθμίσεις κατασκευαστή, BMP280 [30]

Έτσι για παράδειγμα για μια συσκευή χεριού σε δυναμική κατάσταση ο κατασκευαστής προτείνει σε Normal Mode, x4 δειγματοληψία για την πίεση , x1 δειγματοληψία για την θερμοκρασία και συντελεστή φίλτρου x16 ενώ παρέχει πληροφορίες για την ρύθμιση την κατανάλωση ρεύματος όπως επίσης και τον ρυθμό ανανέωσης των μετρήσεων.

Κατά την διάρκεια της λειτουργίας Normal ο αισθητήρας πραγματοποιεί μετρήσεις διαρκώς οι οποίες χαρακτηρίζονται από ένα χρονικό πλαίσιο που ο αυτός είναι ανενεργός . Αυτό το χαρακτηριστικό παραμετροποιεί την λειτουργία του ώστε να είναι ιδανική ανά περίπτωση. Έτσι για παράδειγμα με ένα μεγάλο χρονικό διάστημα ανενεργό, ελαχιστοποιεί την κατανάλωση ενώ αντιθέτως μεγιστοποιεί τον ρυθμό ανανέωσης.



Εικόνα 64. Normal Mode - $t_{standby}$ [30]

Οι διαθέσιμες επιλογές και ο ρυθμός ανανέωσης για κάθε επιλογή δίνονται από τον πίνακα που ακολουθεί [30] .

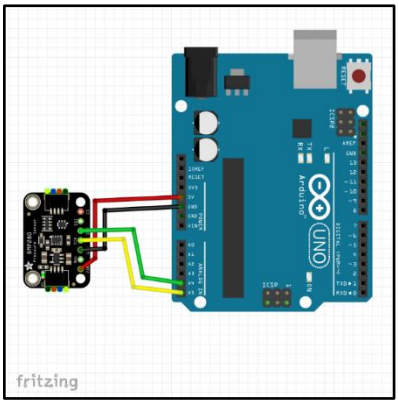
Oversampling setting	$t_{standby}$ [ms]							
	0.5	62.5	125	250	500	1000	2000	4000
Ultra low power	166.67	14.71	7.66	3.91	1.98	0.99	0.50	0.25
Low power	125.00	14.29	7.55	3.88	1.97	0.99	0.50	0.25
Standard resolution	83.33	13.51	7.33	3.82	1.96	0.99	0.50	0.25
High resolution	50.00	12.20	6.92	3.71	1.92	0.98	0.50	0.25
Ultra high resolution	26.32	10.00	6.15	3.48	1.86	0.96	0.49	0.25

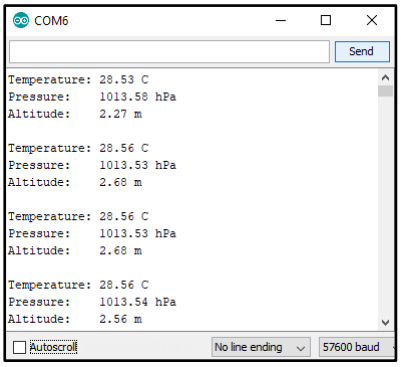
Πίνακας 13. Oversampling και $t_{standby}$, BMP280 [30]

Το module που χρησιμοποιείται ως αισθητήρας μέτρησης πίεσης και θερμοκρασίας είναι το GY BME P/280, συνδέεται με το πρωτόκολλο IIC ενώ δέχεται τάση τροφοδοσίας μέχρι 5 Volt DC. Στην εικόνα 60 φαίνεται η συνδεσμολογία του.

ErriezBMX280_snip

```
#include <Wire.h>
#include <ErriezBMX280.h>

#define SEA_LEVEL_PRESSURE_HPA      1013.85

ErriezBMX280 bmx280 = ErriezBMX280(0x76);


void setup(){

    Serial.begin(57600);
    // Initialize I2C bus
    Wire.begin();
    Wire.setClock(400000);
    bmx280.begin();
    bmx280.setSampling(BMX280_MODE_NORMAL,    // SLEEP, FORCED, NORMAL
                       BMX280_SAMPLING_X1,    // Temp:  NONE, X1, X2, X4, X8, X16
                       BMX280_SAMPLING_X4,    // Press: NONE, X1, X2, X4, X8, X16
                       BMX280_SAMPLING_NONE,  // Hum:   NONE, X1, X2, X4, X8, X16 (BME280)
                       BMX280_FILTER_X16,     // OFF, X2, X4, X8, X16
                       BMX280_STANDBY_MS_0_5);// 0_5, 10, 20, 62_5, 125, 250, 500, 1000
}

void loop(){

    Serial.print(F("Temperature: "));
    Serial.print(bmx280.readTemperature());
    Serial.println(" C");

    Serial.print(F("Pressure:    "));
    Serial.print(bmx280.readPressure() / 100.0F);
    Serial.println(" hPa");

    Serial.print(F("Altitude:    "));
    Serial.print(bmx280.readAltitude(SEA_LEVEL_PRESSURE_HPA));
    Serial.println(" m");

    Serial.println();

    delay(500);
}
```





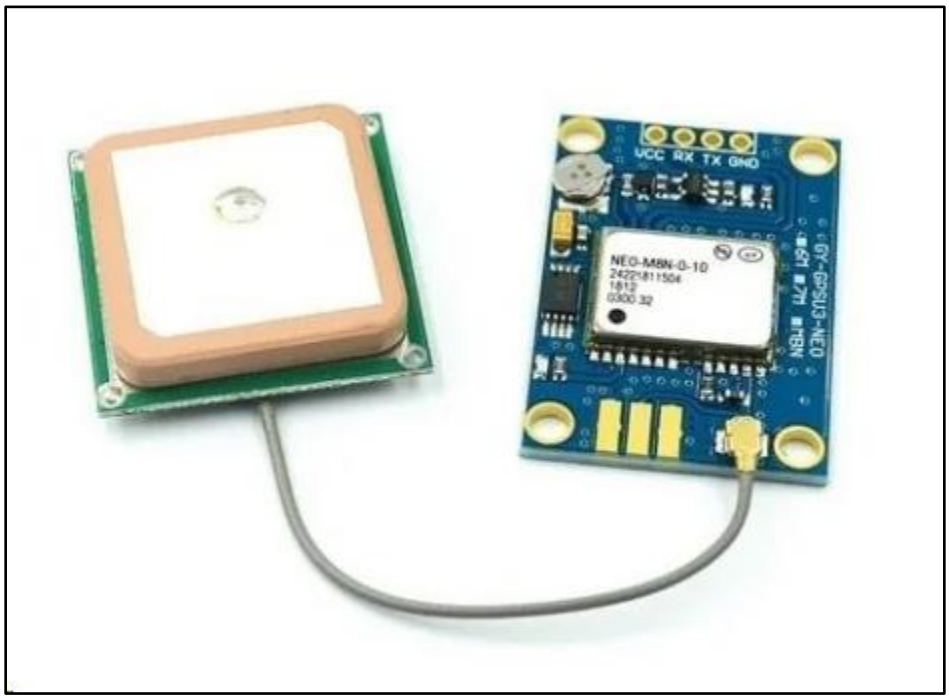
Εικόνα 65. Κώδικας , συνδεσμολογία και αποτελέσματα

Η βιβλιοθήκη που χρησιμοποιείται είναι η ErriezBMX280.h και το πρόγραμμα οδήγησης παρουσιάζεται παρακάτω. Αρχικά δηλώνεται η τρέχουσα πίεση στην επιφάνεια της θάλασσας. Ακολουθεί η IIC διεύθυνση του αισθητήρα. Ρυθμίζεται η ταχύτητα του IIC στα 400 KHz. Γίνεται δημιουργία του object του BMP280 και ακολουθούν οι ρυθμίσεις του, όπως αυτές προαναφέρθηκαν. Στο void loop γίνεται η εκτύπωση των αποτελεσμάτων στο Serial Monitor του IDE. Η θερμοκρασία σε βαθμούς κελσίου, η πίεση μετατρέπεται σε hPa (hectopascal) ενώ το εκτιμώμενο υψόμετρο προκύπτει σε συνάρτηση με την πίεση αναφοράς .

2.14 GPS

2.14.1 GPS Receiver

Ο GPS receiver που χρησιμοποιείται στην κατασκευή είναι το module GY-GPSV3-NEO το οποίο βασίζεται στο chip της U-Blox NEO-M8N. Ο δέκτης είναι ικανός να λάβει πληροφορίες από ως και 3 GNSS (Global Navigation Satellite System), GPS, Galileo, GLONASS, BeiDou . Έχει λήψη από έως και 72 κανάλια ενώ η ευαισθησία κυμαίνεται στα -167 dbm.

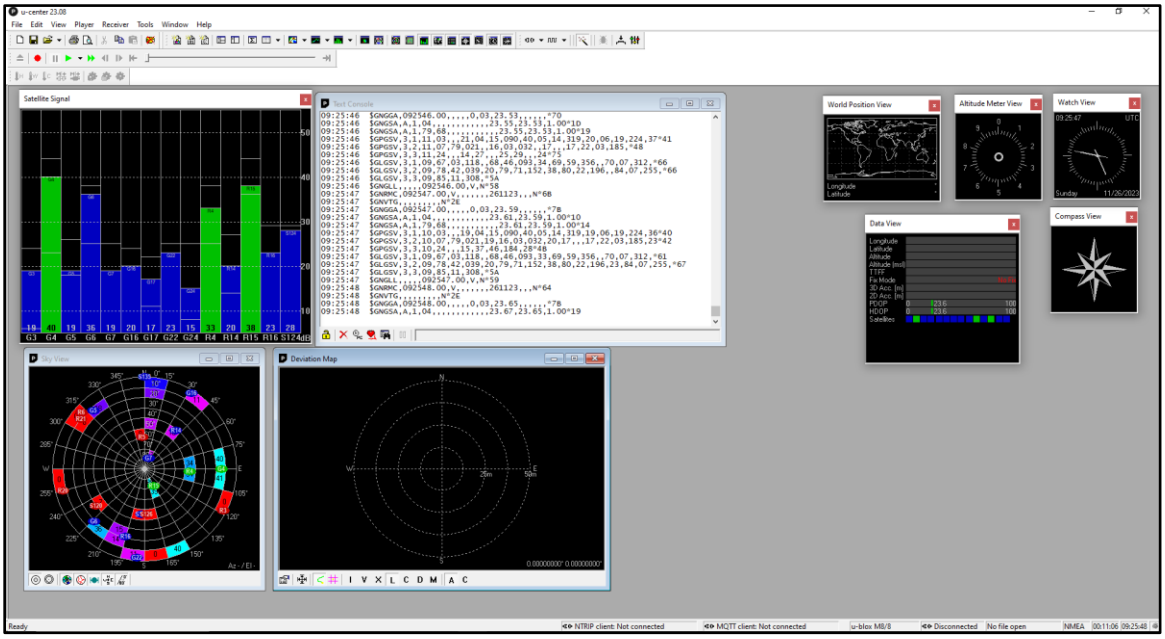


Εικόνα 66. Το GY-GPSV3-NEO

Συνδέεται με κεραία τύπου patch , ενώ διαθέτει ενσωματωμένη μπαταρία για να μειώσει τον χρόνο ΄΄ κλειδώματος ΄΄ δορυφόρων. Η συχνότητα λειτουργίας για το GPS είναι η L1C/A στα 1575.42 MHz. Το module ακολουθεί την τεχνολογία UART και τροφοδοτείται με 5V DC. Η μετάδοση είναι σειριακή στα 9600 bps. Τεχνικές πληροφορίες και αναλυτικά χαρακτηριστικά του chip της U-Blox μπορούν να ανακτηθούν από τον ιστότοπο https://www.u-blox.com/en/product/neo-m8-series .

2.14.2 GPS Software – U Center (U-Blox)

Το software που χρησιμοποιείται για να επεξεργαστεί τις πληροφορίες του δέκτη του GPS είναι και αυτό της U-Blox και είναι το U-Center. Παρέχεται δωρεάν από τον κατασκευαστή του chip και παρέχει στον χρήστη ένα πακέτο από πληροφορίες ώστε να δει , να αποθηκεύσει και ένα επεξεργαστεί τα δεδομένα που παρέχονται από τους δορυφόρους .

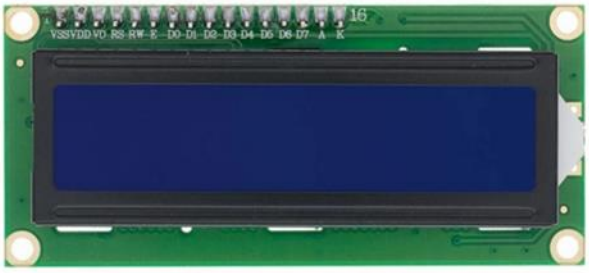


Εικόνα 67. Το U - Center

Αναλυτικά χαρακτηριστικά για το πρόγραμμα μπορεί κανείς να βρει στον ιστότοπο της κατασκευάστριας εταιρίας στην διεύθυνση https://content.u-blox.com/sites/default/files/u-center_Userguide_UBX-13005250.pdf .

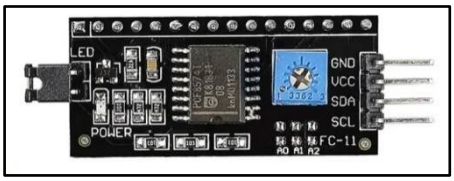
2.15 Οθόνη LCD 16x2 (HITACHI U44780)

Η οθόνη που χρησιμοποιείται στην κατασκευή είναι μια LCD (Liquid Crystal Display) 2 γραμμών και 16 στηλών . Ο controller που διαχειρίζεται το display είναι ο Hitachi U44780 , ικανός να λειτουργήσει την οθόνη σε παράλληλη συνδεσμολογία είτε με σύνδεση 8 bit ή 4 bit, με απαίτηση, σε pins σύνδεσης με τον μικροελεγκτή, 11 και 7 αντίστοιχα. Η οθόνη είναι ικανή αν αποτυπώσει οποιονδήποτε ASCII χαρακτήρα ενώ μπορούν να σχηματιστούν και ειδικοί χαρακτήρες.



Εικόνα 68. LCD 16x2

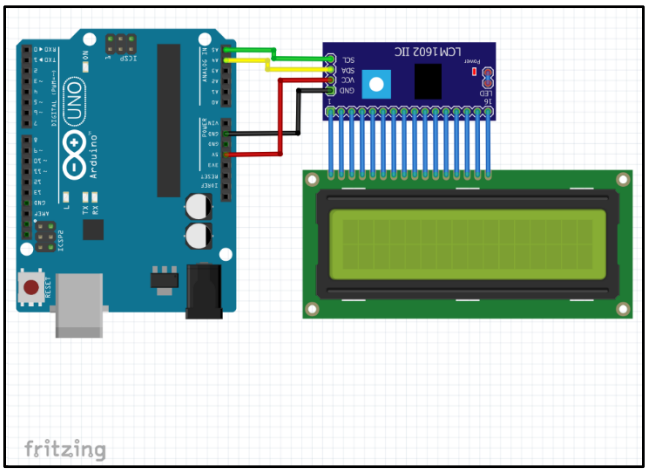
Για λόγους ευκολίας και οικονομίας σε pin χρησιμοποιείται ένας IIC controller, ο PCF8574, ο οποίος μετατρέπει τις εντολές του μικροελεγκτή από IIC σε παράλληλα δεδομένα. Έχει την δυνατότητα να ρυθμίζει το Contrast της οθόνης όπως και το Backlight. Οδηγείται από τον μικροελεγκτή με τροφοδοσία +5V [31].



Εικόνα 69. PCF8574 IIC to parallel

https://www.ultralibrarian.com/2020/08/27/pcf8574-and-i-o-bus-expanders-ulc

Η συνδεσμολογία με το Arduino Uno παρουσιάζεται στην εικόνα 68 και απαιτεί 4 καλώδια, 2 για το IIC και τροφοδοσία και γείωση. Η βιβλιοθήκη που χρησιμοποιείται είναι η LiquidCrystal_I2C.



Εικόνα 70. Η συνδεσμολογία του IIC LCD 16x2 με το Arduino Uno

Το Sketch του IDE είναι πολύ απλό και παρουσιάζεται παρακάτω:

```
#include <LiquidCrystal_I2C.h>

LiquidCrystal_I2C lcd(0x27, 16, 2);


void setup() {

 lcd.init();

 lcd.backlight();

 lcd.clear();

 lcd.setCursor(0, 0);

 lcd.print("Telemetry Rcvr");

 lcd.setCursor(0, 1);

 lcd.print("Station  PG.V1.4");

 delay(3000);

}

Void loop {

// Do nothing here

}
```

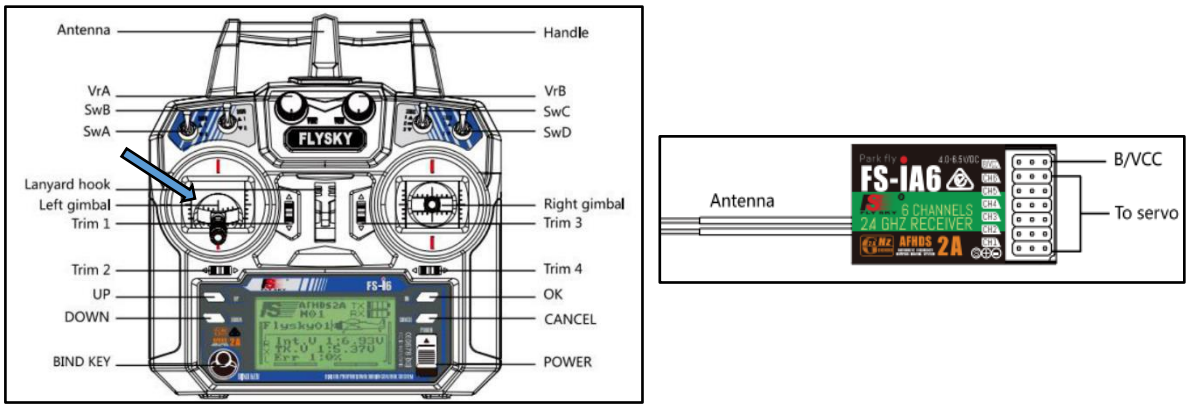
Εικόνα 71. Το Sketch για την λειτουργία του LCD 16x2

Στις δύο πρώτες γραμμές δηλώνεται η βιβλιοθήκη και δημιουργείται το object στην διεύθυνση 0x27 για το IIC για οθόνη 16x2. Με τις εντολές *.init*, *.backlight* και *.clear* αρχικοποιείται η οθόνη, ενεργοποιείται το backlight και καθαρίζουν τυχόν χαρακτήρες ώστε να απεικονισθούν οι επόμενοι. Με την *.setCursor* ορίζουμε που θα αποτυπωθεί ο χαρακτήρας στην οθόνη ενώ με την *.print* τυπώνουμε τον ASCII χαρακτήρα.

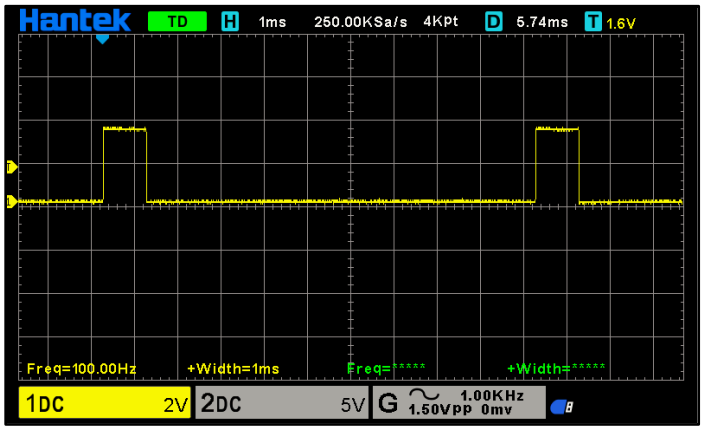
3. Τρόπος λειτουργίας

3.1 Τηλεχειρισμός

Όπως έχει αναφερθεί ο χειριστής μεταβάλλει ένα ποτενσιόμετρο στο control panel (left gimbal, throttle) αυξομειώνοντας το εύρος του παλμού της κυματομορφής όπως αυτή εικονίζεται παρακάτω. Η ελάχιστη τιμή του παλμού είναι 1000μsec και η μέγιστη αντίστοιχη 2000 μsec. Η συχνότητα της κυματομορφής είναι συνήθως 50 Hz (τις περισσότερες φορές οδηγεί κατευθείαν servo), μπορεί όμως να φτάσει και τα 400 Hz, ανάλογα τον τηλεχειρισμό. Στην εικόνα που ακολουθεί φαίνεται το εύρος του παλμού όπως και η συχνότητα της κυματομορφής που στην περίπτωση μας είναι 100Hz. Οι μετρήσεις έχουν προκύψει από το κανάλι 3 (channel 3 – Throttle) του δέκτη του τηλεχειρισμού με την βοήθεια παλμογράφου.



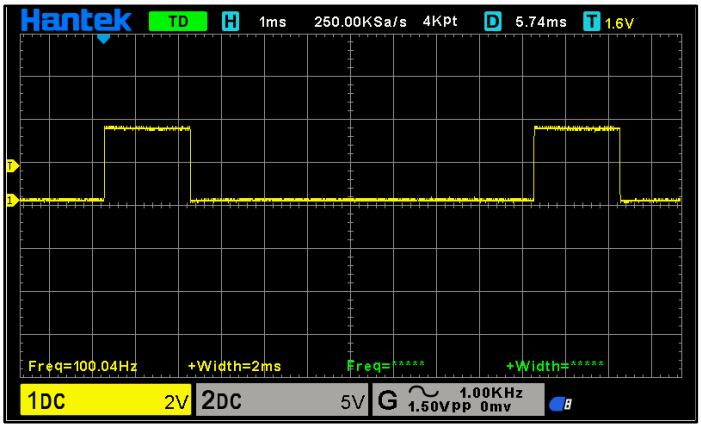
Εικόνα 72. Ο τηλεχειρισμός και ο δέκτης του τηλεχειρισμού [23]



Εικόνα 73. Κυματομορφή εξόδου του δέκτη , 1000 μsec και συχνότητας 100 Hz

Κάθε ποτενσιόμετρο του control panel (εικόνα 72) μεταβάλλει διαφορετική έξοδο στον δέκτη του τηλεχειρισμού . Έτσι το channel 1 είναι το Roll του control panel, το channel 2 είναι το Pitch, το channel 3 είναι το Throttle και το channel 4 είναι το Yaw. Αν εξερεθεί το Throttle τα

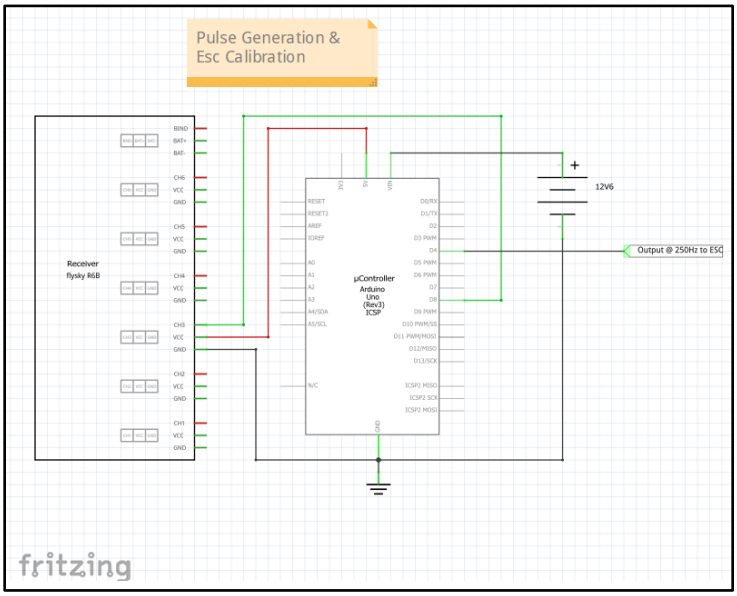
υπόλοιπα ποτενσιόμετρα έχουν κεντρική τιμή τα 1500 μsec και μεταβάλλονται από 500 μsec αριστερά (min) και δεξιά(max) αποδίδοντας και αυτά ένα εύρος παλμού από 1000μsec έως



Εικόνα 74. Κυματομορφή εξόδου του δέκτη , 2000 μsec και συχνότητας 100 Hz

2000μsec. Το Throttle σχεδόν πάντα οδηγεί έναν ESC (Electronic Speed Controller) ο οποίος με την σειρά του ρυθμίζει την ταχύτητα περιστροφής του μοτέρ. Οι υπόλοιπες εντολές οδηγούν Servo μηχανισμούς που στην περίπτωση ενός αεροπλάνου κινούν επιφάνειες aileron, elevator, rudder. Για το Quadcopter είναι διαφορετικά. Το Throttle διεγείρει τα 4 ESC που οδηγούν τα 4 μοτέρ, ενώ τα roll, pitch, yaw, χρησιμοποιούνται για να αλλάξουν την ταχύτητα περιστροφής συγκεκριμένων μοτέρ ώστε να επιτύχουν roll, pitch, yaw. Ο τρόπος που θα επιτευχθεί αυτό θα εξηγηθεί στις παρακάτω παραγράφους .

Το σίγουρο είναι πως πρέπει να βρεθεί τρόπος να αποκωδικοποιηθούν οι τέσσερις κυματομορφές από τον μικροελεγκτή, να μορφοποιηθούν και να αποδοθούν εκ νέου με την κατάλληλη τιμή ώστε να επιτύχουν roll, pitch, yaw ή throttle. Αυτό επιτυγχάνεται με το κύκλωμα που φαίνεται στην εικόνα 75.



Εικόνα 75. Το κύκλωμα για την ανάγνωση και αναπαραγωγή του παλμού

Αριστερά διακρίνεται ο δέκτης του τηλεχειρισμού και το κανάλι 3 (Channel 3 - Throttle) που οδηγείται σε μία ψηφιακή είσοδο του Arduino (D8). Με την βοήθεια κώδικα και την χρήση interrupts θα μετρήσουμε το εύρος του παλμού εισόδου, θα μορφοποιήσουμε τον παλμό και τελικά θα τον οδηγήσουμε σε ένα ESC για να ελέγξουμε την ταχύτητα περιστροφής του μοτέρ. Η διαδικασία είναι κοινή και για τις 4 κυματομορφές. Η έξοδος D4 τελικά δίνει μια κυματομορφή συχνότητας 250Hz, μεταβαλλόμενου εύρους παλμού 1000μsec έως 2000μsec. Για λόγους απλότητας θα περιγραφεί μόνο για το Throttle.

3.2 Το πρόγραμμα για την αναπαραγωγή του παλμού εισόδου

Στο πρόγραμμα που φαίνεται στην εικόνα 76 αποκωδικοποιείται ο παλμός του receiver του τηλεχειρισμού και αναπαράγεται ώστε να οδηγήσει τον ESC. Στις παρακάτω γραμμές επεξηγείται ο κώδικας του προγράμματος. Στις 3 πρώτες γραμμές δηλώνονται όλες οι μεταβλητές

ESC_calibration | Arduino 1.6.9

File Edit Sketch Tools Help

ESC_calibration §

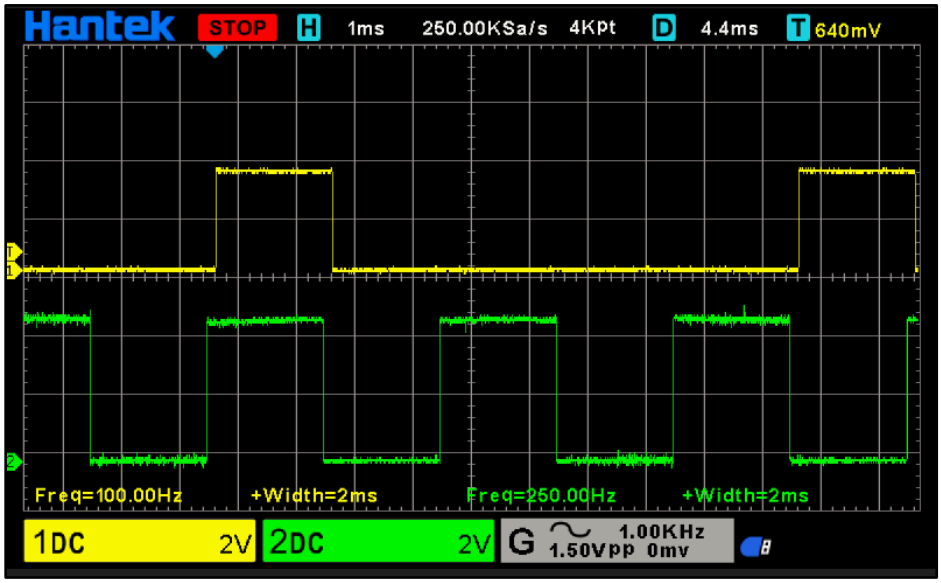
```
int throttle_input,throttle ;
unsigned long timer_1,motor1,esc1waveform, current_time, esc_loop_timer, pulse_time, zero_timer ;
byte last_channel_1;


void setup() {
  Serial.begin(250000);
  PCICR |= (1 << PCIE0);                                    //Pin Change Interrupt Control Register E0 PCINT 0 - 7
  PCMSK0 |= (1 << PCINT0);                                  //Set PCINT0 (digital input 8) to trigger an interrupt on state change,  Throttle input
  DDRD |= B00010000;                                        // Set 4 output
  zero_timer = micros();
  }

void loop() {

     motor1 = throttle_input;
     if (motor1 <1000) motor1 = 1000;
     if (motor1 >2000) motor1 = 2000;

     while(micros() - zero_timer < 4000);                   //250 Hz    program loop
     zero_timer = micros();

     PORTD |= B00010000;                                    // Set 4 High

     esc1waveform = micros() + motor1;                      // μέτρα χρόνο που απαιτείται + έυρος παλμού για κάθε esc - motor

     while ( (PORTD & B00010000 ) != 0) {                   // μείνε στο loop μέχρι και η 4 να γίνει LOW
     esc_loop_timer = micros();
     if(esc1waveform <= esc_loop_timer)PORTD &= B11101111;  //μέτρα χρόνο, μόλις γίνει ίσος με τον παραπάνω που υπολογίστηκε γύρνα την έξοδο σε LOW
     }
     Serial.println(motor1);
   }
ISR(PCINT0_vect){
  current_time = micros();
  //  Throttle
  if(PINB & B00000001){
    if(last_channel_1 == 0){
      last_channel_1 = 1;
      timer_1 = current_time;
    }
  }
  else if(last_channel_1 == 1){
    last_channel_1 = 0;
    throttle_input = current_time - timer_1;
  }
}
```

Εικόνα 76. Ο κώδικας για την ανάγνωση και αναπαραγωγή του παλμού

που θα χρησιμοποιηθούν στον κώδικα. Στο void setup οι εντολές που θα εκτελεστούν μια φορά. Στην γραμμή 11 γίνεται initialize η σειριακή επικοινωνία με τον υπολογιστή και το serial monitor

στα 250 kbps. Οι γραμμές 12,13 έχουν περιγραφεί με λεπτομέρεια στην παράγραφο με τα Interrupts. Στην γραμμή 14 το pin D4 τίθεται ως έξοδος ενώ στην 15 γίνεται reset το χρονόμετρο στην μεταβλητή zero_timer. Στην γραμμή 37 ξεκινά η Interrupt Service Routine για να διαβάσουμε το D8 . Στην γραμμή 38 γίνεται reset το χρονόμετρο και αρχίζει να μετρά χρόνο μέσα στην ISR στην μεταβλητή current time. Η PCINT μπορεί να μετρά μόνο αλλαγή κατάστασης από HIGH σε LOW και ανάποδα. Έτσι στην γραμμή 40, αν υπάρχει αλλαγή κατάστασης, στην γραμμή 41 ελέγχει αν ήταν LOW την κάνει HIGH(γραμμή 42) και την σώζει στην μεταβλητή last_channel_1. Στην γραμμή 43 ξεκινά να μετρά και σώζει τον χρόνο στην μεταβλητή timer_1 .

Στην γραμμή 46, αν ήταν HIGH την κάνει LOW και την σώζει στην μεταβλητή last_channel_1. Στην γραμμή 48 αφαιρεί από τον αρχικό χρόνο τον χρόνο timer_1 και τελικά παίρνει το εύρος του παλμού που το σώζει στην μεταβλητή throttle_input. Στην γραμμή 20 σώζει την τιμή του throttle_input στην μεταβλητή motor1 που είναι και η βασική μεταβλητή. Στις γραμμές 21,22 φιλτράρει τις τιμές ώστε να περιορίζονται μεταξύ 1000μsec και 2000μsec. Στην γραμμή 24 περιμένει μέχρι να περάσουν 4000 μsec ώστε να προχωρήσει στις επόμενες εντολές. Αυτός ο χρόνος είναι η συχνότητα ανανέωσης του προγράμματος(250 Hz). Στην επόμενη γραμμή κάνει reset το χρονόμετρο στην μεταβλητή zero_timer. Στην γραμμή 27 κάνει HIGH την έξοδο D4. Στην γραμμή 29 ορίζει την μεταβλητή esc1waveform και της δίνει τιμή όσο του motor1 και του χρόνου που πέρασε μέχρι εκείνη την στιγμή. Στην γραμμή 31 ξεκινά ένα loop υπό συνθήκη. Κάνει reset το χρονόμετρο στην μεταβλητή esc_loop_timer. Με την if στην γραμμή 33 συγκρίνει την μεταβλητή esc1waveform και την esc_loop_timer. Όταν γίνουν ίσες θέτει την έξοδο D4 LOW και τερματίζει το loop της while. Ο παλμός που έχει διαμορφωθεί έχει διάρκεια όσο το motor1 και συχνότητα 250 Hz. Ακολουθεί μια Serial print που χρησιμοποιείται για debugging.



Εικόνα 77. Η κυματομορφή εισόδου και εξόδου

Η κυματομορφή 1 (κίτρινη) είναι του Rx του τηλεχειρισμού με εύρος παλμού 2000μsec και συχνότητα 100Hz η οποία εισάγεται στον μικροελεγκτή, επεξεργάζεται και προκύπτει ή κυματομορφή 2 (πράσινη), έξοδος του pin D4, ίδιου εύρους αλλά συχνότητας 250 Hz. Αυτή η

κυματομορφή είναι η βασική που θα διαχειρίζεται όλα το μοτέρ του Quadcopter μέσω των ESC's. Αξίζει να σημειωθεί ότι επιθυμούμε η συχνότητα της κυματομορφής εξόδου που θα οδηγήσει τα ESC να είναι τουλάχιστον 200 Hz. Με αυτόν τον τρόπο έχουμε καλύτερο χειρισμό των μοτέρ και τελικά πιο ομαλή πτήση του Quadcopter.

3.3 ESC Throttle Range Setting

Διαφορετικά τηλεχειριστήρια έχουν σήματα ελέγχου ESC σε διαφορετικές περιοχές. Μπορούν να βρίσκονται στο εύρος από 1000 μS έως 2000 μS ή 500 μS έως 2500 μS στα 50 έως 60 ή παραπάνω Hz. Κάθε φορά που συνδέετε ένα ESC σε ένα νέο τηλεχειριστήριο ή μικροελεγκτή τότε πρέπει να ΄΄μάθει΄΄ το ESC σχετικά με τον έλεγχο εύρος σήματος που αντιστοιχεί στο μηδέν και στο πλήρες throttle. Έτσι λοιπόν υπάρχει μια διαδικασία που πρέπει να ακολουθηθεί, ώστε να γίνουν calibrate τα ESC, η οποία και περιγράφεται στις γραμμές που ακολουθούν.

Διαδικασία calibration ESC΄s :

1. Ενεργοποιήστε τον πομπό και μετακινήστε τη θέση του throttle στο μέγιστο.

2. Συνδέστε την μπαταρία στο ESC. (Ο κινητήρας BLDC θα πρέπει να συνδεθεί στο ESC). Ο ESC θα κάνει το αυτοέλεγχο. Εκπέμπετε ειδικός τόνο "beep -beep". Μετά από αυτό θα εκπέμψει αριθμό ηχητικών τόνων που αντιστοιχούν στον αριθμό των cells της μπαταρίας .

3. Περιμένετε άλλα 2 δευτερόλεπτα. Το ESC θα εκπέμψει ηχητικό σήμα δύο φορές που σημαίνει ότι το ESC έχει κλειδώσει τη μέγιστη θέση throttle.

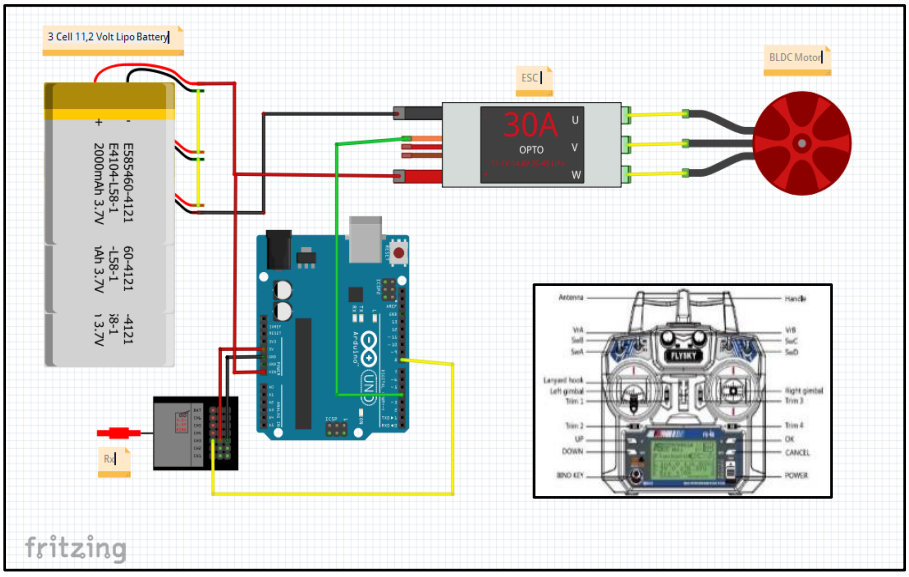
4. Τώρα μέσα στα επόμενα 5 δευτερόλεπτα μετακινήστε το throttle στη θέση μηδέν και περιμένετε 1 δευτερόλεπτο.

Το ESC θα εκπέμψει ένα παρατεταμένο ηχητικό σήμα που υποδεικνύει ότι έχει κλειδώσει τη μηδενική θέση του throttle.

Αυτή η ρύθμιση throttle γκαζιού καταγράφεται μόνιμα στο ESC.

Από την στιγμή **που έχει γίνει calibration ο ESC** το μοτέρ είναι έτοιμο να δεχθεί την **προπέλα** και όλο το setup όπως φαίνεται στην εικόνα 69 μπορεί να ενεργοποιηθεί.

Αξίζει να σημειωθεί ότι αλλάζοντας την συνδεσμολογία ενός ζευγαριού καλωδίων μεταξύ του ESC και του BLDC motor μπορούμε να αλλάξουμε την φορά περιστροφής του μοτέρ από αριστερόστροφα – δεξιόστροφα ή ανάποδα . Αυτό είναι πολύ σημαντικό για το Quadcopter και θα εξηγηθεί παρακάτω. Το κύκλωμα της εικόνας 78 θα χρησιμοποιηθεί για να πάρουμε τις χαρακτηριστικές και των τεσσάρων μοτέρ, Thrust vs Throttle και Current vs Throttle, από τις οποίες θα βγάλουμε χρήσιμα συμπεράσματα για το payload και το μέγεθος της μπαταρίας του Quadcopter [32].



Εικόνα 78. Το Setup για τις χαρακτηριστικές των μοτέρ

3.4 Χαρακτηριστικές των τεσσάρων Μοτέρ , το setup

Το setup της εικόνας 71 χρησιμοποιείται για να μετρήσουμε την απόδοση και την κατανάλωση των τεσσάρων μοτέρ. Το μοτέρ δένεται σε βάρος μεγαλύτερο του 1,5 kg. Και όλη η κατασκευή τοποθετείται σε ζυγαριά η οποία γίνεται tare. Από την παραπάνω συνδεσμολογία οδηγούμε το μοτέρ και από τον υπολογιστή με την βοήθεια της Serial.print(motor1) για διάφορες τιμές του Throttle καταγράφουμε το Thrust από την ζυγαριά και την κατανάλωση του ρεύματος από το τροφοδοτικό. Αξίζει να σημειωθεί ότι οι μετρήσεις της απόδοσης των μοτέρ έγιναν όλες με τάση αναφοράς τα **12.2 Volt**.

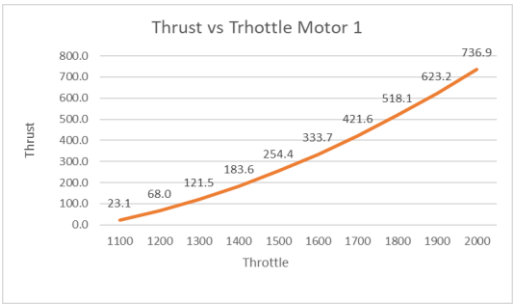
Εικόνα 79. Το setup του πειράματος

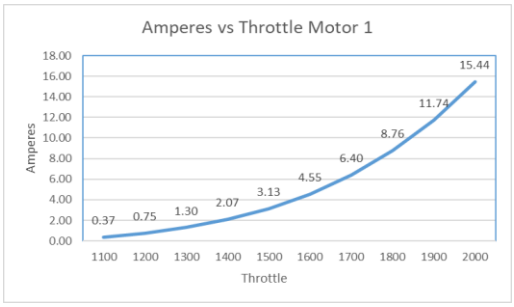
Τροφοδοτικό, ζυγαριά, υπολογιστής, τηλεχειρισμός και η κατασκευή του μοτέρ με το ESC και τις συνδέσεις.

Εικόνα 80. Τα 4 τέσσερα μοτέρ με τις προπέλες τους και τα ESC's

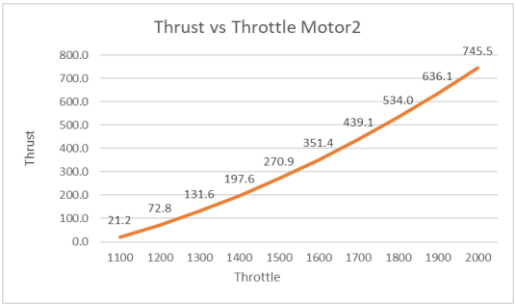
3.4.1 Πίνακες και γραφικές παραστάσεις των τεσσάρων μοτέρ

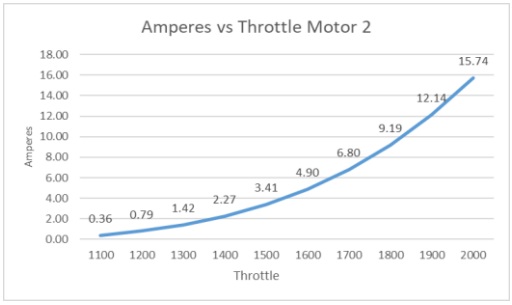
Motor 1		
Throttle	Thrust	Amperes
1100	23.1	0.37
1200	68.0	0.75
1300	121.5	1.30
1400	183.6	2.07
1500	254.4	3.13
1600	333.7	4.55
1700	421.6	6.40
1800	518.1	8.76
1900	623.2	11.74
2000	736.9	15.44



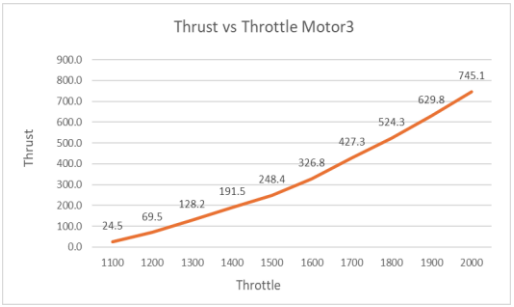


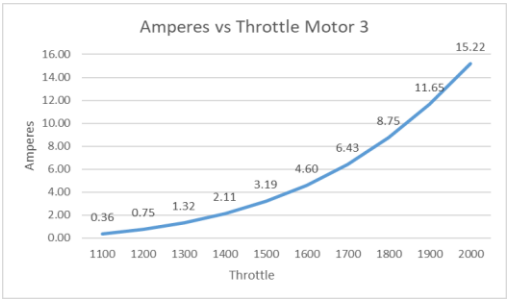
Motor 2		
Throttle	Thrust	Current
1100	21.2	0.36
1200	72.8	0.79
1300	131.6	1.42
1400	197.6	2.27
1500	270.9	3.41
1600	351.4	4.90
1700	439.1	6.80
1800	534.0	9.19
1900	636.1	12.14
2000	745.5	15.74



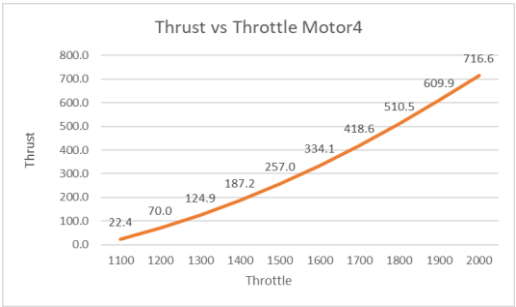


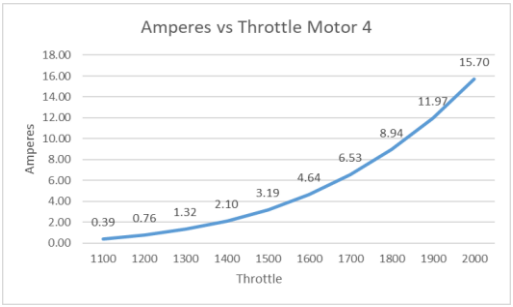
Motor 3		
Throttle	Thrust	Current
1100	24.5	0.36
1200	69.5	0.75
1300	128.2	1.32
1400	191.5	2.11
1500	248.4	3.19
1600	326.8	4.60
1700	427.3	6.43
1800	524.3	8.75
1900	629.8	11.65
2000	745.1	15.22





Motor 4		
Throttle	Thrust	Current
1100	22.4	0.39
1200	70.0	0.76
1300	124.9	1.32
1400	187.2	2.10
1500	257.0	3.19
1600	334.1	4.64
1700	418.6	6.53
1800	510.5	8.94
1900	609.9	11.97
2000	716.6	15.70





Εικόνα 81. Χαρακτηριστικές μοτέρ, Thrust vs Throttle και Amperes vs Throttle

3.4.2 Συμπεράσματα

Από τα παραπάνω δεδομένα προκύπτει ο πίνακας 14 στον οποίο φαίνεται το μέσο μέγιστο Thrust και η μέση μέγιστη κατανάλωση ρεύματος.

Reference Voltage 12V2		
Motor	Max Thrust	Max Amps
Motor 1	736.9	15.44
Motor 2	745.5	15.74
Motor 3	745.1	15.22
Motor 4	716.6	15.70
Average	736.0	15.50

Πίνακας 14. Μέγιστες αποδόσεις και κατανάλωση των 4 μοτέρ

Κατά αυτόν τον τρόπο μπορούμε να πούμε ότι για τάση 12V2 το μέγιστο Thrust των τεσσάρων μοτέρ είναι σχεδόν 3 kg. ενώ οι απαιτήσεις σε ρεύμα για αυτό το Thrust είναι περίπου 60 Amps. Γνωρίζοντας ότι το βάρος του Quadcopter στη βασική διαμόρφωση είναι μεταξύ 900 – 1000 grams(παρακάτω θα αναφερθούμε με ακρίβεια στο βάρος του σκάφους) μαζί με μια μπαταρία 3cell , 2200 mAh (μπαταρία αναφοράς , θα δούμε παρακάτω τι μπαταρία θα επιλέξουμε), παρατηρούμε ότι για Throttle 1500 μsec το Quadcopter δύναται να κάνει hover με μια κατανάλωση σε ρεύμα περίπου 13 Amps.

Reference Voltage 12V2		
1500μsec	Thrust	Amps
Motor 1	254.4	3.10
Motor 2	270.9	3.40
Motor 3	248.4	3.19
Motor 4	257.0	3.20
Average	257.7	3.22
Average x4	1030.7	12.89

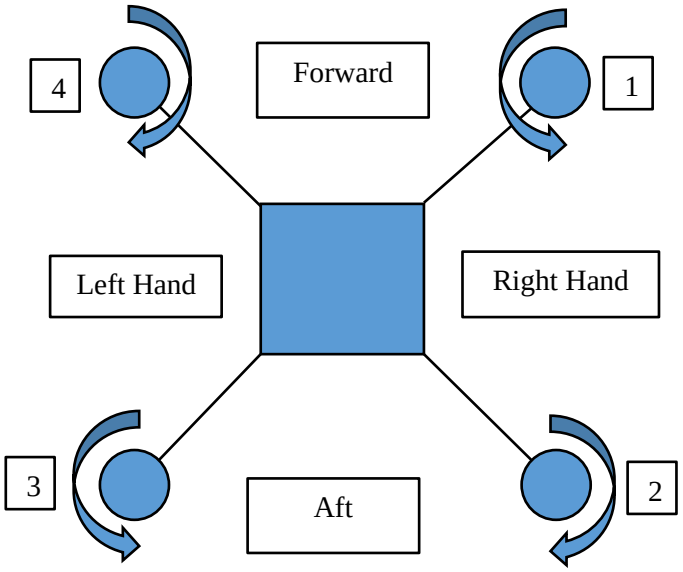
Πίνακας 15. Αποδόσεις και κατανάλωση για Throttle 50%

Η περιοχή λειτουργίας της 3cell μπαταρίας είναι από 12V6 έως 11V3, οπότε για την βασική διαμόρφωση του σκάφους μια μπαταρία 2200 mAh τουλάχιστον 20C (μέγιστο ρεύμα 44 Amps) μας εξυπηρετεί . Πρέπει να αναφερθεί ότι αλλάζοντας το payload του Quadcopter ανάλογα με τις απαιτήσεις πάντα πρέπει να λαμβάνουμε υπόψιν τους πίνακες και τις γραφικές παραστάσεις των μοτέρ ώστε να βγάζουμε σαφή συμπεράσματα για την απόδοση του σκάφους και την διάρκεια της πτήσης. Στο συγκεκριμένη συνθήκη το σκάφος θεωρητικά θα μπορεί να κάνει hover για χρόνο $t = (2{,}2\ Amps * 60\ min)/(13\ Amps) = 10.15\ min$. Αυξάνοντας το capacity μιας μπαταρίας αυξάνεται δραματικά η διάρκεια πτήσης με αντάλλαγμα όμως την αύξηση του βάρους, για παράδειγμα μια μπαταρία 3Cell, 3000mAh είναι κατά περίπου 60 γρ. βαρύτερη από μια αντίστοιχη 2200mAh.

3.5 Λειτουργία πτήσης Quadcopter

Ένα τετρακόπτερο αποτελείται από τέσσερις κινητήρες. Κάθε κινητήρας παράγει ώθηση και ροπή. Το τετρακόπτερο μπορεί να αιωρείται και να παραμένει σταθερό πάνω από το έδαφος (hover) όταν έχουν και οι τέσσερις κινητήρες την η ίδια ροπή και η συνολική ώθηση του σκάφους είναι ίση ή μεγαλύτερη από το βάρος του.

Το Quadcopter της κατασκευής είναι τύπου X shape ή αλλιώς Quad X. Η διάταξη των μοτέρ φαίνεται στην εικόνα 82. Τα μοτέρ 1 και 3 περιστρέφονται αριστερόστροφα ενώ τα μοτέρ 2 και 4 περιστρέφονται δεξιόστροφα .



Εικόνα 82. Τετρακόπτερο τύπου X

Κατά την διάρκεια λειτουργίας του Quadcopter και όταν αυτό βρίσκεται σε κατάσταση ισορροπίας το μοτέρ 1 ακυρώνει την ροπή του μοτέρ 2 ενώ αντίστοιχα το μοτέρ 3 ακυρώνει την ροπή του μοτέρ 4. Όταν θέλουμε να κινηθεί εμπρός(pitch down) αυξάνουμε την ώθηση των μοτέρ 2 και μοτέρ 3 . Το Aft μέρος του σκάφους ανυψώνεται διαταράσσοντας την ισορροπία του αναγκάζοντας το να κινηθεί εμπρός. Αντίστοιχα όταν θέλουμε να κινηθεί όπισθεν (pitch up) αυξάνουμε την ώθηση των μοτέρ 1 και μοτέρ 4. Κατά παρόμοιο τρόπο όταν θέλουμε να κινηθεί αριστερά (roll left) αυξάνουμε την ώθηση των μοτέρ 1 και μοτέρ 2 ενώ για κίνηση δεξιά (roll right) αυξάνουμε την ώθηση των μοτέρ 3 και μοτέρ 4. Τέλος για την περιστροφή του Quadcopter κατά τον άξονα z αρκεί να αυξήσουμε την ώθηση δύο ίδιας περιστροφής μοτέρ. Η αδράνεια θα περιστρέψει το σκάφος προς την αντίθετη φορά.

Η παραπάνω πληροφορία κρίνεται αναγκαία ώστε να καταλάβει κάποιος τον τρόπο ανάπτυξης του κώδικά που θα ακολουθήσει στις επόμενες παραγράφους. Η πρώτη έκδοση του προγράμματος του Quadcopter θα εξηγηθεί γραμμή – γραμμή. Έχει τέσσερις εισόδους Roll, Pitch, Throttle, Yaw και τέσσερις εξόδους για τα τέσσερα ESC's , BLDC's. Η βασική μεταβλητή του throttle_input είναι η throttle. Εισάγονται οι μεταβλητές roll_left, roll right, pitch down, pitch up. Η μέγιστη τιμή του deflection είναι στα 100 μsec. Διάφορες Serial.print χρησιμοποιούνται για debugging απενεργοποιημένες όμως διότι καθυστερούν το loop του προγράμματος. Είναι ακόμα διαθέσιμα 2 κανάλια τηλεχειρισμού για μελλοντική χρήση.

3.6 Προγραμματισμός και υλοποίηση της πρώτης έκδοσης του Quadcopter

Στις γραμμές που ακολουθούν θα εξηγηθεί ο προγραμματισμός του μικροελεγκτή βήμα – βήμα και θα παρουσιαστεί ή πρώτη λειτουργική έκδοση του Quadcopter. Το πρόγραμμα έχει πολλές ομοιότητες με το πρόγραμμα για την αναπαραγωγή του παλμού εισόδου που παρουσιάστηκε παραπάνω. Τις περισσότερες φορές είναι αναπαραγωγή συγκεκριμένου κώδικα για να υποστηρίξει και τα τέσσερα μοτέρ.

Στις πρώτες γραμμές βρίσκονται οι δηλώσεις των μεταβλητών για τα τέσσερα μοτέρ, για τον έλεγχο τους, την ενεργοποίηση και απενεργοποίηση τους καθώς και τον έλεγχο του Quadcopter για κινήσεις roll, pitch και yaw.

```
int roll_input, pitch_input, throttle_input, yaw_input, throttle, roll_right, roll_left,
    pitch_up, pitch_down, yaw_right, yaw_left, start ;

unsigned long timer_1,timer_2, timer_3, timer_4, motor1, motor2, motor3, motor4, esc1waveform,
    esc2waveform, esc3waveform , esc4waveform , current_time, esc_loop_timer, zero_timer ;
byte last_channel_1, last_channel_2, last_channel_3, last_channel_4 ;
```

Εικόνα 83. Επεξήγηση κώδικα πρώτης έκδοσης Quadcopter μέρος α΄

Στο κομμάτι void setup βρίσκεται ο κώδικας που θα εκτελεστεί μία φορά κατά την στιγμή της εφαρμογής τάσης στον μικροελεγκτή. Η Serial.begin εξυπηρετεί την επικοινωνία του μικροελεγκτή με τον υπολογιστή μέσω του serial monitor. Στις παρακάτω γραμμές ενεργοποιείται η PCINT και δηλώνονται τα pin που θα εξυπηρετήσουν το trigger του interrupt ενώ τίθενται τα digital pin 4,5,6,7 ως έξοδοί αφού θα οδηγήσουν τελικά τα ESC των μοτέρ. Τέλος γίνεται reset το χρονόμετρο micros στην μεταβλητή zero_timer και τίθεται αρχική τιμή 0 στην μεταβλητή start.

```
void setup() {
  Serial.begin(250000);
  PCICR |= (1 << PCIE0);                //Pin Change Interrupt Control Register E0 PCINT 0 - 7
  PCMSK0 |= (1 << PCINT0);              //Set PCINT0 (digital input 8) to trigger an interrupt on state change,  Roll input
  PCMSK0 |= (1 << PCINT1);              //Set PCINT1 (digital input 9)to trigger an interrupt on state change,   Pitch input
  PCMSK0 |= (1 << PCINT2);              //Set PCINT2 (digital input 10)to trigger an interrupt on state change,  Throttle input
  PCMSK0 |= (1 << PCINT3);              //Set PCINT3 (digital input 11)to trigger an interrupt on state change,  Yaw input
  // PCMSK0 |= (1 << PCINT4);           //SWC C for future use
  // PCMSK0 |= (1 << PCINT5);           //SWC B for future use
  DDRD |= B11110000;                    // Set 4, 5, 6 ,7  outputs        4=> motor1 CCW , 5=> motor2 CW , 6=> motor3 CCW , 7=> motor4 CW

  zero_timer = micros();
  start = 0;
  }
```

Εικόνα 84. Επεξήγηση κώδικα πρώτης έκδοσης Quadcopter μέρος β΄

Για τις εντολές PCIR, PCMSK και PCINT πληροφορίες υπάρχουν στο κεφάλαιο με τα Interrupts και στην εικόνα 21. Για την εντολή DDRD πληροφορίες μπορούν να ανακτηθούν από την ιστοσελίδα https://docs.arduino.cc/hacking/software/PortManipulation .

Στις επόμενες γραμμές στο void loop το πρόγραμμα θα εκτελείται διαρκώς. Εισάγεται η μεταβλητή start ή οποία θα κάνει arm και disarm τα μοτέρ. Έτσι αν το throttle είναι τέρμα κάτω και το yaw τέρμα αριστερά η start γίνεται 1. Όταν το χειριστήριο γυρίσει στην neutral τιμή , η μεταβλητή start παίρνει την τιμή 2 (arm). Όταν η start έχει την τιμή 2 και το throttle είναι τέρμα κάτω και το yaw τέρμα δεξιά η start παίρνει την τιμή 0 (disarm) . Τέλος τίθεται η μεταβλητή throttle ως βασική μεταβλητή του ελέγχου των μοτέρ.

```
31 void loop() {
32
33      if (throttle_input<=1100 && yaw_input<1050 ) start=1;
34      if (start==1 && throttle_input<=1100 && yaw_input<1520 && yaw_input>1480  ) start =2;
35      if (start==2 && throttle_input<=1100 && yaw_input>1950) start=0;
36
37      throttle = throttle_input;
```

Εικόνα 85. Επεξήγηση κώδικα πρώτης έκδοσης Quadcopter μέρος γ΄

Στην γραμμή 39 αν έχουμε κάνει arm τα μοτέρ (start = 2) εκτελείται η if και στην γραμμή 41 περιορίζουμε το throttle στα 1900 μsec. Στις επόμενες γραμμές ελέγχονται οι μεταβλητές που θα κινήσουν το Quadcopter κατά roll, pitch και yaw. Η λογική είναι παρόμοια οπότε θα εξηγηθεί μόνο το roll. Αναφερθήκαμε παραπάνω ότι τα roll, pitch, yaw σε θέση neutral έχουν τιμή 1500 μsec και κινώντας το stick τέρμα δεξιά παίρνει την τιμή 2000 μsec ενώ τέρμα αριστερά την τιμή 1000 μsec. Σε κάθε περίπτωση η τρέχουσα τιμή αφαιρείται από την κεντρική και διαιρείται δια 5. Έτσι στις ακραίες θέσεις του stick οι μεταβλητές roll_right και roll_left θα έχουν μέγιστη τιμή 100

```
39      if (start == 2){
40
41      if (throttle >1900) throttle = 1900;
42      if (roll_input>1500)roll_right = (roll_input - 1500)/float(5.0);
43      if (roll_input<1500)roll_left = (1500 - roll_input)/float(5.0);
44      if (pitch_input>1500)pitch_down = (pitch_input - 1500)/float(5.0);
45      if (pitch_input<1500)pitch_up = (1500 - pitch_input)/float(5.0);
46      if (yaw_input>1500)yaw_right = (yaw_input - 1500)/float(5.0);
47      if (yaw_input<1500)yaw_left  = (1500 - yaw_input)/float(5.0);
```

Εικόνα 86. Επεξήγηση κώδικα πρώτης έκδοσης Quadcopter μέρος δ΄

μsec. Αυτή η τιμή θα προστεθεί στην τρέχουσα τιμή των μοτέρ, στα αντίστοιχα μοτέρ ώστε τελικά να κινήσουν το Quadcopter σε roll δεξιά ή roll αριστερά.

Στις γραμμές 55 – 58 μορφοποιούνται οι τιμές των μεταβλητών motor1,motor2,motor3 και motor4 ώστε να κινήσουν το Quadcopter προς όλες τις δυνατές κατευθύνσεις. Έτσι για παράδειγμα, αν δώσουμε roll right από το χειριστήριο θα μεταβάλλουμε την τιμή της μεταβλητής roll_right ή οποία θα προστεθεί στην τρέχουσα τιμή των μοτέρ 3 και 4 ώστε να κινήσει το Quadcopter κατά roll δεξιά. Κατά αντιστοιχία θα καταφέρουμε μεταβάλλοντας τις αντίστοιχες μεταβλητές να κινήσουμε το Drone στην επιθυμητή διεύθυνση.

```
55      motor1 = throttle + roll_left + pitch_up + yaw_right ;          //          CW    (4)        (1) CCW
56      motor2 = throttle + roll_left + pitch_down + yaw_left ;         //                      /   /
57      motor3 = throttle + roll_right + pitch_down + yaw_right ;       //                      /   /
58      motor4 = throttle + roll_right + pitch_up + yaw_left ;          //          CCW   (3)        (2) CW
```

Εικόνα 87. Επεξήγηση κώδικα πρώτης έκδοσης Quadcopter μέρος ε΄

Στις γραμμές 62 – 71 θέτουμε όρια για την απόδοση των μοτέρ . Έτσι δεν θέλουμε η τιμή τους να πέσει κάτω από 1100 μsec (είναι start = 2) και να μην ξεπεράσει την τιμή των 2000 μsec, τιμή για την οποία έχουν βαθμονομήσει τους ESC ως μέγιστη.

.

```
62      if (motor1 < 1100) motor1 = 1100;
63      if (motor2 < 1100) motor2 = 1100;
64      if (motor3 < 1100) motor3 = 1100;
65      if (motor4 < 1100) motor4 = 1100;
66
67
68      if(motor1 > 2000)motor1 = 2000;
69      if(motor2 > 2000)motor2 = 2000;
70      if(motor3 > 2000)motor3 = 2000;
71      if(motor4 > 2000)motor4 = 2000;
```

Εικόνα 88. Επεξήγηση κώδικα πρώτης έκδοσης Quadcopter μέρος ζ΄

Για κάθε άλλη περίπτωση που δεν είναι start = 2 θέλουμε στα μοτέρ να πηγαίνει η τιμή 1000 μsec, τιμή που έχει οριστεί ως ελάχιστη κατά την διαδικασία βαθμονόμησης των ESC’s.

```
74      else {
75      motor1 = 1000;
76      motor2 = 1000;
77      motor3 = 1000;
78      motor4 = 1000;
79      }
```

Εικόνα 89. Επεξήγηση κώδικα πρώτης έκδοσης Quadcopter μέρος η΄

Οι επόμενες γραμμές έχουν εξηγηθεί παραπάνω κατά τον σχηματισμό της κυματομορφής οδήγησης του ESC και απλώς επαναλαμβάνονται για τα 4 μοτέρ.

```
83      while(micros() - zero_timer < 4000);                            //250 Hz    program loop
84      zero_timer = micros();
85
86      PORTD |= B11110000;                                             // όλες οι κυματομορφές( esc - motors) HIGH
87
88      esc1waveform = micros() + motor1;                               // μέτρα χρόνο που απαιτείται + έυρος παλμού για κάθε esc - motor
89      esc2waveform = micros() + motor2;
90      esc3waveform = micros() + motor3;
91      esc4waveform = micros() + motor4;
92
93
94
95
96      while ( (PORTD & B11110000 ) != 0) {                            // μείνε στο loop μέχρι και οι 4 έξοδοι γίνουν LOW
97      esc_loop_timer = micros();
98      if(esc1waveform <= esc_loop_timer)PORTD &= B11101111;           //μέτρα χρόνο, μόλις γίνει ίσος με τον παραπάνω που υπολογίστηκε γύρνα την έξοδο σε LOW
99      if(esc2waveform <= esc_loop_timer)PORTD &= B11011111;
100     if(esc3waveform <= esc_loop_timer)PORTD &= B10111111;
101     if(esc4waveform <= esc_loop_timer)PORTD &= B01111111;
```

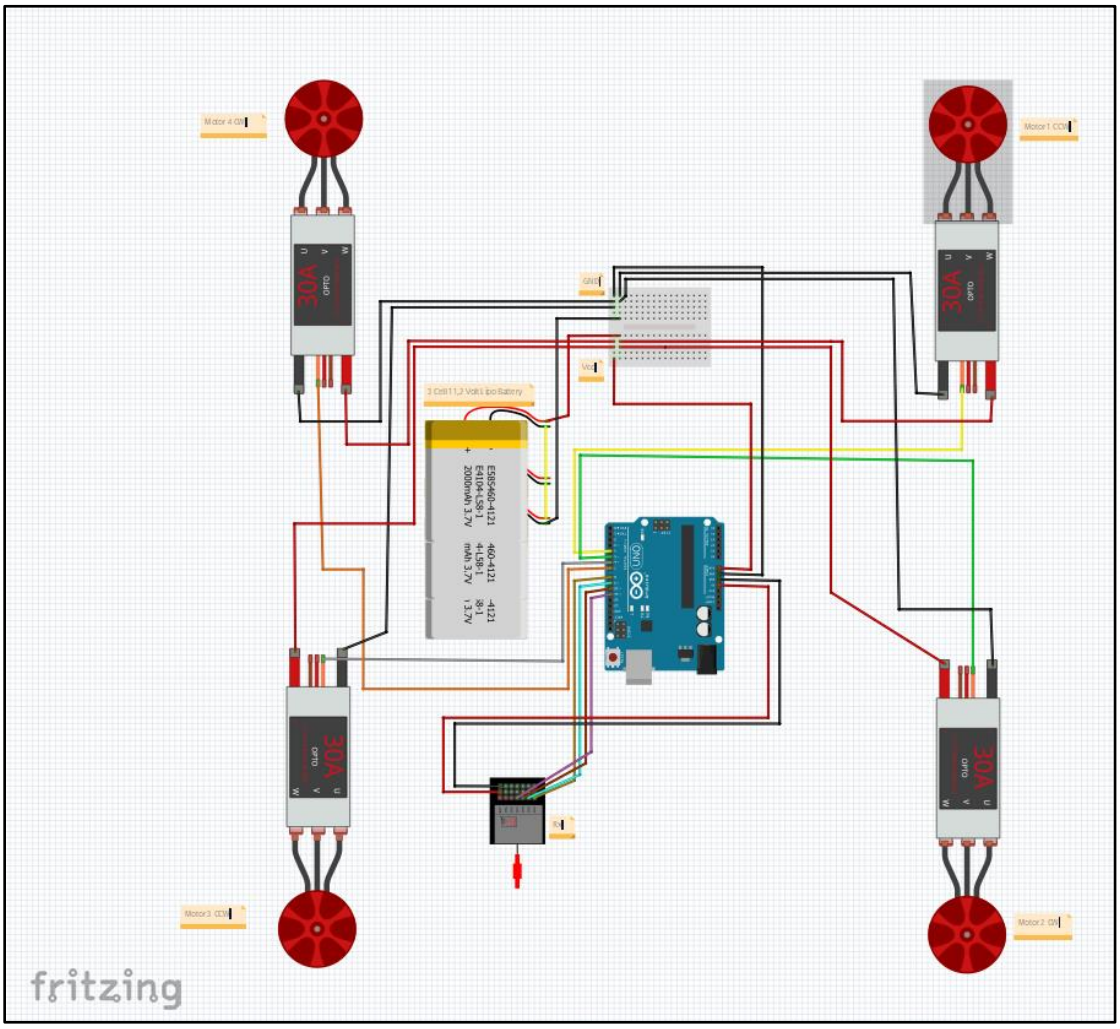
```
124 ISR(PCINT0_vect){
125   current_time = micros();
126
127   if(PINB & B00000001){
128     if(last_channel_1 == 0){
129       last_channel_1 = 1;
130       timer_1 = current_time;
131     }
132   }
133   else if(last_channel_1 == 1){
134     last_channel_1 = 0;
135     roll_input = current_time - timer_1;
136   }
137
138
139   if(PINB & B00000010 ){
140     if(last_channel_2 == 0){
141       last_channel_2 = 1;
142       timer_2 = current_time;
143     }
144   }
145   else if(last_channel_2 == 1){
146     last_channel_2 = 0;
147     pitch_input = current_time - timer_2;
148   }
```

```
151   if(PINB & B00000100 ){
152     if(last_channel_3 == 0){
153       last_channel_3 = 1;
154       timer_3 = current_time;
155     }
156   }
157   else if(last_channel_3 == 1){
158     last_channel_3 = 0;
159     throttle_input = current_time - timer_3;
160
161   }
162
163
164   if(PINB & B00001000 ){
165     if(last_channel_4 == 0){
166       last_channel_4 = 1;
167       timer_4 = current_time;
168     }
169   }
170   else if(last_channel_4 == 1){
171     last_channel_4 = 0;
172     yaw_input = current_time - timer_4;
173   }
```

Εικόνα 90. Επεξήγηση κώδικα πρώτης έκδοσης Quadcopter μέρος θ΄

3.7 Σχεδιασμός και υλοποίηση του κυκλώματος του Quadcopter

Στην εικόνα 91 διακρίνεται το κύκλωμα και οι συνδέσεις της πρώτης έκδοσης του Quadcopter. Έχει δημιουργηθεί με το πρόγραμμα fritzing και διακρίνονται τα 4 μοτέρ και ESC's, ο μικροελεγκτής Arduino Nano, ο δέκτης του τηλεχειρισμού και η μπαταρία τριών cell ή 11,2V τεχνολογίας Lipo. Οι συνδέσεις περιγράφονται στον πίνακα που ακολουθεί .

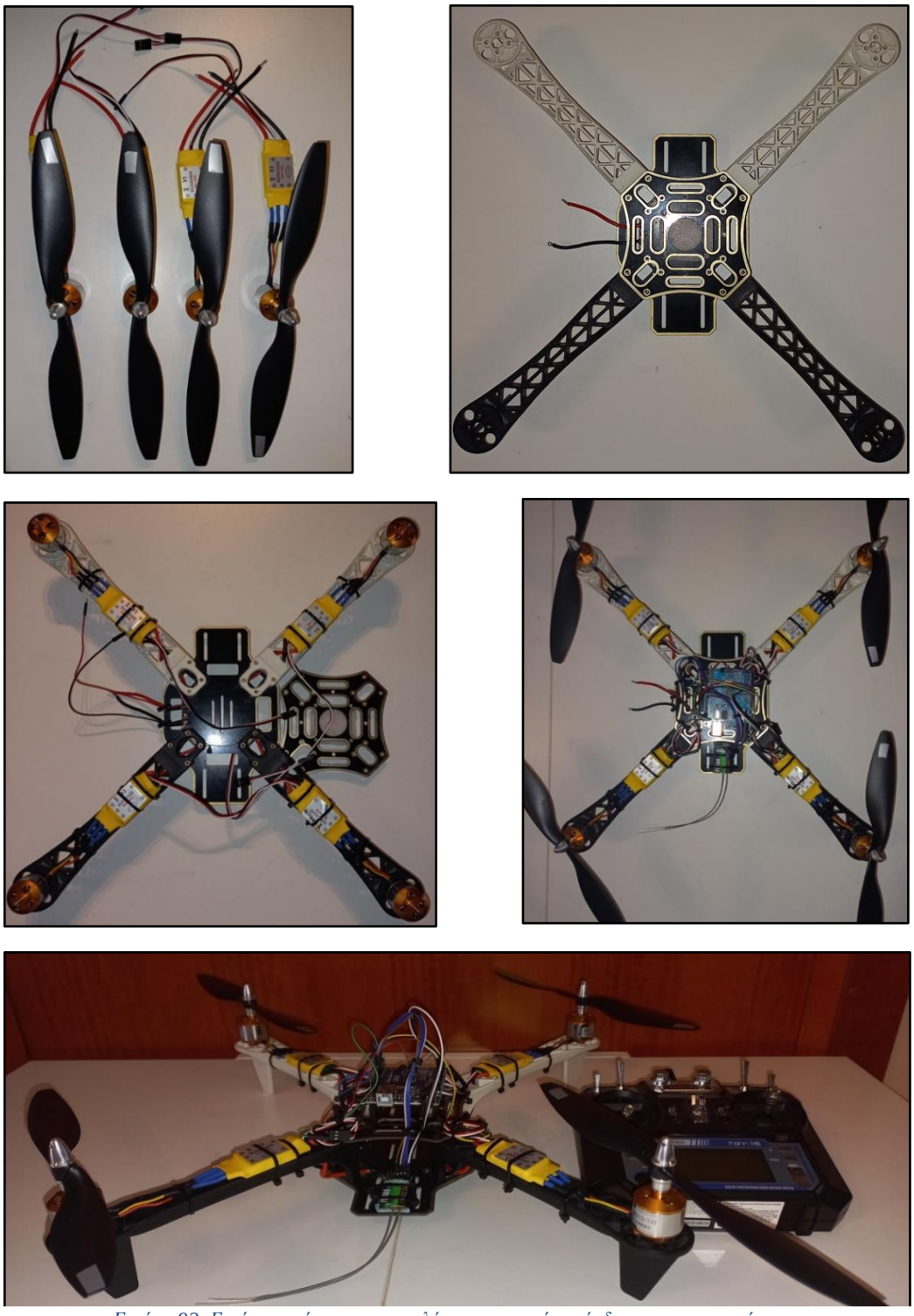


Εικόνα 91. Η πρώτη έκδοση του τετρακόπτερου στο fritzing

Wiring	Description	Arduino Uno Digital Pins	Motor Rotation
ESC's	ESC Motor 1	4	CCW
	ESC Motor 2	5	CW
	ESC Motor 3	6	CCW
	ESC Motor 4	7	CW
Receiver	Roll	8	
	Pitch	9	
	Throttle	10	
	Yaw	11	

Πίνακας 16. Οι συνδέσεις των ESC's και του δέκτη του τηλεχειρισμού με το Arduino Uno

3.8 Εικόνες από την υλοποίηση της πρώτης έκδοσης του Quadcopter



Εικόνα 92. Εικόνες από την συναρμολόγηση της πρώτης έκδοσης του τετρακόπτερου

3.9 Συμπεράσματα

Στην εικόνα 92 παρουσιάζεται η πρώτη λειτουργική έκδοση του Quadcopter. Εφαρμόζοντας την τάση τροφοδοσίας κάνει boot το πρόγραμμα του μικροελεγκτή και ενεργοποιούνται τα τέσσερα μοτέρ και ESC's. Κάνοντας arm τα μοτέρ αυτά ενεργοποιούνται και λειτουργούν σε κατάσταση idle σε τιμή throttle 1100 μsec. Αυξομειώνοντας με προσοχή τα ρυθμιστικά του control panel του τηλεχειρισμού παρατηρούμε την αλλαγή της συμπεριφοράς του Drone. Κάνοντας Disarm τα μοτέρ αυτά σταματούν να περιστρέφονται με το πρόγραμμα να στέλνει στα ESC's την τιμή των 1000 μsec, την ελάχιστη τιμή για την οποία βαθμονομήθηκαν τα ESC's.

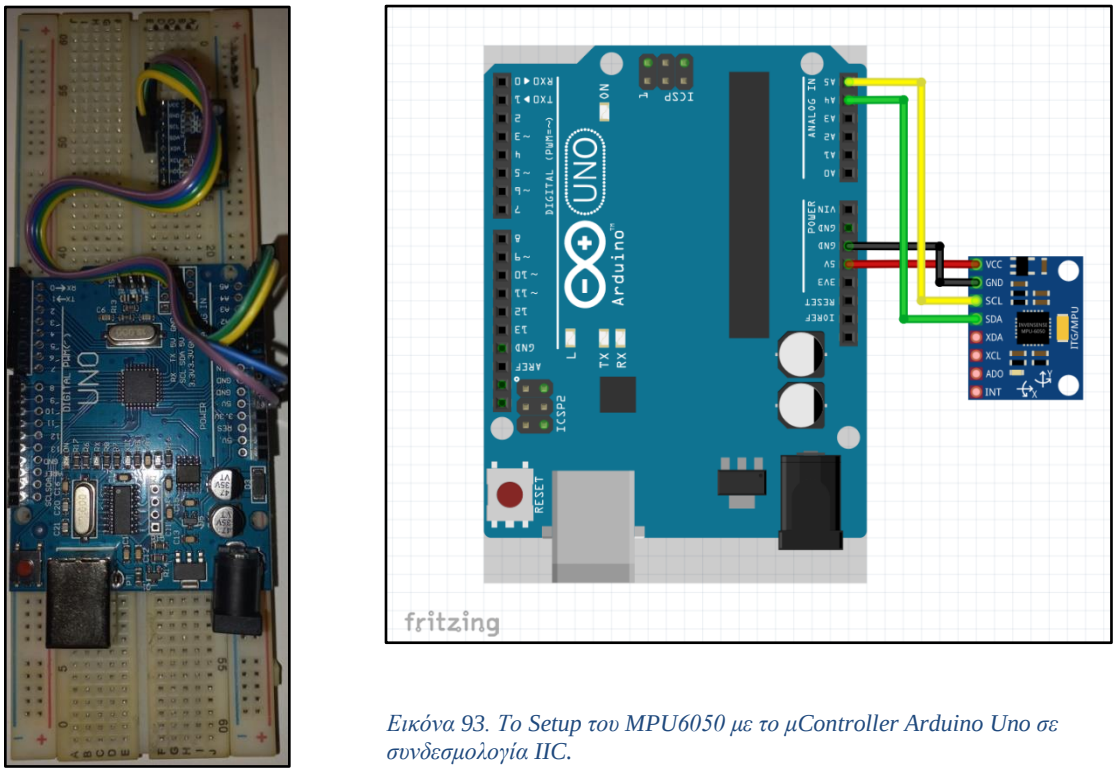
Τα τετρακόπτερα είναι σκάφη τα οποία πτητικά είναι αδύνατο να πετάξουν δίχως την ύπαρξη κάποιου controller ο οποίος πολλές φορές το δευτερόλεπτο θα διορθώνει την απόδοση των μοτέρ ώστε να καθιστά το σκάφος αξιόπλοο. Λόγω του ότι δεν έχουν κανένα αεροδυναμικό σχεδιασμό ή όπως λέγεται δεν ΄΄πλανάρουν΄΄ είναι σχεδόν αδύνατο να πετάξουν ΄΄ manual΄΄. Έτσι και στην περίπτωση της πρώτης έκδοσης του Quadcopter , για την περίπτωση που θα αυξήσουμε το throttle, πολύ γρήγορα το σκάφος θα ανατραπεί και θα καταστραφεί.

Για αυτόν το λόγο πρέπει να εισαχθεί στον σχεδιασμό μια βαθμίδα η οποία θα γνωρίζει κάθε στιγμή την κατάσταση του Drone σχετικά με τις γωνίες roll, pitch και yaw και με υπολογισμούς θα ΄΄ διορθώνει ΄΄ την απόδοση των μοτέρ ώστε να παραμένει το Drone σε κατάσταση level ή αλλιώς σε ισορροπία. Αυτή η βαθμίδα ονομάζεται IMU ή Inertial Measurement Unit και στην περίπτωση του Quadcopter της κατασκευής είναι το MPU-6050 ένας αισθητήρας 6 Degrees of Freedom .

Στις επόμενες παραγράφους θα περιγραφεί με λεπτομέρεια η λειτουργία του αισθητήρα και ο τρόπος με τον οποίο θα εισαχθεί στην επόμενη έκδοση του Quadcopter και προγραμματιστικά και σχεδιαστικά .

3.10 Inertial Measurement Unit (IMU)

Η βαθμίδα που θα χρησιμοποιηθεί ώστε να γίνεται γνωστή η κατάσταση του Drone στον χώρο είναι το MPU-6050, 6 Degrees of Freedom Gyro Accelerometer. Αρχικά θα παρουσιαστεί το Hardware ή setup του MPU και στην συνέχεια θα εξηγηθεί ο κώδικας με την βοήθεια του οποίου θα καταφέρουμε να ΄΄διαβάζουμε΄΄ τις γωνίες roll και pitch. Θα παρουσιαστούν δύο εκδοχές οδήγησης του αισθητήρα, μία διαβάζοντας raw data και μορφοποιώντας τα και η δεύτερη χρησιμοποιώντας το Digital Motion Processing του MPU με την μορφοποιημένη βιβλιοθήκη του Jeff Rowberg.



Εικόνα 93. Το Setup του MPU6050 με το μController Arduino Uno σε συνδεσμολογία IIC.

3.10.1 Το πρόγραμμα οδήγησης του MPU6050 (Raw Data)

Στην παράγραφο αυτή θα εξηγηθεί το πρόγραμμα οδήγησης του MPU6050 , συχνά – πυκνά θα γίνεται αναφορά στα manual του αισθητήρα ώστε να γίνεται αντιληπτό το πρόγραμμα.

```
#include <Wire.h>

int gyro_roll, gyro_pitch, gyro_yaw;
long acc_roll, acc_pitch, acc_yaw, acc_total_vector;
int temperature;
long zero_timer;
float pitch, roll;
float roll_acc, pitch_acc;
```

Εικόνα 94. MPU-6050 Raw Data Code μέρος α΄

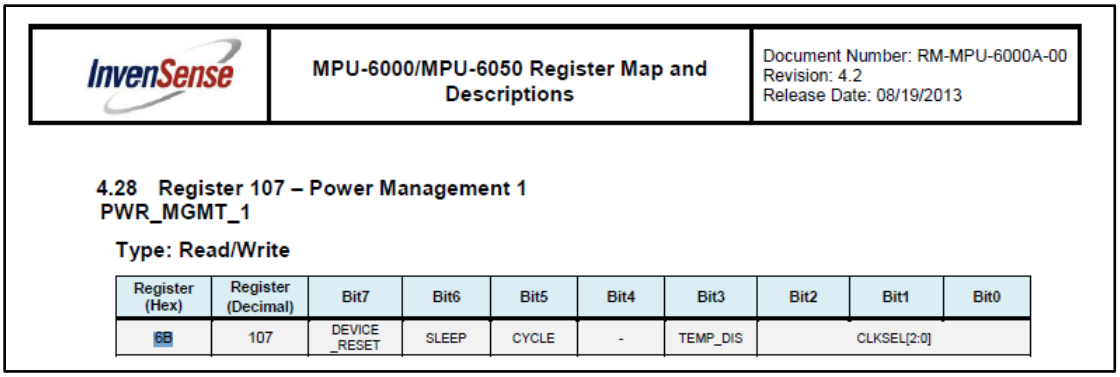
Έτσι λοιπόν στην εικόνα 94 γίνονται αρχικά οι δηλώσεις των μεταβλητών που θα χρησιμοποιηθούν κατά την ανάπτυξη του προγράμματος. Η βιβλιοθήκη Wire εξυπηρετεί την επικοινωνία του αισθητήρα με τον μικροελεγκτή σύμφωνα με το πρωτόκολλο IIC, η συχνότητα λειτουργίας είναι στα 400 Hz.

Στο void setup ο κώδικας που θα εκτελεστεί μια φορά κατά την εφαρμογή τάσης λειτουργίας στον μικροελεγκτή.

```
void setup() {
  Wire.begin();                                          //Start I2C as master
  Serial.begin(38400);                                   //Use only for debugging

  Wire.beginTransmission(0x68);                          //Start communicating with the MPU-6050
  Wire.write(0x6B);                                      //Send the requested starting register
  Wire.write(0x00);                                      //Set the requested starting register
  Wire.endTransmission();                                //End the transmission
  //accelerometer (+/-8g)
  Wire.beginTransmission(0x68);                          //Start communicating with the MPU-6050
  Wire.write(0x1C);                                      //Send the requested starting register
  Wire.write(0x10);                                      //Set the requested starting register
  Wire.endTransmission();                                //End the transmission
  // gyro (500dps)
  Wire.beginTransmission(0x68);                          //Start communicating with the MPU-6050
  Wire.write(0x1B);                                      //Send the requested starting register
  Wire.write(0x08);                                      //Set the requested starting register
  Wire.endTransmission();


  zero_timer = micros();                                 //Reset the loop timer
}
```

Εικόνα 95. MPU-6050 Raw Data Code μέρος β΄

Ξεκινά η επικοινωνία με το IIC και δηλώνεται η ταχύτητα επικοινωνίας με το Arduino μέσω του Serial Monitor στα 38400 Kbps. Η Wire ξεκινά επικοινωνία με το MPU στην default διεύθυνση του MPU-6050, 0x68, ενώ από στην διεύθυνση 6B του καταχωρητή 107 δίνει τιμή 0 και ενεργοποιεί τον εσωτερικό ταλαντωτή συχνότητας 8 MHz. Στις επόμενες γραμμές θα ορίσει την ευαισθησία του accelerometer και του gyro .



InvenSense | MPU-6000/MPU-6050 Register Map and Descriptions | Document Number: RM-MPU-6000A-00 Revision: 4.2 Release Date: 08/19/2013

4.28 Register 107 – Power Management 1
PWR_MGMT_1

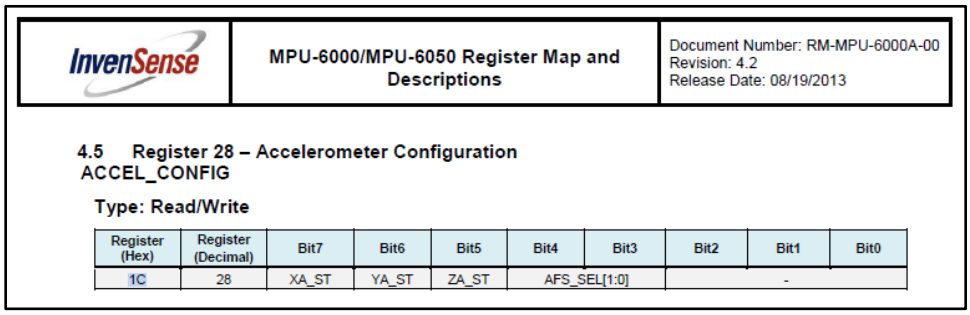
Type: Read/Write

Register (Hex)	Register (Decimal)	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
6B	107	DEVICE _RESET	SLEEP	CYCLE	-	TEMP_DIS	CLKSEL[2:0]		

CLKSEL	Clock Source
0	Internal 8MHz oscillator
1	PLL with X axis gyroscope reference
2	PLL with Y axis gyroscope reference
3	PLL with Z axis gyroscope reference
4	PLL with external 32.768kHz reference
5	PLL with external 19.2MHz reference
6	Reserved
7	Stops the clock and keeps the timing generator in reset

Εικόνα 96. MPU-6050 Raw Data Code μέρος γ΄

Στον register (Hex) 1C θα δώσει την τιμή 10 που αντιστοιχεί +- 8g, ενώ στον register (Hex) 1B θα δώσει την τιμή 08 που αντιστοιχεί se +- 500 d/sec full scale . Όλα αυτά φαίνονται στις εικόνες που ακολουθούν. Στην τελευταία γραμμή γίνεται reset η μεταβλητή zero_timer που θα λειτουργήσει ως χρονόμετρο.



InvenSense | MPU-6000/MPU-6050 Register Map and Descriptions | Document Number: RM-MPU-6000A-00 Revision: 4.2 Release Date: 08/19/2013

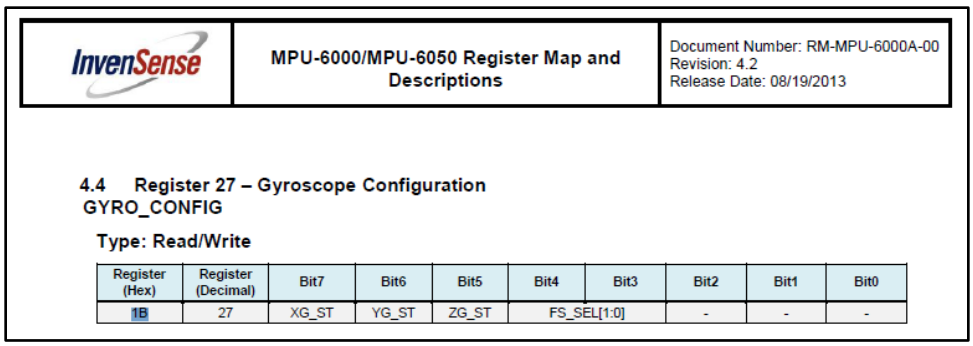
4.5 Register 28 – Accelerometer Configuration
ACCEL_CONFIG

Type: Read/Write

Register (Hex)	Register (Decimal)	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
1C	28	XA_ST	YA_ST	ZA_ST	AFS_SEL[1:0]		-		

AFS_SEL	Full Scale Range
0	± 2*g*
1	± 4*g*
2	± 8*g*
3	± 16*g*

Εικόνα 97. MPU-6050 Raw Data Code μέρος δ΄



InvenSense | MPU-6000/MPU-6050 Register Map and Descriptions | Document Number: RM-MPU-6000A-00 Revision: 4.2 Release Date: 08/19/2013

4.4 Register 27 – Gyroscope Configuration
GYRO_CONFIG

Type: Read/Write

Register (Hex)	Register (Decimal)	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
1B	27	XG_ST	YG_ST	ZG_ST	FS_SEL[1:0]		-	-	-

FS_SEL	Full Scale Range
0	± 250 °/s
1	± 500 °/s
2	± 1000 °/s
3	± 2000 °/s

Εικόνα 98. MPU-6050 Raw Data Code μέρος ε΄

Ακολουθεί η void loop η οποία θα εκτελείται διαρκώς με μια συχνότητα 100 Hz όπως θα εξηγηθεί στις επόμενες γραμμές. Αρχίζοντας από τον καταχωρητή 3B θα ζητήσουμε 14 bytes τα οποία θα καταχωρηθούν στις 7 μεταβλητές με 2 byte ανά μεταβλητή Low και High. Η μεταβλητή temperature αφορά το compensate της θερμοκρασίας του chip στις τιμές εξόδου των καταχωρητών αλλά δεν θα μας απασχολήσει.

```
34 void loop(){
35
36   Wire.beginTransmission(0x68);                    //Start communicating with the MPU-6050
37   Wire.write(0x3B);                                //Send the requested starting register
38   Wire.endTransmission();                          //End the transmission
39   Wire.requestFrom(0x68,14);                       //Request 14 bytes from the MPU-6050
40   while(Wire.available() < 14);                    //Wait until all the bytes are received
41   acc_roll = Wire.read()<<8|Wire.read();           //Add the low and high byte to the acc_roll variable
42   acc_pitch = Wire.read()<<8|Wire.read();          //Add the low and high byte to the acc_pitchl variable
43   acc_yaw = Wire.read()<<8|Wire.read();            //Add the low and high byte to the acc_yaw variable
44   temperature = Wire.read()<<8|Wire.read();        //Add the low and high byte to the temperature variable
45   gyro_roll = Wire.read()<<8|Wire.read();          //Add the low and high byte to the gyro_roll variable
46   gyro_pitch = Wire.read()<<8|Wire.read();         //Add the low and high byte to the gyro_pitch variable
47   gyro_yaw = Wire.read()<<8|Wire.read();
```

Εικόνα 99. MPU-6050 Raw Data Code μέρος ζ΄

Στις παρακάτω γραμμές στην εικόνα 100 υπολογίζονται οι γωνίες pitch και roll του accelerometer αφού μετατραπούν από rad σε degrees (180/π = 57,296), διότι η συνάρτηση ASIN επιστρέφει rad.

```
51   acc_total_vector = sqrt((acc_roll*acc_roll)+(acc_pitch*acc_pitch)+(acc_yaw*acc_yaw));  //Calculate the total accelerometer vector.
52   pitch_acc = asin((float)acc_pitch/acc_total_vector)* 57.296;      //Calculate the pitch angle.
53   roll_acc = asin((float)acc_roll/acc_total_vector)* -57.296;       //Calculate the roll angle.
```

Εικόνα 100. MPU-6050 Raw Data Code μέρος η΄

Από το manual του MPU6050 για sensitivity 500 degrees / sec, όπως επιλέξαμε στο πρόγραμμα, ή έξοδος του καταχωρητή πρέπει να διαιρεθεί με 65.5 ώστε να πάρουμε την τρέχουσα τιμή σε degrees / sec. Με δεδομένο όμως ότι η συχνότητα loop του προγράμματος είναι 100 Hz ή 10000 μsec για να εξάγουμε το αποτέλεσμα του gyro σε degrees πολλαπλασιάζουμε με 10000μsec. Έτσι (1/65,5) * 10000 10^{-6} = 0,0001526 sec.

```
56  pitch += gyro_pitch * 0.0001526;                //0.0001526 = ((1 / 65.5) / 100Hz)
57  roll += gyro_roll * 0.0001526;
```

FS_SEL	Full Scale Range	LSB Sensitivity
0	± 250 °/s	131 LSB/°/s
1	± 500 °/s	65.5 LSB/°/s
2	± 1000 °/s	32.8 LSB/°/s
3	± 2000 °/s	16.4 LSB/°/s

Εικόνα 101. MPU-6050 Raw Data Code μέρος θ΄

Από την θεωρία γνωρίζουμε ότι το Gyro είναι ευαίσθητο στην ολίσθηση ενώ το accelerometer είναι ευαίσθητο στους κραδασμούς. Πρέπει να βρεθεί τρόπος να συνδυάσουμε τα δύο χαρακτηριστικά του MPU6050 ώστε να καταλήξουμε σε ασφαλή συμπεράσματα. Αυτό συμβαίνει με τις επόμενες 2 γραμμές κώδικα όπως φαίνεται στην εικόνα που ακολουθεί.

```
60  pitch = pitch * 0.98 + pitch_acc * 0.02;        //Correct the drift of the gyro pitch angle with the accelerometer pitch angle
61  roll = roll * 0.98 + roll_acc * 0.02;           //Correct the drift of the gyro roll angle with the accelerometer roll angle
```

Εικόνα 102. MPU-6050 Raw Data Code μέρος ί

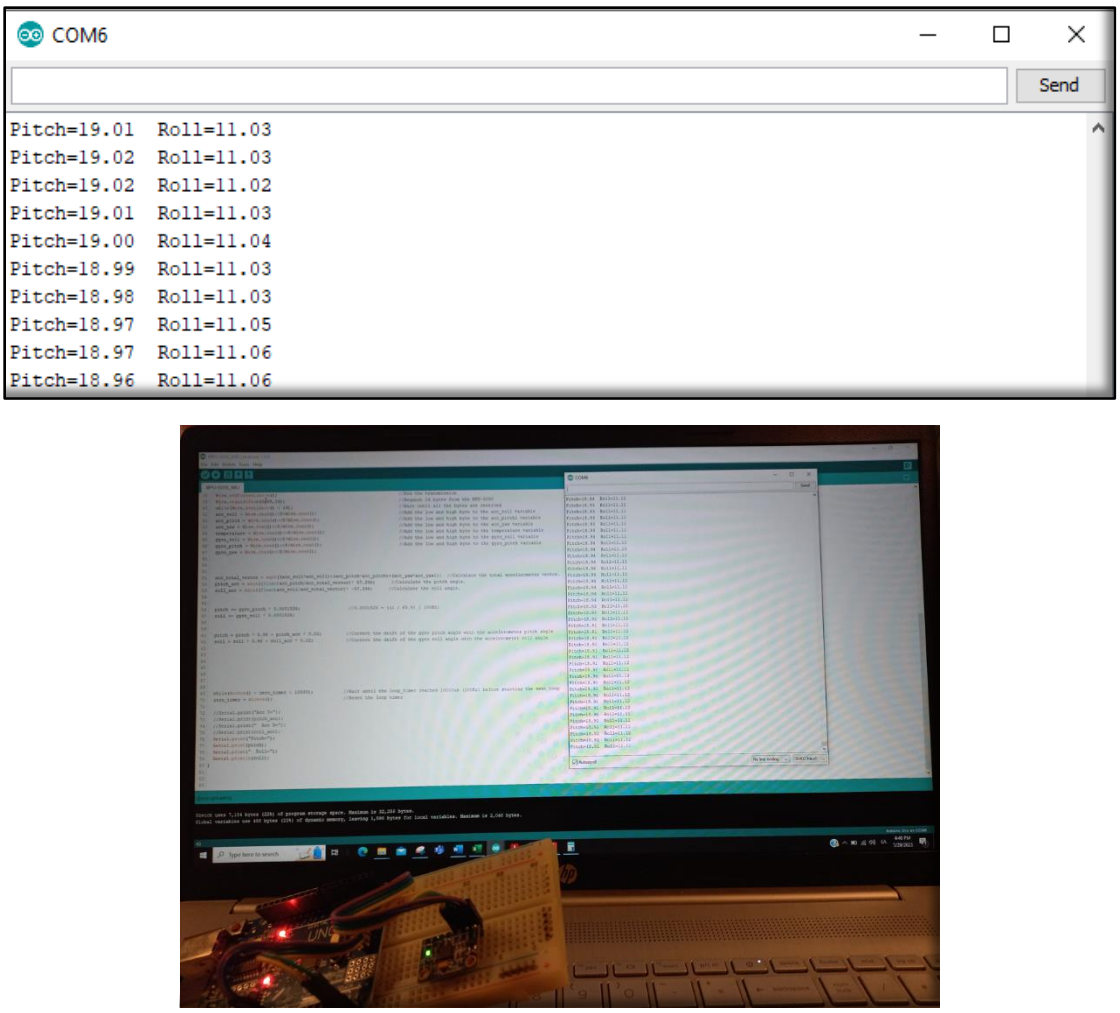
Ουσιαστικά παίρνουμε 98% από την τιμή του Gyro και 2% από την τιμή του accelerometer. Έτσι στις 100 φορές που θα εκτελεστεί ο κώδικας 2 φορές το accelerometer θα διορθώσει το drift του Gyro. Στην συνέχεια οι επόμενες 2 γραμμές είναι ο ρυθμός εκτέλεσης του προγράμματος ο οποίος είναι 100 Hz. Τέλος ακολουθούν κάποιες Serial.print για να δούμε τα αποτελέσματα στο Serial Monitor.

```
69  while(micros() - zero_timer < 10000);           //Wait until the loop_timer reaches 10000us (100Hz) before starting the next loop
70  zero_timer = micros();                          //Reset the loop timer
```

Εικόνα 103. MPU-6050 Raw Data Code μέρος κ΄

```
76  Serial.print("Pitch=");
77  Serial.print(pitch);
78  Serial.print("  Roll=");
79  Serial.println(roll);
```

Εικόνα 104. MPU-6050 Raw Data Code μέρος λ΄



Εικόνα 105. MPU-6050 Raw Data Code γωνίες κλίσης

3.10.2 Το πρόγραμμα οδήγησης του MPU-6050 (DMP)

Στην παράγραφο αυτή θα εξηγηθεί η δεύτερη εκδοχή του προγράμματος οδήγησης του MPU-6050 με την χρήση Digital Motion Processing και την βιβλιοθήκη MPU-6050_6Axis_MotionApps612 του Jeff Rowberg. Κατά αυτόν το τρόπο στις πρώτες γραμμές ορίζονται οι βιβλιοθήκες που θα χρησιμοποιηθούν MPU-6050_6Axis_MotionApps612, Wire, I2Cdev ενώ στην συνέχεια ορίζεται το object mpu του MPU-6050. Ακολουθούν οι δηλώσεις των αναγκαίων μεταβλητών και ορίζονται το Quaternion **q** w,x,y,z και το διάνυσμα της βαρύτητας **gravity** x,y,z. Τέλος ορίζονται τα yaw,pitch,roll .

```
IMU_DMP §
#include "MPU6050_6Axis_MotionApps612.h"
#include "Wire.h"
#include "I2Cdev.h"
MPU6050 mpu;


// MPU control/status vars
bool dmpReady = false;  // set true if DMP init was successful
uint8_t devStatus;      // return status after each device operation
uint16_t packetSize;    // expected DMP packet size (default is 42 bytes)
uint16_t fifoCount;     // count of all bytes currently in FIFO

uint8_t fifoBuffer[64]; // FIFO storage buffer

Quaternion q;           // [w, x, y, z]         quaternion container
VectorFloat gravity;    // [x, y, z]            gravity vector

float ypr[3], yaw, pitch, roll;          // [yaw, pitch, roll]   yaw/pitch/roll container and gravity vector
unsigned long  zero_timer ;
```

Εικόνα 106. MPU-6050 DMP Code μέρος α΄

```
void setup() {
          Serial.begin(38400);
          Wire.begin();
          Wire.setClock(400000);

          mpu.initialize();
          devStatus = mpu.dmpInitialize();

          mpu.setXGyroOffset(51);
          mpu.setYGyroOffset(8);
          mpu.setZGyroOffset(21);
          mpu.setXAccelOffset(1150);
          mpu.setYAccelOffset(-50);
          mpu.setZAccelOffset(1060);

          mpu.CalibrateAccel(6);
          mpu.CalibrateGyro(6);
          mpu.PrintActiveOffsets();

          mpu.setDMPEnabled(true);

          dmpReady = true;

          packetSize = mpu.dmpGetFIFOPacketSize();

          zero_timer = micros();
             }
```

Εικόνα 107. MPU-6050 DMP Code μέρος β΄

Στην συνέχεια στο void setup ξεκινά η σειριακή επικοινωνία στα 38400 Kbps, ξεκινά η Wire που είναι υπεύθυνη για την επικοινωνία με το IMU και τίθεται το ρολόι του I2C στα 400 KHz. Στην συνέχεια γίνεται initialize το MPU-6050 και ακολουθούν κάποιες τιμές οι οποίες έχουν προκύψει από το calibration του MPU (IMU_Zero), είναι ο μέσος όρος 1000 μετρήσεων, αφορά το drift του gyro και το offset του accelerometer και αφαιρούνται διαρκώς από τις τρέχουσες τιμές . Τέλος παίρνει το πρώτο πακέτο FIFO να το έχει έτοιμο για το πρώτο loop του προγράμματος και ορίζεται ένα ρολόι στην μεταβλητή zero_timer.

```
void loop() {

        if (mpu.dmpGetCurrentFIFOPacket(fifoBuffer)) {

            mpu.dmpGetQuaternion(&q, fifoBuffer);
            mpu.dmpGetGravity(&gravity, &q);
            mpu.dmpGetYawPitchRoll(ypr, &q, &gravity);

            yaw = ypr[0] * (180 / M_PI);
            pitch = ypr[1] * (180 / M_PI) ;
            roll = ypr[2] * (180 / M_PI);
            }

            //Serial.print("  pitch=");
            Serial.print(pitch);
            Serial.print("     ");
            Serial.println(roll);

            while(micros() - zero_timer < 10000);
            zero_timer = micros();

}
```

Εικόνα 110. MPU-6050 DMP Code μέρος γ΄

Στην void loop την ρουτίνα που εκτελείται διαρκώς, αν το fifoBuffer είναι έτοιμο πάρε το τρέχον πακέτο και χρησιμοποίησέ το για να διαμορφώσεις το Quaternion **q**. Στον πίνακα 17 φαίνεται η δομή του FIFO πακέτου. Έτσι στην ρουτίνα mpu.dmpGetQuaternion όπως αυτή φαίνεται μέσα από την βιβλιοθήκη στην εικόνα 99 και εικόνα 101, δώσε στην QUAT W τα byte 0,1, στο X τα 4,5 , στο Y τα 8,9 και στο QUAT Z τα 12,13 στις αντίστοιχες μεταβλητές. Τώρα πια έχει διαμορφωθεί το Quaternion **q**.

```
|======================================================================= *
| Default MotionApps v6.12 28-byte FIFO packet structure:               |
|                                                                       |
| [QUAT W][      ][QUAT X][      ][QUAT Y][      ][QUAT Z][      ]     |
|   0   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15      |
|                                                                       |
| [GYRO X][GYRO Y][GYRO Z][ACC X ][ACC Y ][ACC Z ]                      |
|  16  17  18  19  20  21  22  23  24  25  26  27                       |
* ======================================================================= |
```

Εικόνα 109. FIFO packet

Μεταβλητή	Quaternion	Byte	Description
data[0]	w	0,1	Real
data[1]	x	4,5	X axis
data[2]	y	8,9	Y axis
data[3]	z	12,13	Z axis

Πίνακας 17. Η δομή του πακέτου FIFO

```

uint8_t MPU6050::dmpGetQuaternion(int16_t *data, const uint8_t* packet) {
    // TODO: accommodate different arrangements of sent data (ONLY default supported now)
    if (packet == 0) packet = dmpPacketBuffer;
    data[0] = ((packet[0] << 8) | packet[1]);
    data[1] = ((packet[4] << 8) | packet[5]);
    data[2] = ((packet[8] << 8) | packet[9]);
    data[3] = ((packet[12] << 8) | packet[13]);
    return 0;
}
```

Εικόνα 108. MPU-6050 DMP Code μέρος δ΄

Στην επόμενη ρουτίνα στην εικόνα 111 αφού έχει διαμορφωθεί το Quaternion **q** θα ορίσουμε τα vectors του **gravity** από τις σχέσεις:

$$vx = 2 * (qx*qz-qw*qy)$$

$$vy = 2* (qw*qx +qy*qz)$$

$$vz = (qw)^2-(qx)^2-(qy)^2+(qz)^2$$

Quaternion	Symbol
qw	q - > w
qx	q - > x
qy	q - > y
qz	q - > z

```
12 uint8_t MPU6050::dmpGetGravity(VectorFloat *v, Quaternion *q) {
13     v -> x = 2 * (q -> x*q -> z - q -> w*q -> y);
14     v -> y = 2 * (q -> w*q -> x + q -> y*q -> z);
15     v -> z = q -> w*q -> w - q -> x*q -> x - q -> y*q -> y + q -> z*q -> z;
16     return 0;
```

Εικόνα 111. MPU-6050 DMP Code μέρος ε΄

Τέλος από την ρουτίνα mpu.dmpGetYawPitchRoll και αφού έχουμε το Quaternion **q** κάνουμε extract το ήδη διαμορφωμένο **gravity** και προκύπτουν τα **Yaw data[0], Pitch data[1], Roll data[2]** σε radians αφού συνάρτηση ATAN επιστρέφει radians.

```
19 uint8_t MPU6050::dmpGetYawPitchRoll(float *data, Quaternion *q, VectorFloat *gravity) {
20     // yaw: (about Z axis)
21     data[0] = atan2(2*q -> x*q -> y - 2*q -> w*q -> z, 2*q -> w*q -> w + 2*q -> x*q -> x - 1);
22     // pitch: (nose up/down, about Y axis)
23     data[1] = atan2(gravity -> x , sqrt(gravity -> y*gravity -> y + gravity -> z*gravity -> z));
24     // roll: (tilt left/right, about X axis)
25     data[2] = atan2(gravity -> y , gravity -> z);
26     if (gravity -> z < 0) {
27         if(data[1] > 0) {
28             data[1] = PI - data[1];
29         } else {
30             data[1] = -PI - data[1];
31         }
32     }
33     return 0;
34 }
```

Quaternion	Symbol
qw	q - > w
qx	q - > x
qy	q - > y
qz	q - > z
VectorGravity	Symbol
vx	gravity - > x
vy	gravity - > y
vz	gravity - > z

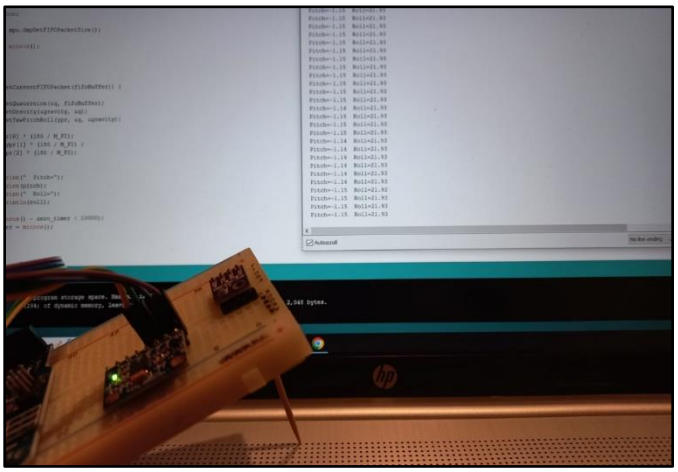
Εικόνα 112. MPU-6050 DMP Code μέρος ζ΄

data [0] = atan2 (2*qx*qy-2*qw*qz, $2*(qw)^2+2*(qx)^2-1$) **Yaw**

data [1] = atan2 (vx, sqrt($(vy)^2+(vz)^2$)) **Pitch**

data [2] = atan2 (vy, vz) **Roll**

Για την περίπτωση που το Yaw είναι ανεστραμμένο, δύο if ρυθμίζουν την γωνία του Pitch. Τέλος, μετατρέπουμε τα radians σε degrees και τα αποθηκεύουμε στις μεταβλητές yaw, pitch, roll και οι απαραίτητες Serial.print για να δούμε τα αποτελέσματα στο Serial Monitor , με την while ρυθμίζουμε την συχνότητα εκτέλεσης του loop στα 100 Hz (10000μsec).



Εικόνα 113. MPU-6050 DMP Code γωνίες κλίσης

3.10.3 Το πρόγραμμα οδήγησης του MPU-9250 (DMP)

Το MPU-9250 είναι η δεύτερη πιο εξελιγμένη έκδοση IMU που χρησιμοποιείται στην κατασκευή . Πρόκειται για έναν αισθητήρα MARG (Magnetic, Angular Rate and Gravity), ή αλλιώς έναν υβριδικό αδρανειακό σύστημα μέτρησης το οποίο ενσωματώνει μία πυξίδα. Ένα αδρανειακό σύστημα μέτρησης μόνο του παρέχει πληροφορίες attitude σε σχέση με την κατεύθυνση της βαρύτητας. Ένα MARG σύστημα ή αλλιώς AHRS (Attitude and Heading Reference System) παρέχει πληροφορίες attitude σε σχέση με την κατεύθυνση της βαρύτητας αλλά και με το μαγνητικό πεδίο της γης . Πιο απλά θα λέγαμε ότι το MPU-9250 εκτός από τις γωνίες pitch και roll , με την γωνία yaw μας δίνει την γωνία κλίσης ως προς τον μαγνητικό βορά.

Στην περίπτωση της κατασκευής , οι πληροφορίες αυτές εξάγονται με την χρήση της βιβλιοθήκης του Hideaki Tai την οποία την βρίσκουμε από το Arduino Reference στην διεύθυνση https://www.arduino.cc/reference/en/libraries/mpu9250/ . Η εξαγωγή αποτελεσμάτων στηρίζεται στα Quaternions ενώ χρησιμοποιείται το φίλτρο Madgwick . Έτσι όπως και στην περίπτωση του MPU-6050 οι γωνίες pitch, roll, yaw προκύπτουν από τα Quaternions και τις γωνίες Euler (σχέσεις 1.13 σελ. 25), και φαίνονται παρακάτω στην εικόνα 105, ένα απόκομμα από την βιβλιοθήκη του MPU9250.

```
void update_rpy(float qw, float qx, float qy, floa
    // Define output variables from updated quater
    // In this coordinate system, the positive z-a
    // Yaw is the angle between Sensor x-axis and
    // Pitch is angle between sensor x-axis and Ea
    // Roll is angle between sensor y-axis and Ear
    // These arise from the definition of the homo
    // Tait-Bryan angles as well as Euler angles a
    // applied in the correct order which for this
    // For more see http://en.wikipedia.org/wiki/G
    float a12, a22, a31, a32, a33;  // rotation ma
    a12 = 2.0f * (qx * qy + qw * qz);
    a22 = qw * qw + qx * qx - qy * qy - qz * qz;
    a31 = 2.0f * (qw * qx + qy * qz);
    a32 = 2.0f * (qx * qz - qw * qy);
    a33 = qw * qw - qx * qx - qy * qy + qz * qz;
    rpy[0] = atan2f(a31, a33);
    rpy[1] = -asinf(a32);
    rpy[2] = atan2f(a12, a22);
    rpy[0] *= 180.0f / PI;
    rpy[1] *= 180.0f / PI;
    rpy[2] *= 180.0f / PI;
    rpy[2] += magnetic_declination;
    if (rpy[2] >= +180.f)
        rpy[2] -= 360.f;
    else if (rpy[2] < -180.f)
        rpy[2] += 360.f;
```

$$\begin{matrix} \varphi \\ \theta = \\ \psi \end{matrix} \begin{matrix} atan2\{\,2(q1q2 + q0q3), [1 - 2(q0^2 + q1^2)]\} \\ sin^{-1}[2(q1q3 - q0q2)] \\ atan2\{2(q0q1 + q2q3), [[1 - 2(q1^2 + q2^2)]\} \end{matrix} \quad (1.13)$$

Εικόνα 114. MPU-9250 , εξαγωγή γωνιών RPY

Αφού υπολογιστούν οι γωνίες, μετατρέπονται από rad σε degrees, προστίθεται το magnetic declination στο yaw (rpy[2] , Heading), ενώ στο τέλος με δύο if διαμορφώνεται η έξοδος του Heading σε ±180 ° .

Η χρήση της βιβλιοθήκης κάνει πολύ απλή την εξαγωγή αποτελεσμάτων και επιτυγχάνεται με την χρήση τριών γραμμών κώδικα. Στην εικόνα 115 φαίνεται το sketch στην πιο απλή του μορφή.

```
#include "MPU9250.h"

MPU9250 mpu;

void setup() {
    Serial.begin(115200);
    Wire.begin();
    delay(2000);
    }

void loop() {
    if (mpu.update()) {
    Serial.print("Yaw, Pitch, Roll: ");
    Serial.print(mpu.getYaw(), 2);
    Serial.print(", ");
    Serial.print(mpu.getPitch(), 2);
    Serial.print(", ");
    Serial.println(mpu.getRoll(), 2);
    }
}
```

```
float getRoll() const { return rpy[0]; }
float getPitch() const { return rpy[1]; }
float getYaw() const { return rpy[2]; }
```

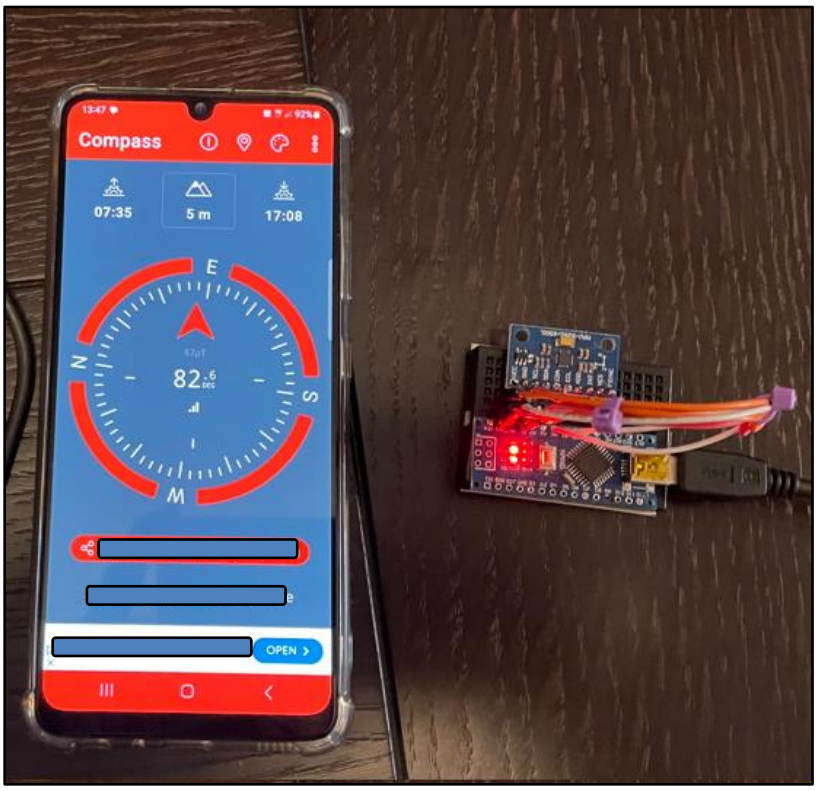
Εικόνα 115. MPU-9250, το sketch στην πιο απλή του μορφή.

Όπως και στην περίπτωση του MPU-6050 για να επιτευχθεί ακρίβεια στις μετρήσεις απαιτείται να τρέξει το sketch ΄΄calibration΄΄ της βιβλιοθήκης και τα αποτελέσματα του να δηλωθούν στο void setup.

```
#include "MPU9250.h"

MPU9250 aux_mpu;

void setup() {
    Serial.begin(115200);
    Wire.begin();
    delay(2000);

//**********************************MPU 9250 Setup ******************************//
    aux_mpu.setup(0x68);                                                        //
    aux_mpu.setAccBias(87.45, 34.34, 31.69);                                    //
    aux_mpu.setGyroBias(15.39, -6.01, -3.05);                                   //
    aux_mpu.setMagBias(216.43, 51.30, -61.15);                                  //
    aux_mpu.setMagScale(0.99, 1.00, 1.00);                                      //
    aux_mpu.setMagneticDeclination(5.1);                                        //
//*******************************************************************************//
  }
void loop() {
    if (aux_mpu.update()) {
    Serial.print("Yaw, Pitch, Roll: ");
    Serial.print(aux_mpu.getYaw(), 2);
    Serial.print(", ");
    Serial.print(aux_mpu.getPitch(), 2);
    Serial.print(", ");
    Serial.println(aux_mpu.getRoll(), 2);
    }
}
```

Εικόνα 116. MPU-9250, το sketch και η εξαγωγή αποτελεσμάτων

Μετά τις δηλώσεις του ΄΄calibration΄΄ ακολουθεί το magnetic declination, ενώ στο void loop η εκτύπωση των αποτελεσμάτων στο serial monitor.




```
Yaw, Pitch, Roll: 81.45, 4.59, 4.05
Yaw, Pitch, Roll: 81.46, 4.59, 4.05
Yaw, Pitch, Roll: 81.51, 4.61, 4.06
Yaw, Pitch, Roll: 81.56, 4.63, 4.07
Yaw, Pitch, Roll: 81.59, 4.64, 4.08
Yaw, Pitch, Roll: 81.61, 4.64, 4.10
Yaw, Pitch, Roll: 81.52, 4.62, 4.09
Yaw, Pitch, Roll: 81.47, 4.62, 4.09
Yaw, Pitch, Roll: 81.52, 4.63, 4.11
```

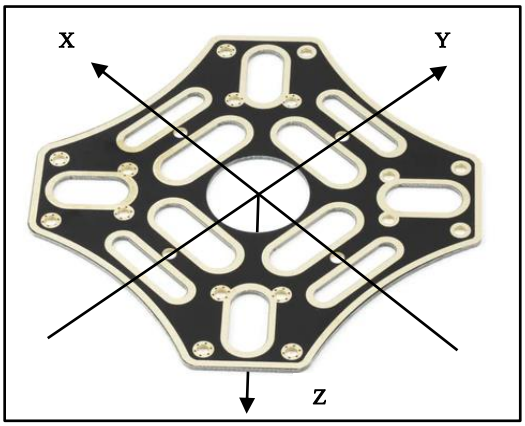
Εικόνα 117. MPU-9250, τα αποτελέσματα και η σύγκριση με την πυξίδα κινητού τηλεφώνου

Η βιβλιοθήκη του MPU-9250 όπως προαναφέρθηκε έχει σαν επιλογή την χρήση του Madgwick ή Mahony φίλτρου. Η διαφορά τους έγκειται στο γεγονός ότι ο Madgwick χρησιμοποιεί ελεγκτή P ενώ ο Mahony τύπου PI. Η default επιλογή είναι αυτή του Madgwick και η υλοποίηση του φαίνεται στο QuaternionFilter.h της βιβλιοθήκης. Κατά τον Sebastian O.H. Madgwick το φίλτρο φαίνεται να επιτυγχάνει καλλίτερα αποτελέσματα από αυτό του Kalman κατά $< 0.6 \circ$ στατικό RMS λάθος και κατά $< 0.8 \circ$ δυναμικό [33].

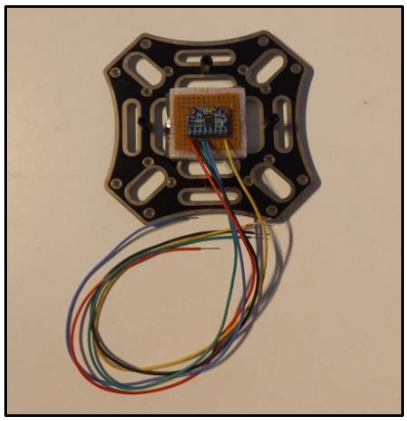
.

3.11 Τοποθέτηση του IMU στο πλαίσιο του Quadcopter

Το IMU MPU-6050 πρέπει να τοποθετηθεί ακριβώς στο κέντρο τομής των αξόνων περιστροφής X,Y,Z . Όσο πιο ακριβής είναι η τοποθέτηση τόσο πιο αληθοφανή θα είναι τα αποτελέσματα. Επίσης εγκαθίσταται πάνω σε αφρώδες υλικό σε μια προσπάθεια να απορροφηθεί το δυνατό περισσότερος θόρυβος προερχόμενος κυρίως από τα μοτέρ περιστροφής και τις προπέλες . Το αποτέλεσμα φαίνεται στις εικόνες που ακολουθούν. Διακρίνονται τα καλώδια σύνδεσης, τροφοδοσία, γείωση, SCL, SDA, και INT.



Εικόνα 119. Σημείο τομής των αξόνων περιστροφής



Εικόνα 118. Το IMU τοποθετημένο στο πλαίσιο του Quadcopter, πάνω σε αφρώδες υλικό.

3.12 Ενσωμάτωση κώδικα PID και επεξήγηση

Στην ενότητα αυτή θα εξηγηθεί η λογική του ελέγχου PID και ο τρόπος που θα γίνει ενσωμάτωση στο πρόγραμμα του Flight Controller. Η λογική είναι ίδια για το Pitch, Roll και Yaw, για το λόγο αυτό θα αναλυθεί μόνο για τον άξονα του Pitch.

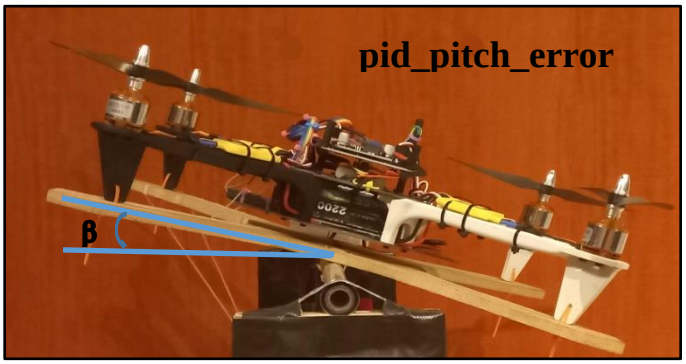
```
//.............................PID Pitch.....................................................................//

    pid_pitch_setpoint = map(pitch_input,1000, 2000, -10, 10);
    if (pid_pitch_setpoint > -2 && pid_pitch_setpoint < 2) pid_pitch_setpoint = 0;

    pid_pitch_error = pitch - pid_pitch_setpoint ;


    pitch_Kp = pid_pitch_error * Kp_for_pitch ;


    if (pitch_input > 1492  && pitch_input <1508 ) pitch_Ki += pid_pitch_error * Ki_for_pitch ;
    else pitch_Ki = 0 ;


    pitch_Kd = (pid_pitch_error - previous_pid_pitch_error) * Kd_for_pitch;


    PID_pitch_total = pitch_Kp + pitch_Ki + pitch_Kd;

    PID_pitch_total = constrain(PID_pitch_total, -PID_pitch_total_max, PID_pitch_total_max);

    previous_pid_pitch_error = pid_pitch_error;

//.............................................................................................//
```

Εικόνα 120. Επεξήγηση κώδικα PID

Κατά αυτόν το τρόπο όπως αποτυπώνεται στην εικόνα 120 είναι :

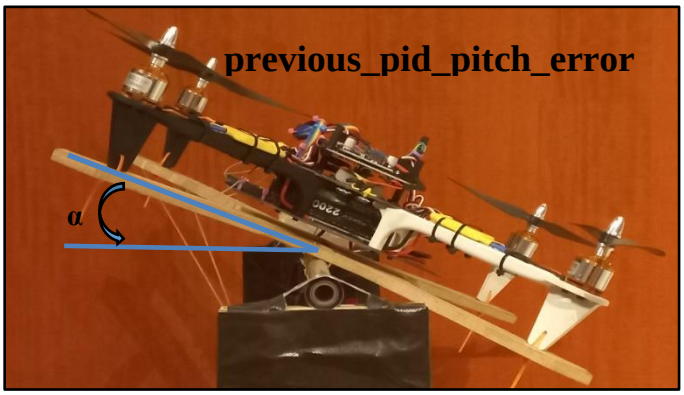
- pid_pitch_setpoint: Η επιθυμητή θέση του Quadcopter. Όταν το stick του pitch είναι neutral το setpoint είναι 0 ή αλλιώς το Quadcopter έχει εντολή να είναι level.
- pid_pitch_error(β): Η διαφορά της τρέχουσας θέσης και της επιθυμητής.
- Kp_for_pitch: Ο συντελεστής Kp του PID του pitch.
- Ki_for_pitch: Ο συντελεστής Ki του PID του pitch.
- Kd_for_pitch: Ο συντελεστής Kd του PID του pitch.
- pitch_Kp: Το αποτέλεσμα του Kp με το error.
- pitch_Ki: Το αποτέλεσμα του Ki με το error.
- pitch_Kp Το αποτέλεσμα του Kd με το error.
- PID_pitch_total: Το άθροισμα όλων των επιμέρους pitch_Kp, pitch_Ki, pitch_Kd.
- previous_pid_pitch_error(α): Το προηγούμενο error του Quadcopter, ακριβώς πριν την εκτέλεση της σειράς των εντολών της εικόνας 123.

Εικόνα 121. Το Quadcopter σε θέση level. Το error είναι 0.



Εικόνα 122. Το Quadcopter σε τυχαία θέση. Το error είναι β.

Έχει ήδη εκτελεστεί μια φορά ο κώδικας του PID.



Εικόνα 123. Το Quadcopter σε τυχαία θέση. Το error είναι α.

Ακριβώς πρίν εκτελεστεί ο κώδικας του PID.

Η θεωρία του PID έχει αναλυθεί σε προηγούμενη ενότητα για το λόγο αυτό θα αναφερθούμε μόνο στο κομμάτι των εντολών που πλαισιώνουν την ολοκλήρωση της υπορουτίνας. Έτσι λοιπόν στην σειρά 11 με την χρήση της εντολής *map* αποτυπώνεται το εύρος του ποτενσιόμετρου του τηλεχειρισμού (1000 – 1500 – 2000) σε μέγιστες γωνίες κλίσης -10^o, 10^o κατά αντιστοιχία. Στην περίπτωση που η θέση του stick είναι neutral λόγω ενός μικρού jitter το σύστημα εισάγει θόρυβο ο οποίος περνά στον υπολογισμό του PID , δίνοντας ψευδές *setpoint* . Έτσι στην γραμμή 12 δημιουργούμε ένα dead band με εύρος -2^o, 2^o όπου το *setpoint* είναι 0. Ο Integral όρος του PID θέλουμε να ενεργεί μόνο όταν το stick είναι σε θέση neutral, λόγω του ότι είναι συσωρευτικός έρχεται να αντιδράσει στην μεγάλης διάρκειας κλίση δίνοντας ανεπιθύμητα αποτελέσματα όταν αφεθεί το ποτενσιόμετρο ελέγχου σε μηδενική θέση. Αυτό επιτυγχάνεται με το set των εντολών των γραμμών 20,21. Μία *constrain* περιορίζει τις μέγιστες και ελάχιστες τιμές του συνολικού PID που τελικά θα οδηγηθούν στα μοτέρ. Ενώ τέλος σώζεται σε μια μεταβλητή το *error* ώστε στο επόμενο loop του κώδικα να χρησιμοποιηθεί ως *previous_error* ώστε να υπολογιστεί στο D μέρος του PID.

Το σύνολο των παραπάνω εντολών χρησιμοποιείται για το υπολογισμό και του Roll , ενώ για το Yaw ο ελεγκτής είναι PI. Στον πίνακα 18 παρουσιάζονται οι τρείς όροι του PID και πως αυτοί υλοποιούνται στο προγραμματιστικό περιβάλλον στο Arduino IDE για παράδειγμα για την γωνία pitch .

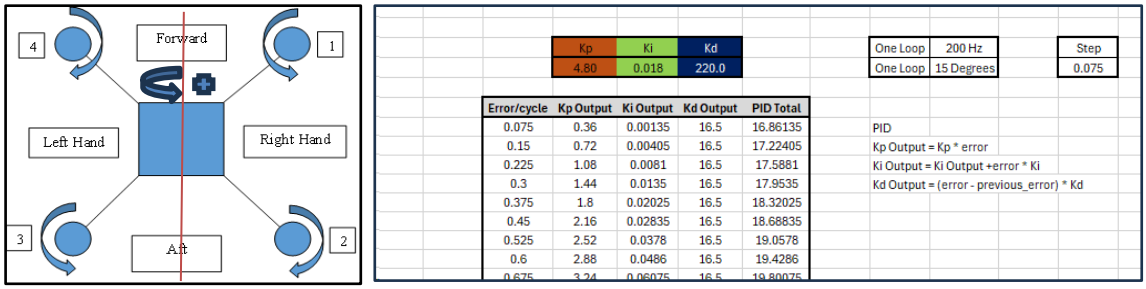
PID	PID on IDE
$u\_P\ (t) = \ K\_P\ \ e(t)$	pitch_Kp = Kp_for_pitch * error
$u_I(t) = \ K_I \int_0^t e(t)dt$	pitch_Ki += pid_pitch_error * Ki_for_pitch
$u_D(t) = \ K_D \ \frac{d}{dt}\ e(t)$	pitch_Kd = (pid_pitch_error - previous_pid_pitch_error) * Kd_for_pitch

Πίνακας 18. Αναπαραγωγή εξισώσεων PID σε επίπεδο κώδικα , Arduino IDE

3.12.1 Ένα παράδειγμα βασισμένο στους συντελεστές P,I,D της κατασκευής

Στην παράγραφο αυτή θα θέσουμε ένα υποτιθέμενο σενάριο εκτροπής της ισορροπίας του τετρακόπτερου και θα εξετάσουμε τον τρόπο που ο PID controller αποκρίνεται. Τα δεδομένα είναι ότι η συχνότητα ανανέωσης του προγράμματος είναι 200 Hz (program loop time) και ότι σε χρονική διάρκεια ενός δευτερολέπτου το τετρακόπτερο έχει παρεκκλίνει , έστω στην γωνία Roll, κατά 15 μοίρες . Θεωρούμε για λόγους ευκολίας ότι ο ρυθμός περιστροφής του τετρακόπτερου είναι σταθερός.

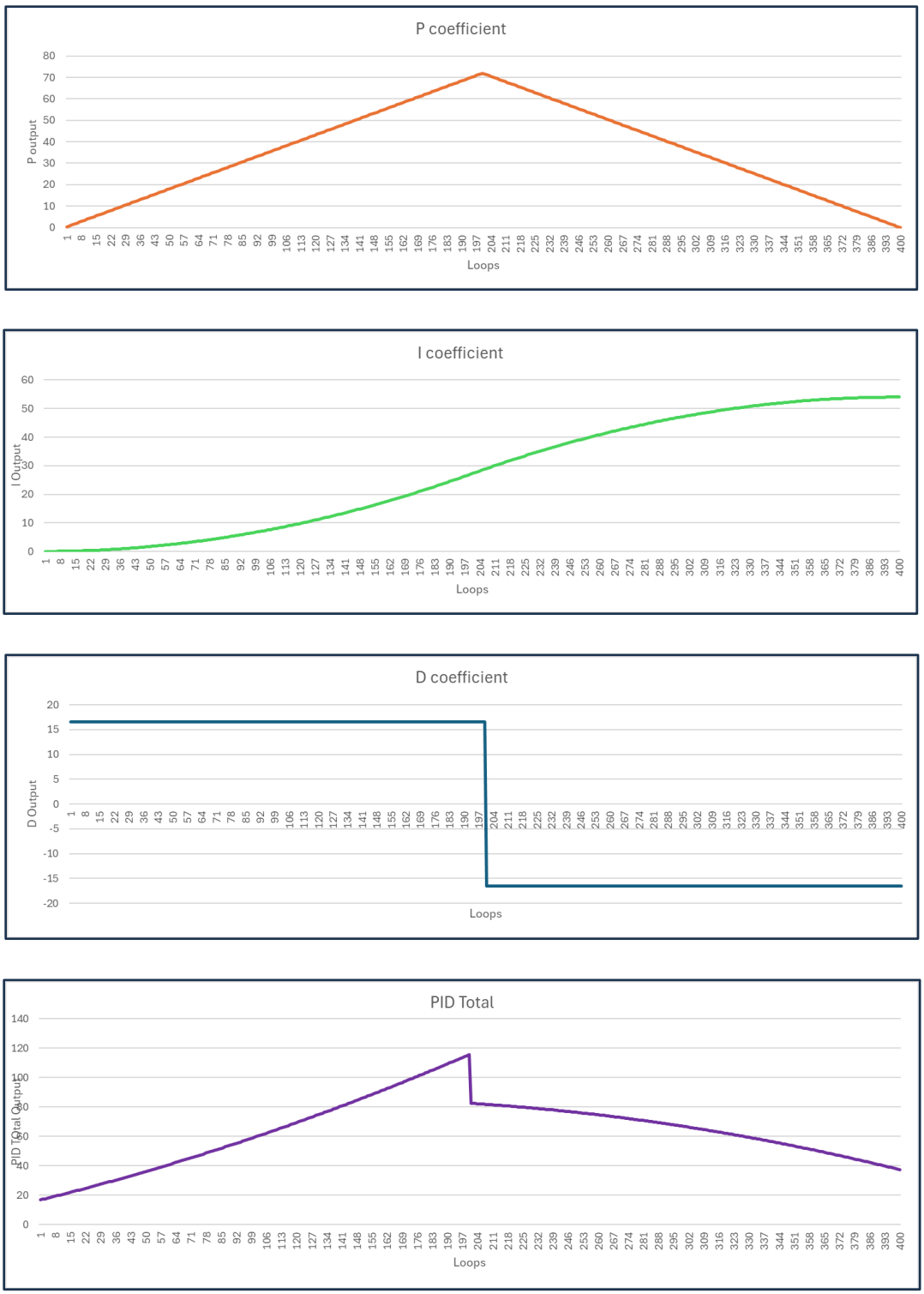
Κατά τον τρόπο αυτό σε χρονική διάρκεια ενός δευτερολέπτου ο κώδικας έχει εκτελεστεί 200 φορές και έχουμε επιτύχει εκτροπή 15 μοίρες, συνεπώς ο ρυθμός περιστροφής του τετρακόπτερου είναι 15/200 = 0,075 μοίρες ανά loop . Το τετρακόπτερο είναι σε θέση ΄΄ κοιτάω μπροστά΄΄ (Looking FWD) και η περιστροφή είναι δεξιόστροφη θετική. Η συνολική έξοδος του PID θα προστεθεί στην ταχύτητα περιστροφής των μοτέρ 1 και 2 ώστε να διορθώσει την εκτροπή. Ο κώδικας θα τρέξει για 2 δευτερόλεπτα, το πρώτο θα είναι της εκτροπής και το δεύτερο της διόρθωσης . Με την χρήση του προγράμματος Microsoft Excel αναπαράγουμε τις εξισώσεις για κάθε παράγοντα αλλά και για το συνολικό PID. Λόγω του ότι χρειαζόμαστε 400 ΄΄step΄΄ θα παρουσιαστεί ένα κομμάτι του Excel ενώ όλο το αρχείο θα βρίσκεται στο συνοδευτικό CD.



Εικόνα 124. Το σενάριο περιστροφής κατά τον άξονα Roll , excel υπολογισμοί

Ακολουθούν οι γραφικές παραστάσεις με το error , με τις αποκρίσεις κάθε συντελεστή και την συνολική έξοδο του PID.

Εικόνα 125. PID error



Εικόνα 126. PID , οι έξοδοι κάθε συντελεστή και το συνολικό PID

Παρατηρούμε ότι το συνολικό PID αυξάνει μέχρι την γωνία των 15 μοιρών και αφού στη συνέχεια διορθώσει την γωνία του τετρακόπτερου, όταν αυτό διέρχεται από την θέση ισορροπίας έχει έξοδο 37.5 (Excel sheet) που θεωρητικά πολύ σύντομα στην συνέχεια θα μηδενίσει.

3.13 Το πρόγραμμα του Flight Controller

Το πρόγραμμα του μικροελεγκτή που θα αποτελεί τον flight Controller του τετρακόπτερου θα παρουσιαστεί στην ολότητα του σε ακόλουθη ενότητα. Σε αυτή την ενότητα θα παρουσιαστούν και εξηγηθούν τα βασικά σημεία του. Κάθε κομμάτι του προγράμματος έχει εμπλουτιστεί με σχόλια ώστε να ξεχωρίζει που αναφέρεται . Κατά αυτό τον τρόπο:

```
//Batch1/3.....DMP implementation.....//

#include "MPU6050_6Axis_MotionApps612.h"

#include "Wire.h"

#include "I2Cdev.h"

MPU6050 mpu;

bool dmpReady = false;  // set true if DMP init was successful

uint8_t mpuIntStatus;   // holds actual interrupt status byte from MPU

uint8_t devStatus;      // return status after each device operation (0 = success, !0 = Rollerror)

uint16_t packetSize;    // expected DMP packet size (default is 42 bytes)

uint16_t fifoCount;     // count of all bytes currently in FIFO

uint8_t fifoBuffer[64]; // FIFO storage buffer

Quaternion q;           // [w, x, y, z]         quaternion container

VectorFloat gravity;    // [x, y, z]            gravity vector

float ypr[3], yaw, pitch, roll;

//.....Batch1/3//
```

Εικόνα 127. Επεξήγηση βασικού κώδικα μέρος α΄

Στην εικόνα 127 παρουσιάζεται το πρώτο κομμάτι του κώδικα του DMP. Ορίζονται οι βιβλιοθήκες, δημιουργείται το object mpu, δηλώνονται οι απαραίτητες μεταβλητές και δημιουργούνται τα Quaternion vector, gravity vector όπως επίσης και οι μεταβλητές yaw, pitch, roll.

Στην εικόνα 128 που ακολουθεί, παρουσιάζονται οι δηλώσεις των μεταβλητών που αφορούν την ενσωμάτωση του PID. Έτσι ορίζονται οι απαραίτητες μεταβλητές και δίνονται σταθερές τιμές στις παραμέτρους P, I, D για όλες τις κινήσεις του τετρακόπτερου στον χώρο, pitch, roll και yaw. Επίσης δίνεται μια μέγιστη τιμή στην έξοδο που μας ενδιαφέρει να φτάσει το ολικό PID. Οι τιμές είναι ενδεικτικές, οι τελικές θα προκύψουν ύστερα από τους τελικούς πειραματισμούς και όταν το τετρακόπτερο βρίσκεται στην τελική του διαμόρφωση.

```
//Start1.....PID implementation.....//

    //.....Roll pid imlementation.....//

    float pid_roll_setpoint, pid_roll_error, previous_pid_roll_error, roll_Kp, roll_Ki, roll_Kd,
PID_roll_total ;

    float Kp_for_roll = 4.80 ;

    float Ki_for_roll = 0.015;

    float Kd_for_roll = 220.0 ;

    float PID_roll_total_max = 250 ;

    //.....Pitch pid implementation.....//

    float pid_pitch_setpoint, pid_pitch_error, previous_pid_pitch_error, pitch_Kp, pitch_Ki,
pitch_Kd, PID_pitch_total  ;

    float Kp_for_pitch = Kp_for_roll;

    float Ki_for_pitch = Ki_for_roll;

    float Kd_for_pitch = Kd_for_roll;

    float PID_pitch_total_max = PID_roll_total_max ;

    //.....Yaw pid implementation.....//

    float pid_yaw_setpoint, pid_yaw_error, previous_pid_yaw_error, yaw_Kp, yaw_Ki,
yaw_Kd, PID_yaw_total  ;

    float Kp_for_yaw = 4.0;

    float Ki_for_yaw = 0.0;

    float Kd_for_yaw = 0.0;

    float PID_yaw_total_max = 250.0;

    //.....End1.//
```

Εικόνα 128. Επεξήγηση βασικού κώδικα μέρος β΄

```
//……General Variables Declaration .....//

int roll_input, pitch_input, throttle_input, yaw_input, throttle, start ;

unsigned long timer_1,timer_2, timer_3, timer_4, motor1, motor2, motor3, motor4,
esc1waveform, esc2waveform, esc3waveform , esc4waveform , current_time, esc_loop_timer,
zero_timer, general_timer ;

byte last_channel_1, last_channel_2, last_channel_3, last_channel_4 ;

//.....//
```

Εικόνα 129. Επεξήγηση βασικού κώδικα μέρος γ΄

Τέλος στην εικόνα 129 δηλώνονται όλες οι υπόλοιπες μεταβλητές που θα χρησιμοποιηθούν στο πρόγραμμα.

```
void setup() {

   Serial.begin(38400);

   //Batch2/3.....DMP implementation.....//

        Wire.begin();

        Wire.setClock(400000);

        mpu.initialize();

        devStatus = mpu.dmpInitialize();

        mpu.setXGyroOffset(129);

        mpu.setYGyroOffset(-12);

        mpu.setZGyroOffset(-48);

        mpu.setXAccelOffset(-2316);

        mpu.setYAccelOffset(868);

        mpu.setZAccelOffset(968);

        mpu.CalibrateAccel(6);

        mpu.CalibrateGyro(6);

        mpu.PrintActiveOffsets();

        mpu.setDMPEnabled(true);

        dmpReady = true;

        packetSize = mpu.dmpGetFIFOPacketSize();

 //.....Batch2/3//

 //.....PCINT related code .....//

 PCICR |= (1 << PCIE0);

 PCMSK0 |= (1 << PCINT0);

 PCMSK0 |= (1 << PCINT1);

 PCMSK0 |= (1 << PCINT2);

 PCMSK0 |= (1 << PCINT3);

 // PCMSK0 |= (1 << PCINT4);     //SWC C for future use

 // PCMSK0 |= (1 << PCINT5);      //SWC B for future use

   DDRD |= B11110000;

  zero_timer = micros();

 start = 0;

 }

//.....//
```

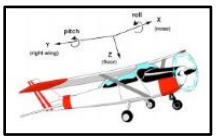
Εικόνα 130. Επεξήγηση βασικού κώδικα μέρος δ΄

Στην εικόνα 130 η υπορουτίνα void setup εκτελείται μια φορά κατά την έναρξη του προγράμματος. Έτσι ξεκινά η επικοινωνία με την σειριακή θύρα και το Serial Monitor στην ταχύτητα των 38400 bps και στην συνέχεια ξεκινά η Wire, τίθεται η συχνότητα του clock της wire στα 400KHz και γίνεται αρχικοποίηση το mpu. Ακολουθεί το calibration του MPU όπως αυτό έχει περιγραφεί σε προηγούμενη ενότητα με τις τιμές να προέρχονται από το sketch IMU_Zero της βιβλιοθήκης που χρησιμοποιούμε. Οι τιμές αυτές είναι μοναδικές για κάθε MPU. Τέλος ετοιμάζεται το πρώτο πακέτο FIFO για να χρησιμοποιηθεί στον πρώτο κύκλο της void loop που ακολουθεί.

Στο δεύτερο κομμάτι ακολουθεί ο κώδικας που αφορά τα interrupts. Στην ενότητα με τα interrupts έχουν εξηγηθεί οι PCICR και PCMSK0. Με την DDRD τίθενται ως έξοδοι οι ψηφιακές θύρες 4,5,6,7 (port manipulation). Οι ψηφιακές θύρες 8 και 9 ενδεχόμενα να εξυπηρετήσουν τα υπόλοιπα 2 κανάλια του τηλεχειρισμού. Στη συνέχεια γίνεται αρχικοποίηση της συνάρτησης micros() στην μεταβλητή zero_timer. Χρησιμοποιείται παρακάτω για να ρυθμίσει το loop του προγράμματος. Η μεταβλητή start χρησιμοποιείται για το arm / disarm του τετρακόπτερου . Δίνεται η τιμή 0 για λόγους ασφαλείας (disarm) .

Στην εικόνα 131 παρουσιάζεται το τελευταίο κομμάτι κώδικα που αφορά την εξαγωγή των γωνιών pitch, roll, yaw από το Quaternion. Έτσι αφού υπάρξει διαθέσιμο πακέτο FIFO προερχόμενο από το αντίστοιχο buffer, σχηματίζεται το Quaternion. Ακολουθεί η εξαγωγή του διανύσματος gravity από το Quaternion και τελικά παράγονται οι τρείς γωνίες σε rad . Γίνεται μετατροπή από rad μοίρες. Στο pitch δίνονται αρνητικές τιμές ώστε να ταιριάξουν με τον συμβατικό κανόνα του αεροπλάνου ,τον κανόνα του δεξιού χεριού και τον προσανατολισμό του MPU-6050.

```
//Batch3/3.....DMP implementation.....//

void dmp_yaw_pitch_roll() {

        if (mpu.dmpGetCurrentFIFOPacket(fifoBuffer)) {

        mpu.dmpGetQuaternion(&q, fifoBuffer);

        mpu.dmpGetGravity(&gravity, &q);

        mpu.dmpGetYawPitchRoll(ypr, &q, &gravity);

        yaw = ypr[0] * (180 / M_PI);

        pitch = -ypr[1] * (180 / M_PI) ;

        roll = ypr[2] * (180 / M_PI);

        }
```



Εικόνα 131. Επεξήγηση βασικού κώδικα μέρος ε΄

```
void pidcalculation(){

 //.....PID ROLL.....//

  pid_roll_setpoint = map(roll_input,1000, 2000, -10.0, 10.0);

  if (pid_roll_setpoint > -2.0 && pid_roll_setpoint < 2.0) pid_roll_setpoint = 0.0;

  pid_roll_error = roll - pid_roll_setpoint ;

  roll_Kp = pid_roll_error * Kp_for_roll ;

  if (roll_input > 1492  && roll_input <1508 ) roll_Ki += pid_roll_error * Ki_for_roll ;

  else roll_Ki = 0 ;

  roll_Kd = (pid_roll_error - previous_pid_roll_error) * Kd_for_roll;

  PID_roll_total = roll_Kp + roll_Ki + roll_Kd;

  PID_roll_total = constrain(PID_roll_total, -PID_roll_total_max, PID_roll_total_max);

  previous_pid_roll_error = pid_roll_error;
```

Εικόνα 132. Επεξήγηση βασικού κώδικα μέρος ζ΄

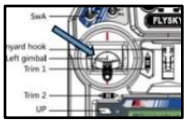
Η υπορουτίνα pidcalculation αφορά τον υπολογισμό του PID για κάθε μια από τις τρείς κινήσεις. Θα εξηγηθεί μόνο για την κίνηση roll αφού είναι παρόμοια και για τις άλλες .

Στην πρώτη έκδοση του τετρακόπτερου είχαμε δει ότι για να δώσουμε κλίση στο σκάφος π.χ. αριστερά θα έπρεπε να αυξήσουμε το thrust των δεξιών μοτέρ. Για την περίπτωση που έχει ενσωματωθεί ο κώδικας του PID αυτό δεν βοηθάει διότι όπως αναφέρθηκε στο θεωρητικό μέρος, το PID προσπαθεί να ισορροπήσει στο setpoint. Έτσι αν το setpoint είναι μηδέν (level) και προσπαθούμε να δώσουμε κλίση στον ίδιο άξονα, αυξάνοντας το thrust των αντιθέτων μοτέρ, το PID θα αντιστέκεται δίνοντας μη επιθυμητά αποτελέσματα. Για αυτό το λόγο θα δίνουμε κλίση αλλάζοντας το setpoint και αφήνοντας το PID να ρυθμιστεί γύρω από το νέο setpoint. Αυτό επιτυγχάνεται κάνοντας map την τιμή του roll του τηλεχειρισμού (από 1000 σε 2000 μsec) στο pid_roll_setpoint (-10 σε +10 degrees) αντίστοιχα. Δημιουργείτε στη συνέχεια το pid_roll_error και οι γνωστές πια εξισώσεις που θα παράγουν το PID_roll_total.

Η συνάρτηση micros() έχει resolution 4 msec. Έτσι για παράδειγμα όταν το stick του Roll του τηλεχειρισμού είναι center – neutral, το roll ενδέχεται να έχει την τιμή 1496 μsec έως 1504 μsec. Αυτό το μικρό jitter δημιουργεί θόρυβο στο PID με μη αποδεκτά αποτελέσματα . Για αυτό δημιουργείται ένα dead band (1η ΄΄if΄΄), από -2 μοίρες έως +2 μοίρες που θέτουμε το setpoint 0 (level). Παράλληλα επειδή ο παράγοντας I του PID είναι συσσωρευτικός (cumulative), όταν το τετρακόπτερο είναι σε κλίση για αρκετή ώρα, η έξοδος του I αυξάνεται αρκετά δίνοντας μη αποδεκτά αποτελέσματα. Για αυτό το λόγο ρυθμίζουμε το I να λειτουργεί μόνο όταν το stick του roll είναι σε center – neutral θέση(2η ΄΄if΄΄). Τέλος μια constrain έρχεται να περιορίσει την έξοδο του PID_roll_total στις επιθυμητές τιμές που έχουν οριστεί παραπάνω .

Το κομμάτι του κώδικα στην εικόνα 133 ασχολείται με το arm / disarm των μοτέρ του τετρακόπτερου, λαμβάνοντας δεδομένα από τις εντολές του stick που αφορά το Throttle και το Yaw. Επίσης τίθεται η μεταβλητή throttle ως η βασική μεταβλητή που θα οδηγεί τα μοτέρ του drone.

```
if (throttle_input<=1100 && yaw_input<1050 && yaw_input>1000 ) start=1;

 if (start==1 && throttle_input<=1100 && yaw_input<1520 && yaw_input>1480  ) start =2;

if (start==2 && throttle_input<=1100 && yaw_input>1950) start=0;

throttle = throttle_input;
```



Εικόνα 133. Επεξήγηση βασικού κώδικα μέρος η΄

```
if (start == 2){

  if (throttle >1600) throttle = 1600;

    motor1 = throttle + PID_roll_total + PID_pitch_total - PID_yaw_total ;

    motor2 = throttle + PID_roll_total - PID_pitch_total + PID_yaw_total ;

    motor3 = throttle - PID_roll_total - PID_pitch_total - PID_yaw_total ;

    motor4 = throttle - PID_roll_total + PID_pitch_total + PID_yaw_total ;

    if (motor1 < 1100) motor1 = 1100;

    if (motor2 < 1100) motor2 = 1100;

    if (motor3 < 1100) motor3 = 1100;

    if (motor4 < 1100) motor4 = 1100;

    if(motor1 > 2000)motor1 = 2000;

    if(motor2 > 2000)motor2 = 2000;

    if(motor3 > 2000)motor3 = 2000;

    if(motor4 > 2000)motor4 = 2000;      }

    else {

    motor1 = 1000;

    motor2 = 1000;

    motor3 = 1000;

    motor4 = 1000;

    roll_Ki = 0;

    previous_pid_roll_error = 0;

    pitch_Ki = 0;

    previous_pid_pitch_error = 0;   }
```

Εικόνα 134. Επεξήγηση βασικού κώδικα μέρος θ΄

Αν τα μοτέρ είναι σε κατάσταση arm (start =2), τότε με μια ΄΄if΄΄ θέτουμε την μέγιστη τιμή που θέλουμε να πάρει το throttle (π.χ. 1600). Στις επόμενες γραμμές ορίζονται οι τελικές τιμές των μοτέρ μετά την εφαρμογή του PID. Έτσι για την περίπτωση που το PID_roll_total είναι θετικό τα μοτέρ 1,2 αυξάνουν το thrust τους, τα 3,4 το μειώνουν και τελικά το τετρακόπτερο κάνει roll

αριστερά(εικόνα 82). Κατά αντιστοιχία και για το pitch και roll. Ακολουθούν εντολές ΄΄if΄΄ με ρόλο constrain περιορίζοντας το συνολικό thrust σε μέγιστο 2000 μsec και ελάχιστο 1100 μsec. Τέλος αν τα μοτέρ γίνουν disarm (start <>2) τα μοτέρ παίρνουν την τιμή 1000 μsec και σταματάνε να γυρίζουν. Όταν συμβεί αυτό γίνονται reset και οι μεταβλητές roll_Ki, previous_pid_roll_error, pitch_Ki , previous_pid_pitch_error ώστε στο επόμενο arm να έχουν τιμές 0 (bump less start). Το τελευταίο κομμάτι του προγράμματος αφορά τα interrupts και την αναπαραγωγή των PWM παλμών από τον δέκτη του τηλεχειρισμού. Έτσι λοιπόν για παράδειγμα για το digital pin 8 στην interrupt service routine, reset το χρονόμετρο στην μεταβλητή current_time με την συνάρτηση micros(), αν το pin 8 είναι Low (0) κάνε το High (1) και κάνε reset τον timer_1 αλλιώς αν είναι High(1), κάνε το Low(0) και υπολόγισε το εύρος του παλμού αφαιρώντας από τον συνολικό χρόνο τον χρόνο που ήταν HIGH (1). Η ίδια διαδικασία επαναλαμβάνεται για όλα τα digital pin που συμμετέχουν στην ISR.

```
ISR(PCINT0_vect){
  current_time = micros();
  if(PINB & B00000001){
    if(last_channel_1 == 0){
      last_channel_1 = 1;
      timer_1 = current_time;
    }
  }
  else if(last_channel_1 == 1){
    last_channel_1 = 0;
    roll_input = current_time - timer_1;
  }
```

Εικόνα 135. Επεξήγηση βασικού κώδικα μέρος ι΄

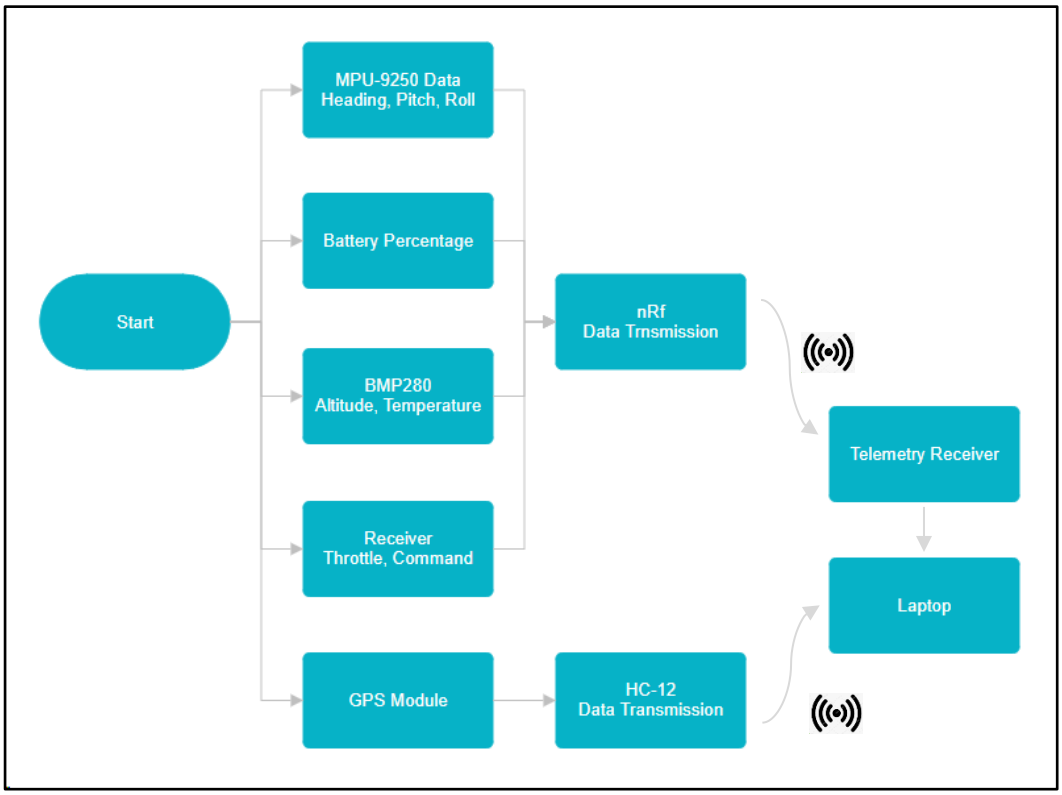
3.14 Secondary MCU

Η υλοποίηση του τετρακόπτερου όπως έχει προαναφερθεί περιλαμβάνει δύο μικροελεγκτές. Ο ένας έχει αποκλειστικά τον ρόλο του Flight Controller ενώ ο δεύτερος αναλαμβάνει την υποχρέωση να εξυπηρετήσει όλους τους υπόλοιπους αισθητήρες. Η λογική πίσω από την επιλογή 2 Arduino Nano είναι να αποφορτιστεί ο βασικός ελεγκτής από τις παράλληλες εργασίες και να γίνει το program loop χρονικά όσο το δυνατό συντομότερο. Αυτό θα επιτύχει μεγαλύτερο refresh rate στα ESC ΄s των μοτέρ και τελικά καλλίτερο έλεγχο.

Το δεύτερο κομμάτι της κατασκευής είναι μια ξεχωριστή πλακέτα που θα φιλοξενεί τους παρακάτω αισθητήρες και ηλεκτρονικά εξαρτήματα :

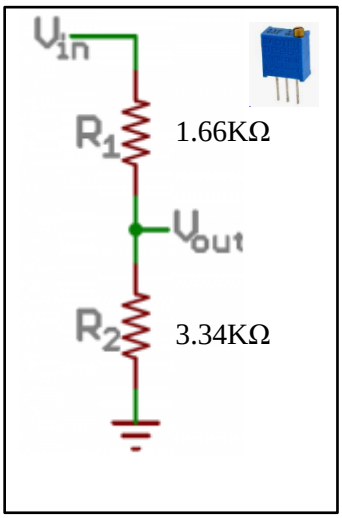
- MPU-9250
- BMP280
- GPS NEO M8N
- Receiver inputs
- Battery voltage circuit
- nRF24L01
- DC-DC Step Down Converter 5V to 3V3

Το μπλοκ διάγραμμα της εικόνας 136 παρουσιάζει συνοπτικά την υλοποίηση του κυκλώματος.



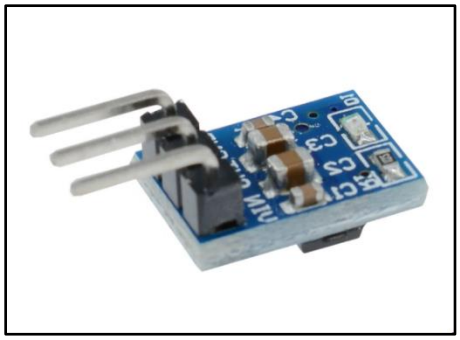
Εικόνα 136. Μπλοκ διάγραμμα Secondary MCU

Το MPU-9250 έχει παρουσιαστεί στην παράγραφο 3.10.3 . Με την χρήση της βιβλιοθήκης του Hideaki Tai εξάγουμε τις γωνίες pitch, roll και heading. Το BMP280 έχει παρουσιαστεί στην παράγραφο 2.13. Για τις εντολές του τηλεχειρισμού, με splice παίρνουμε τις ίδιες τιμές που οδηγούν τον flight controller και με την χρήση interrupts αποκωδικοποιούμε την τιμή τους. Το GPS module , NEO M8N είναι ένα ανεξάρτητο κύκλωμα που τροφοδοτείται και εξάγει δεδομένα σε μορφή UART . Αυτά εισέρχονται στον πομποδέκτη HC-12 και εκπέμπονται ασύρματα προς το Laptop (§2.12, §2.14). Αναφορικά με το ποσοστό φόρτισης της μπαταρίας , εκεί χρησιμοποιείται ένας διαιρέτης τάσης παίρνοντας το 1/3 της τιμής της μπαταρίας (12.6V σε 4.2V) ώστε να μπορέσει σε μια analog input (5V max) να μετατραπεί σε ψηφιακή μορφή. Με την χρήση εξωτερικού τροφοδοτικού έχει βαθμονομηθεί η περιοχή μέγιστης και ελάχιστης μπαταρίας (12.6V, 11.1 V). Αυτή η περιοχή με την εντολή map έχει αντιστοιχηθεί σε ποσοστό 100% και 15%. Σαν διαιρέτης τάσης επιλέχθηκε ένα πολύστροφο trimmer μεγέθους 5KΩ.



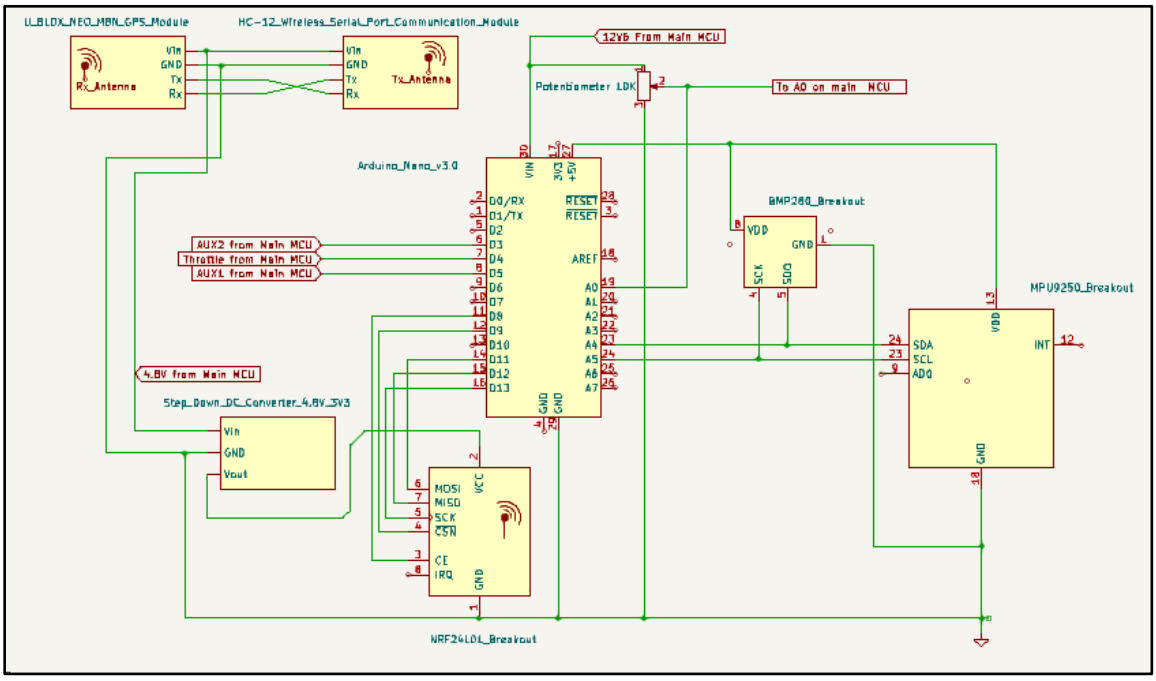
Εικόνα 137. Διαιρέτης τάσης

Για το nRF24L01 η συνδεσμολογία είναι SPI και όλες οι πληροφορίες υπάρχουν στην παράγραφο 2.11. Η τροφοδοσία του είναι επιπέδου TTL, για τον λόγο αυτό ένα DC-DC Step Down Converter , το AMS117, έχει μετατρέψει τα 5V σε 3V3 σταθεροποιημένα .

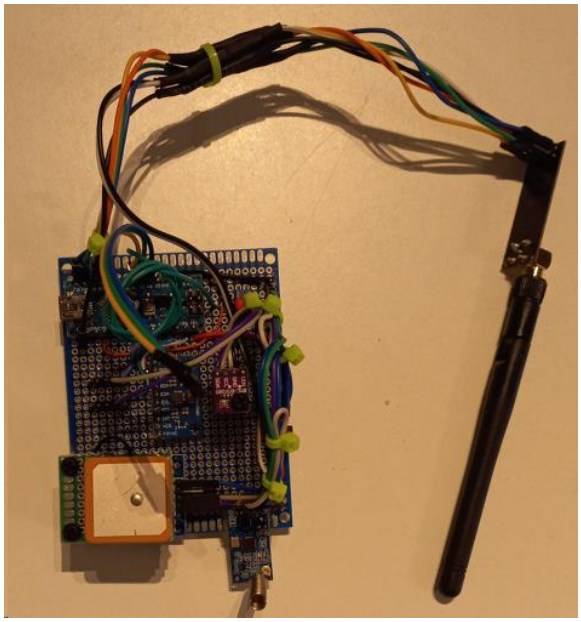


Εικόνα 138. AMS1117 Step Down Converter 800 mA

Στην εικόνα 139 παρουσιάζεται το ηλεκτρονικό σχέδιο της δεύτερης πλακέτας, Secondary MCU, στο πρόγραμμα KiCad.



Εικόνα 139. Secondary MCU, το ηλεκτρονικό σχέδιο.



Εικόνα 140. Secondary MCU

3.15 Το πρόγραμμα του Secondary MCU

```
#include "MPU9250.h"

#include <ErriezBMX280.h>

#include <SPI.h>

#include <nRF24L01.h>

#include <RF24.h>

MPU9250 aux_mpu;                                        // MPU 9250 object

ErriezBMX280 bmp280 = ErriezBMX280(0x76);               // BMP 280 object

//*****************************RF24 Object & Data_Package*********************************//

RF24 radio(8, 9); // CE, CSN         `                                                      //

const byte address[6] = "00001";                                                           //

   struct Data_Package {                                                                   //

   float  heading_tx, altitude_tx, temp_tx, throttle_tx, command_tx, percentage_tx, pitch_tx, roll_tx ;  };      //

   Data_Package data;                                                                      //

//**** ************************************************************************************//

float tmp_pres, tmp_acum, altitude, temp ;                  // BMP 280 variables declaration

float yaw,pitch,roll ;                                      // MPU 9250 variables declaration

unsigned long timer_1, timer_2, current_time, zero_timer ;  // Variables related to the interrupts

byte last_channel_1, last_channel_2 ;                       // Same as above

int  throttle_input, command_input, throttle, command ;     // Same as above

float Vbat_hot, percentage ;                                // Battery percentage related variables
                                                            declaration

void setup() {

   Wire.begin();

   delay(500);

   pinMode(A0,INPUT);

   PCICR |= (1 << PCIE2);                  //Pin Change Interrupt Control Register E2 PCINT 16 - 23

   PCMSK2 |= (1 << PCINT20);               //Set PCINT20(digital input 4)to trigger an interrupt on state
                                           change,  Throttle input

   PCMSK2 |= (1 << PCINT21);               //Set PCINT20(digital input 5)to trigger an interrupt on state
                                           change,  Command input
```

Αρχικά δηλώνονται οι βιβλιοθήκες που πρόκειται να χρησιμοποιηθούν και στην συνέχεια δημιουργούνται τα objects. Για το RF24 δημιουργείται ένα struct Data Package το οποίο θα πάρει τις τρέχουσες τιμές των μεταβλητών και θα εκπεμφθεί. Ακολουθούν οι δηλώσεις των μεταβλητών. Στην void setup γίνεται αρχικοποίηση της wire για να υποστηρίξει τους IIC αισθητήρες. Η αναλογική A0 δηλώνεται ως input και είναι αυτή που θα λάβει την τάση από την μπαταρία (max 4.2V, min 3.7V). Ακολουθεί η PCICR ώστε να υποστηρίξει τα digital pins 4 (Throttle) και 5 (Command) ως interrupts. Το interrupt Command αφορά έναν διακόπτη τριών θέσεων του control panel του τηλεχειρισμού (1000μsec,1500μsec, 2000μsec) του οποίου η κατάσταση εκπέμπεται και παρέχει περαιτέρω επιλογές απεικόνισης στο telemetry receiver.

Εικόνα 141. Secondary MCU, επεξήγηση κώδικα μέρος α΄

```
  //*********************************MPU 9250 Setup ***********************//
   aux_mpu.setup(0x68);                                                    //
   aux_mpu.setAccBias(-40.62, 31.50, 62.18);                               //
   aux_mpu.setGyroBias(1.22, -0.98, 0.28);                                 //
   aux_mpu.setMagBias(62.11, -76.56, -125.70);                             //
   aux_mpu.setMagScale(0.92, 0.98, 1.11);                                  //
   aux_mpu.setMagneticDeclination(5.1);                                    //
//*************************************************************************//
//*********************************BMP 280 Setup ***************************************//
   bmp280.begin();                                                                       //
   bmp280.setSampling(BMX280_MODE_NORMAL,     // SLEEP, FORCED, NORMAL                 //
                BMX280_SAMPLING_X2,          // Temp:  NONE, X1, X2, X4, X8, X16       //
                BMX280_SAMPLING_X4,          // Press: NONE, X1, X2, X4, X8, X16       //
                BMX280_SAMPLING_NONE,    // Hum:   NONE, X1, X2, X4, X8, X16 (BME280)  //
                BMX280_FILTER_X16,           // OFF, X2, X4, X8, X16                   //
                BMX280_STANDBY_MS_0_5);  // 0_5, 10, 20, 62_5, 125, 250, 500, 1000     //
   delay(200);                                                                           //
   for (int cal = 0; cal < 20 ; cal ++){           //Average of 20 readings loop        //
   tmp_pres = bmp280.readPressure() / 100.0F;                                            //
   delay(50);                                                                            //
   tmp_acum += tmp_pres;                                                                 //
   }                                                                                     //
//*************************************************************************************//
//*********************************nRFSetup*********************************//
  radio.begin();                                                                  //
  radio.openWritingPipe(address);              // pipe address 00001              //
  radio.setChannel(100);                       // set channel = 2400 + (number) , 2400+100 = 2500 Mhz //
  radio.setPALevel(RF24_PA_MAX);               // RF24_PA_MIN=-18dBm, RF24_PA_LOW=-12dBm,
                                               RF24_PA_HIGH=-6dBM, and RF24_PA_MAX=0dBm  //
  radio.setDataRate(RF24_250KBPS);             // RF24_250KBPS for 250kbs, RF24_1MBPS for 1Mbps, or
                                               RF24_2MBPS for 2Mbps                      //
  radio.stopListening();                       // Transmit only                   //
//*************************************************************************************//
zero_timer = micros();
}
```

Στο κομμάτι αυτό του προγράμματος δηλώνονται οι παράμετροι του MPU-9250 όπως έχει αναφερθεί σε προηγούμενη παράγραφο. Στην συνέχεια ακολουθεί η παραμετροποίηση του BMP280 που αφορά το ρυθμό δειγματοληψίας για τις : θερμοκρασία, πίεση , φίλτρο και duty cycle . Ακολουθεί ένα loop από 20 επαναλήψεις . Παίρνει τις τρέχουσες τιμές πίεσης και τις αθροίζει στην μεταβλητή tmp_acum . Αργότερα θα τις διαιρέσει και θα προκύψει η μέση τιμή αναφοράς πίεσης. Ακολουθεί η αρχικοποίηση του nRF δηλώνοντας την διεύθυνση pipe, το κανάλι της ζεύξης, το επίπεδο εκπομπής,το baud rate και ότι τίθεται ως πομπός. Τέλος στην μεταβλητή zero_timer η συνάρτηση micros ή οποία ενδέχεται να χρησιμοποιηθεί ως χρονόμετρο.

Εικόνα 142. Secondary MCU, επεξήγηση κώδικα μέρος β΄

```
void loop() {

//*********************************MPU 9250values*********************************//

    if (aux_mpu.update()) {                                                       //

         yaw = aux_mpu.getYaw();                                                  //

        pitch = aux_mpu.getPitch();                                               //

        roll = aux_mpu.getRoll();                                                 //

        if (yaw <0.0 ) yaw  = yaw +360.0 ;                                        //

    }                                                                             //

//*************************************************************************************//

//************************BatteryPercentage*****************************************//

    Vbat_hot  = analogRead(A0);                                  // 845 => 12.6 Volt,746  =>11.12 Volt//

    Vbat_hot = (0.999F* Vbat_hot) + (0.001F * analogRead(A0)); // Complementary filter  //

    percentage = map(Vbat_hot,845,746,100,15);                  // map Vbat_hot to percentage  //

    if (percentage > 100) percentage  = 100 ;                   //Upper Limit  //

    if (percentage <15 ) percentage = 15 ;                      // Lower Limit  //

//*************************************************************************************//

//*****************************BMP 280 Altitude & Temperature**************************//

    static const float pressure_cal = tmp_acum / 20.0F ;  // Ave. reference pressure for altitude calculation
                                                                                  //

    altitude = bmp280.readAltitude(pressure_cal);    //  altitude calculation in terms of reference pressure
                                                                                  //

    if ( altitude  < 0.0 ) altitude = 0.0 ;          //  canceling negative values  //

    temp = bmp280.readTemperature();                 // Temperature reading  //

//*************************************************************************************//

//************throttle & command assignment to variables to be transmitted*****************//

    throttle  = throttle_input;                                                   //

    command   = command_input;                                                    //

//*************************************************************************************//

//***************************Data Structure to be transmitted*****************************//

    data.heading_tx = yaw ;                                                       //

    data.pitch_tx = pitch ;                                                       //

    data.roll_tx = roll ;                                                         //

    data.altitude_tx = altitude ;                                                 //

    data.throttle_tx = throttle;                                                  //

    data.temp_tx = temp ;                                                         //

    data.command_tx = command ;                                                   //

    data.percentage_tx = percentage ;                                             //

    radio.write(&data, sizeof(Data_Package));          // Transmit Data Package   //

}
```

Στην void loop γίνεται εξαγωγή των γωνιών yaw, pitch, roll και το Heading(yaw) ρυθμίζεται στην μορφή 0^0 - 360^0. Ακολουθεί η analogRead της A0. Πρόκειται για το 1/3 της τάσης τροφοδοσίας ώστε να μπορέσει να μετατραπεί σε ψηφιακή μορφή στον ADC του Arduino. Με ένα complementary filter παίρνουμε μέρος της τρέχουσας τιμής ώστε να φιλτράρουμε τις απότομες αυξομειώσεις της τάσης. Μια map εντολή αντιστοιχεί τις βαθμονομημένες τιμές της τάσης σε μέγιστο και ελάχιστο ποσοστό ενώ τίθενται άνω και κάτω όρια . Για το BMP280 , διαιρείται η τιμή των 20 μετρήσεων με τον αριθμό των μετρήσεων ώστε να προκύψει η μέση τιμή. Αυτή δίνεται σαν τιμή αναφοράς για να προκύψει το υψόμετρο ΄΄0΄΄ . Το υψόμετρο αυτό είναι το σχετικό ΄΄0΄΄ του τετρακόπτερου την στιγμή της εκκίνησης και όχι το πραγματικό. Για να προκύψει το πραγματικό χρειαζόμαστε την τοπική ατμοσφαιρική πίεση. Ακολουθούν δύο ακόμη μεταβλητές αυτές των throttle και command. Τέλος όλες οι τιμές των μεταβλητών αντιστοιχίζονται στις τιμές του Data Package και τελικά αυτό εκπέμπεται. Όλες είναι float μεγέθους 4 bytes , συνολικά 32 bytes, το μέγιστο μέγεθος πακέτου που μπορεί να εκπεμφθεί σε ένα κύκλο προγράμματος.

Εικόνα 143. Secondary MCU, επεξήγηση κώδικα μέρος γ΄

```
//**Interrupt Service Routine to track and measure state change
//and duty cycle of  receiver waveform respectively**        //
ISR(PCINT2_vect){                                            //
  current_time = micros();                                   //
  if(PIND & B00010000 ){                                     //
    if(last_channel_1 == 0){                                 //
      last_channel_1 = 1;                                    //
      timer_1 = current_time;                                //
    }                                                        //
  }                                                          //
  else if(last_channel_1 == 1){                              //
    last_channel_1 = 0;                                      //
    throttle_input = current_time - timer_1;                 //Throttle
  }                                                          //
  if(PIND & B00100000 ){                                     //
    if(last_channel_2 == 0){                                 //
      last_channel_2 = 1;                                    //
      timer_2 = current_time;                                //
    }                                                        //
  }                                                          //
  else if(last_channel_2 == 1){                              //
    last_channel_2 = 0;                                      //
    command_input = current_time - timer_2;                  //Command
  }                                                          //
}                                                            //
 //*******************End************************//
```

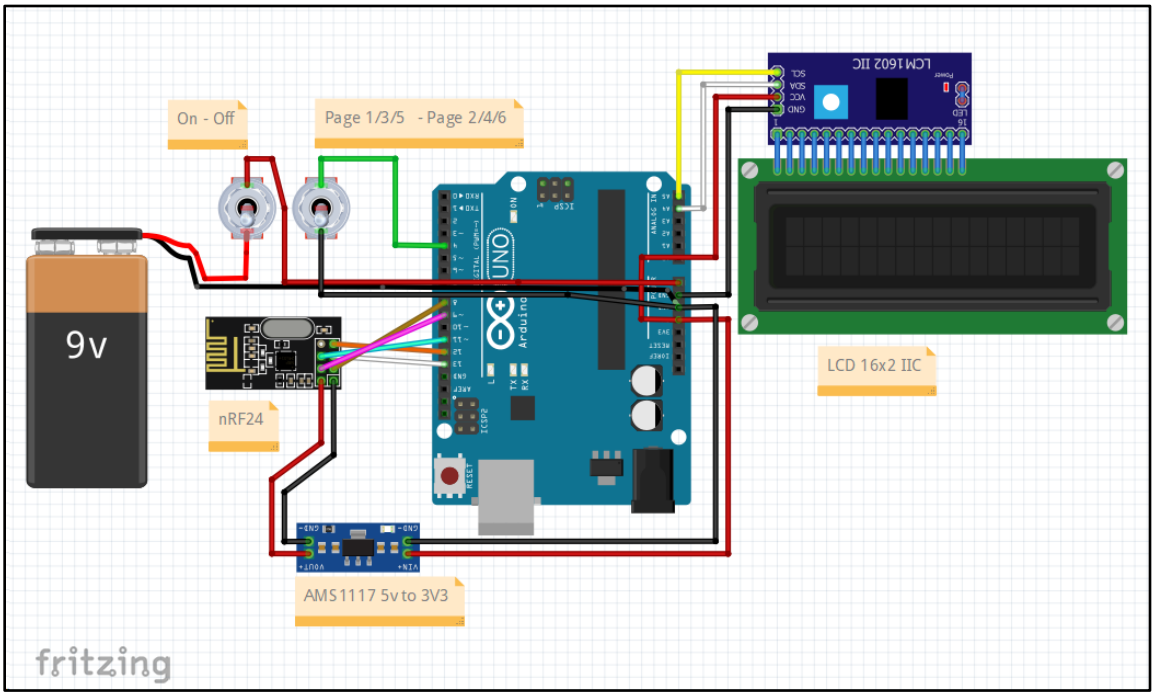
// Copy – Paste Compilation verified//

Τέλος παρουσιάζεται η ISR (Interrupt Service Routine) για υπολογίσει το PWM του τηλεχειρισμού σχετικά με το throttle και το command .

Εικόνα 144. Secondary MCU, επεξήγηση κώδικα μέρος δ΄

3.16 Telemetry Receiver

Ο δέκτης της τηλεμετρίας ολοκληρώνεται σε ένα πλαστικό κουτί μεγέθους 12.5 x 8 x 3.2 εκατοστά που φιλοξενεί ένα Arduino Uno , ένα 16x2 LCD, το nRF24, ένα σταθεροποιητή τάσης AMS1117, δύο μίνι διακόπτες , μια μπαταρία 9V και τις απαραίτητες καλωδιώσεις .



Εικόνα 145. Telemetry Receiver - Fritzing

Ο σκοπός του δέκτη είναι να λάβει ασύρματα τα δεδομένα, να τα αποκωδικοποιήσει και να αναπαραστήσει στην LCD οθόνη . Η οθόνη LCD 16x2 μπορεί να απεικονίσει 32 χαρακτήρες ASCII , λίγοι για τις απαιτήσεις της εφαρμογής. Για τον λόγο αυτό, με την χρήση ενός διακόπτη δύο καταστάσεων και με την χρήση της εντολής τριών καταστάσεων από τον τηλεχειρισμό ΄΄ Command΄΄ μπορούμε συνολικά να έχουμε διαθέσιμες 2x3 = 6 διαφορετικές οθόνες για την απεικόνιση δεδομένων. Τελικά χρησιμοποιούνται 3 οθόνες και 3 είναι διαθέσιμες για μελλοντική χρήση.

SWITCH (State)	COMMAND (μsec)	PAGE
LOGIC AND		
LOW	1000	1
LOW	1500	2
LOW	2000	3
HIGH	1000	4
HIGH	1500	5
HIGH	2000	6

Εικόνα 146. Telemetry Receiver – έξι διαφορετικές εικόνες

Οι δύο οθόνες απεικονίζουν δεδομένα από το τετρακόπτερο όπως :

- Throttle [(1000 – 2000) μsec]
- Battery percentage (15% - 100%)
- Temperature (Celsius)
- Altitude (relative meters)
- Heading (Azimuth 0-360 degrees)
- Command [(1000 – 1500 – 2000) μsec]
- Pitch (degrees)
- Roll (degrees)

Εικόνα 147. Telemetry Receiver, page 1

Εικόνα 148. Telemetry Receiver, page 2

Ο δέκτης έχει την δυνατότητα να συνδεθεί μέσω της σειριακής του Arduino Uno με το Excel και με το Add-In Data Streamer, να αποδίδει ζωντανά αλλά και να καταγράφει όλα τα δεδομένα από το τετρακόπτερο . Έτσι η τρίτη οθόνη απλώς ενημερώνει ότι ο δέκτης έχει συνδεθεί με τον υπολογιστή και μεταφέρει ενσύρματα τα δεδομένα στο Excel.

Εικόνα 149. Telemetry Receiver, Data Streaming

Data In (From Source)

Data coming from the current data source will appear below as it is received.

Current Data

Time	Throttle	Battery(%)	Temperature(Celcius)	Altitude(m)	Pitch	Roll	Heading	Command
53:14.0	1000	92	23.08	0.58	-0.02	1.19	356.05	1496

Historical Data

Time	Throttle	Battery(%)	Temperature(Celcius)	Altitude(m)	Pitch	Roll	Heading	Command
52:09.4	1148	92	22.77	0.62	-2.16	0.67	357.65	1500
52:09.5	1156	92	22.77	0.6	3.07	1.47	357.37	1496
52:09.7	1252	92	22.77	0.57	1.58	-1.1	355.66	1496
52:09.8	1388	82	22.77	0.62	3.5	-1.62	354.34	1500
52:09.9	1516	85	22.77	0.66	7.33	-1.93	358.36	1504
52:10.1	1576	86	22.77	0.66	3.11	-3.17	355.84	1496
52:10.2	1592	86	22.77	0.66	0.16	3.44	355.49	1500
52:10.3	1584	83	22.77	0.54	-1.07	3.73	355.86	1496
52:10.5	1528	87	22.77	0.6	1.09	3.28	355.64	1496
52:10.6	1432	85	22.77	0.52	-0.82	1.35	355.53	1500
52:10.7	1352	88	22.76	0.55	0.95	0.38	354.44	1496
52:10.8	1300	90	22.76	0.58	-0.07	3.34	355.1	1500
52:11.0	1232	86	22.76	0.58	-6.07	5.11	358.23	1484
52:11.1	1228	90	22.76	0.52	-6.65	5.19	358.79	1496
52:11.2	1212	88	22.76	0.61	-6.1	3.67	358.05	1500
52:11.4	1208	88	22.76	0.61	1.64	2.05	357.33	1496
52:11.5	1212	89	22.76	0.61	3.1	3.09	356.65	1500
52:11.6	1208	88	22.76	0.61	-5.27	1.56	356.37	1492
52:11.7	1208	90	22.76	0.61	-1.09	-1.45	355.42	1496

Sheet1 | Data In | Data Out | Settings

Εικόνα 150. Telemetry Receiver, Data Streamer Layout

Η εικόνα 150 απεικονίζει το περιβάλλον του Data Streamer του Excel. Τα δεδομένα που αποθηκεύονται μπορούν να επεξεργαστούν χρησιμοποιώντας όλες τις δυνατότητες του Microsoft Excel .

Για την περίπτωση που δεν υπάρχει ασύρματο δίκτυο η παρακάτω οθόνη εμφανίζεται

Εικόνα 151. Telemetry Receiver, Link lost

Εικόνα 152. Telemetry Receiver

3.17 Το πρόγραμμα του Telemetry Receiver

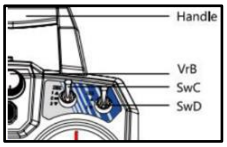
```
#include <SPI.h>

#include <nRF24L01.h>

#include <RF24.h>

#include <LiquidCrystal_I2C.h>

LiquidCrystal_I2C lcd(0x27, 16, 2);

RF24 radio(8, 9); // CE, CSN

const byte address[6] = "00001";

struct Data_Package {

      float  heading_tx, altitude_tx, temp_tx, throttle_tx, command_tx, percentage_tx,pitch_tx, roll_tx ;

                };

      Data_Package data;

unsigned long timer ;

int switch_state;

void setup() {

  Serial.begin(250000);

  pinMode(4,INPUT_PULLUP);

   lcd.init(); // initialize the lcd

  lcd.backlight();

  lcd.clear();

  lcd.setCursor(0, 0);

  lcd.print("Telemetry Rcvr");

  lcd.setCursor(0, 1);

  lcd.print("Station  PG.V1.4");

  delay(3000);

  radio.begin();

  radio.openReadingPipe(0, address);

  radio.setChannel(100);

  radio.setPALevel(RF24_PA_MAX);

  radio.setDataRate(RF24_250KBPS);

   radio.startListening();

  timer = micros();

}
```

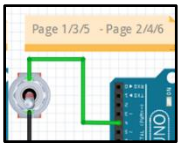
Αρχικά δηλώνονται οι βιβλιοθήκες που πρόκειται να χρησιμοποιηθούν και στην συνέχεια δημιουργούνται τα objects των LCD και nRF24, δηλώνεται η pipe address του nRF24 και σχηματίζεται το data package όμοιο με αυτό του Secondary MCU. Ακολουθούν οι δηλώσεις των μεταβλητών timer και switch_state. Στην void setup αρχικοποιείται η σειριακή επικοινωνία σε συγκεκριμένο baud rate και τίθεται το digital pin 4 σε κατάσταση εισόδου pullup αφού θα σκανδαλιστεί με λογικό ΄΄0΄΄. Ακολουθεί η αρχικοποίηση της LCD οθόνης και ένα μήνυμα εισαγωγής διάρκειας 3 δευτερολέπτων. Τέλος οι ρυθμίσεις του nRF24 όμοιες με αυτές του Secondary MCU, κανάλι 100, power level Max, baud rate 250 KBPS, κατάσταση δέκτη. Στην μεταβλητή timer η συνάρτηση micros.

Εικόνα 153. Telemetry Receiver , επεξήγηση κώδικα μέρος α΄

```
void loop() {

  if (radio.available()) {

  radio.read(&data, sizeof(Data_Package));

  if (data.throttle_tx <1000 ) data.throttle_tx = 1000;

  switch_state = digitalRead(4);

    if  (data.command_tx >1400 && data.command_tx <1600 && switch_state == LOW) {

        lcd.clear();

        lcd.setCursor(1, 0);

        lcd.print("Data Streaming");

        Serial.print(data.throttle_tx,0);

        Serial.print(",");

        Serial.print(data.percentage_tx,0);

        Serial.print(",");

        Serial.print(data.temp_tx);

        Serial.print(",");

        Serial.print(data.altitude_tx);

        Serial.print(",");

        Serial.print(data.pitch_tx);

        Serial.print(",");

        Serial.print(data.roll_tx);

        Serial.print(",");

        Serial.print(data.heading_tx);

        Serial.print(",");

        Serial.print(data.command_tx);

        Serial.print(",");

        Serial.println();

        delay(100);

        }
```

Στην void loop, εάν υπάρχει ζεύξη διάβασε το πακέτο Data_Package. Το throttle δεν θέλουμε να δείχνει τιμές κάτω από 1000 μsec. Στη συνέχεια διάβασε την κατάσταση του διακόπτη στην ψηφιακή είσοδο 4. Ακολουθούν όλοι οι λειτουργικοί συνδυασμοί μεταξύ των μεταβλητών data.command_tx και switch_state , ή πράξη είναι λογικό ΄΄και΄΄ , δηλαδή θα πρέπει να ισχύουν και οι δύο συνθήκες για να εκτελεστεί η ΄΄if΄΄ .Έχει προαναφερθεί ότι η εντολή command στέλνει ένα PWM με τιμές 1000-1500-2000 μsec ανάλογα με την θέση του διακόπτη 3 θέσεων του control panel του τηλεχειρισμού. Κατά αυτό τον τρόπο όταν ο διακόπτης command είναι στην μεσαία θέση (1500 SwC του τηλεχειρισμού)και ο διακόπτης switch_state είναι σε κατάσταση LOW ,





ενεργοποίησε το Data Streaming και στείλε όλα τα δεδομένα στο Excel . Στην οθόνη τύπωσε το μήνυμα ΄΄ Data Streaming ΄΄ . Το delay είναι απαραίτητο για να παρέχει τον απαιτούμενο χρόνο να απεικονιστούν τα δεδομένα στο Excel.

Εικόνα 154. Telemetry Receiver , επεξήγηση κώδικα μέρος β΄

```
else if   (data.command_tx <1200 && switch_state == LOW) {
                lcd.clear();
                lcd.setCursor(0,0);
                lcd.print("T=");
                lcd.print(data.throttle_tx,0);
                lcd.setCursor(9,0);
                lcd.print("V(%)");
                lcd.print(data.percentage_tx,0);
                lcd.setCursor(0,1);
                lcd.print("Tmp:");
                lcd.print(data.temp_tx,1);
                lcd.setCursor(9,1);
                lcd.print("A:");
                lcd.print(data.altitude_tx,1);
                delay(200);   }
else if   (data.command_tx <1200 && switch_state == HIGH){
                lcd.clear();
                lcd.setCursor(0,0);
                lcd.print("H(D):");
                lcd.print(data.heading_tx,0);
                lcd.setCursor(10,0);
                lcd.print("C:");
                lcd.print(data.command_tx);
                lcd.setCursor(0,1);
                lcd.print("P:");
                lcd.print(data.pitch_tx,1);
                lcd.setCursor(9,1);
                lcd.print("R:");
                lcd.print(data.roll_tx,1);
                delay(200);   }
      else {
            lcd.clear();
            lcd.setCursor(1, 0);
            lcd.print("***Reserved***");
            lcd.setCursor(0, 1);
            lcd.print("*for future use*");
            delay(200);    }
 }
```

Για την περίπτωση που η command είναι 1000 μsec (<1200) και ο διακόπτης switch_state είναι LOW τότε εκτύπωσε στην οθόνη τα πρώτα 4 δεδομένα : throttle, Voltage percentage, Temperature, Altitude.

Συνεχίζοντας με ίδια τη συνθήκη της command και το switch_state να είναι HIGH τότε τύπωσε τα υπόλοιπα 4 δεδομένα : Heading, command, pitch, roll.

Σε κάθε άλλη περίπτωση τύπωσε το μήνυμα ΄΄ Reserved for future use ΄΄ .

Εικόνα 155. Telemetry Receiver, επεξήγηση κώδικα μέρος γ΄

```
else {
   lcd.clear();
   lcd.setCursor(1, 0);
   lcd.print("***No Data***");
   lcd.setCursor(0, 1);
   lcd.print("***Check Link***");
   delay(200);
 }
}
                          //Copy – Paste Compilation Verified
```

Τελειώνοντας, η else έρχεται να συμπληρώσει την περίπτωση που δεν υπάρχει ασύρματο δίκτυο τυπώνοντας το μήνυμα ΄΄ No Data - Check Link΄΄

Εικόνα 156. Telemetry Receiver, επεξήγηση κώδικα μέρος δ΄

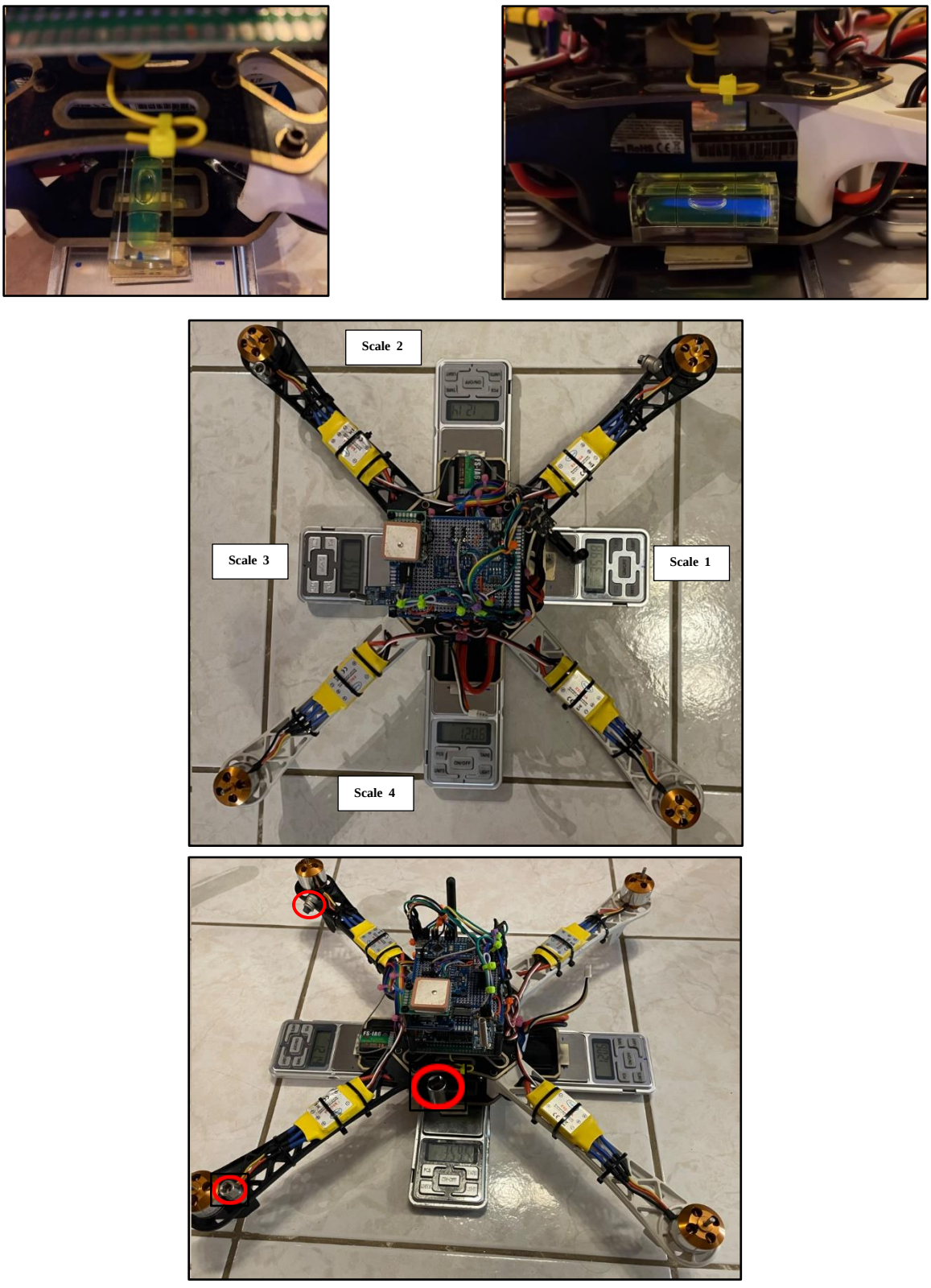
3.18 Η διαδικασία Weight and Balance και Prop Balancing

Η διαδικασία του Weight and Balance και του Propeller Balancing είναι η τελευταία της ολοκλήρωσης του τετρακόπτερου. Η σημασία τους έχει εξηγηθεί σε προηγούμενη ενότητα. Σε αυτή την παράγραφο θα αναλύσουμε τον τρόπο που θα επιτευχθεί.

Για την πραγματοποίηση της ζύγισης και ευθυγράμμισης του τετρακόπτερου θα χρησιμοποιήσουμε 4 ζυγαριές ακριβείας , μέγιστου βάρος 500 γρ. και ακρίβειας 0.1 γρ. πάνω στις οποίες θα τοποθετήσουμε το τετρακόπτερο. Με την βοήθεια διάφορων υλικών spare parts και την χρήση μηχανικού κλισιμέτρου (bubble level) θα ευθυγραμμίσουμε το τετρακόπτερο στις δύο διαστάσεις ΄΄pitch΄΄ και ΄΄roll΄΄ . Η συμμετρία των ζυγαριών όσο αφορά τα σημεία στήριξης όπως και η ευθυγράμμιση του τετρακόπτερου θα διαδραματίσει σημαντικό ρόλο στην εξαγωγή των αποτελεσμάτων . Οι δύο ζυγαριές ανά άξονα θα μας δώσουν την κατανομή του βάρους όπως επίσης και το συνολικό βάρος σε αυτόν τον άξονα . Ομοίως και για το άλλο άξονα . Με την βοήθεια spare parts σε ρόλο αντίβαρων θα επιτύχουμε την συμμετρία ανά άξονα . Το σημείο τομής των δύο αξόνων είναι το κέντρο βάρους του τετρακόπτερου για τις δύο διαστάσεις και το σύνολο των βαρών από τις ζυγαριές αποτελεί το συνολικό βάρος του τετρακόπτερου.

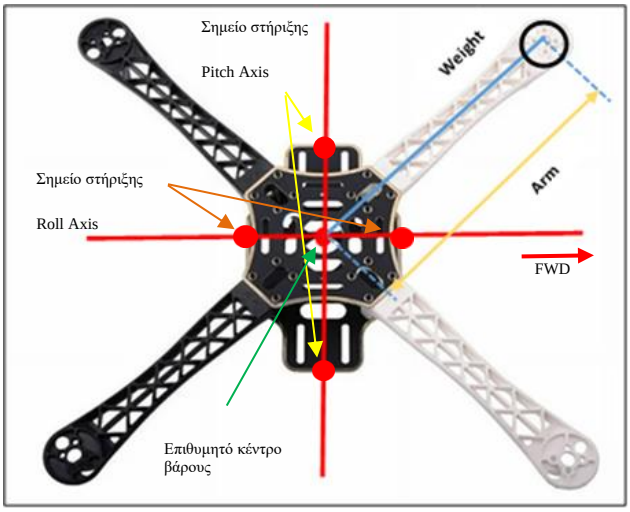
Εικόνα 157. W&B, η ζυγαριά, το κλισίμετρο και τα αντίβαρα

Στις εικόνες που ακολουθούν φαίνεται ο τρόπος εξαγωγής των αποτελεσμάτων της μέτρησης. Το τετρακόπτερο γίνεται level και στις δύο κατευθύνσεις ΄΄ pitch΄΄ και ΄΄roll΄΄ ενώ με την προσθήκη αντίβαρων επιτυγχάνουμε την ζύγιση του τετρακόπτερου. Τα αντίβαρα τελικά θα τοποθετηθούν μόνιμα κάτω από τα μοτέρ ώστε να μεγαλώσει το ΄΄ arm΄΄ και να επιτύχουμε το επιθυμητό ΄΄moment΄΄ με το ελάχιστο δυνατό ΄΄ weight΄΄. Το άθροισμα των τιμών των τεσσάρων ζυγαριών και των τεσσάρων προπελών θα εξάγουν το τελικό βάρος του τετρακόπτερου.



Εικόνα 158. W&B, η ζύγιση και ισορρόπηση του τετρακόπτερου

Εικόνα 159. W&B, η ζύγιση των προπελών



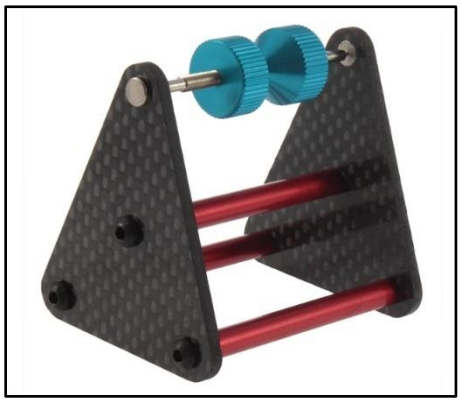
Εικόνα 160. W&B, τα σημεία στήριξης και το επιθυμητό κέντρο βάρους

Axis	Scale No	Weight(gr)
Roll Axis	Scale 1	359.8
	Scale 3	359.6
Pitch Axis	Scale 2	121.4
	Scale 4	120.6
Props(4x)	Scale 5	54
Total		1015.4

Εικόνα 161. W&B, ανάλυση βαρών τετρακόπτερου

Για την ισορρόπηση των προπελών (Propeller Balancing) χρησιμοποιείται ο Propeller Balancer της εικόνας 162. Η προπέλα εφαρμόζει σε άξονα ο οποίος μαγνητικά στέκεται στον Balancer. Η λογική είναι η προπέλα να ισορροπήσει και τα σημεία συμμετρικά του άξονα περιστροφής να ισαπέχουν από σταθερό σημείο. Υπάρχουν πολλοί τρόποι να επιτευχθεί αυτό. Είτε

προστίθεται βάρος στο ελαφρύ μέρος της προπέλας (ταινία , κόλλα, μπογιά) είτε αφαιρείται υλικό από το βαρύ μέρος της προπέλας (υαλόχαρτο). Το αποτέλεσμα μιας ισορροπημένης προπέλας φαίνεται στην εικόνα 163 .

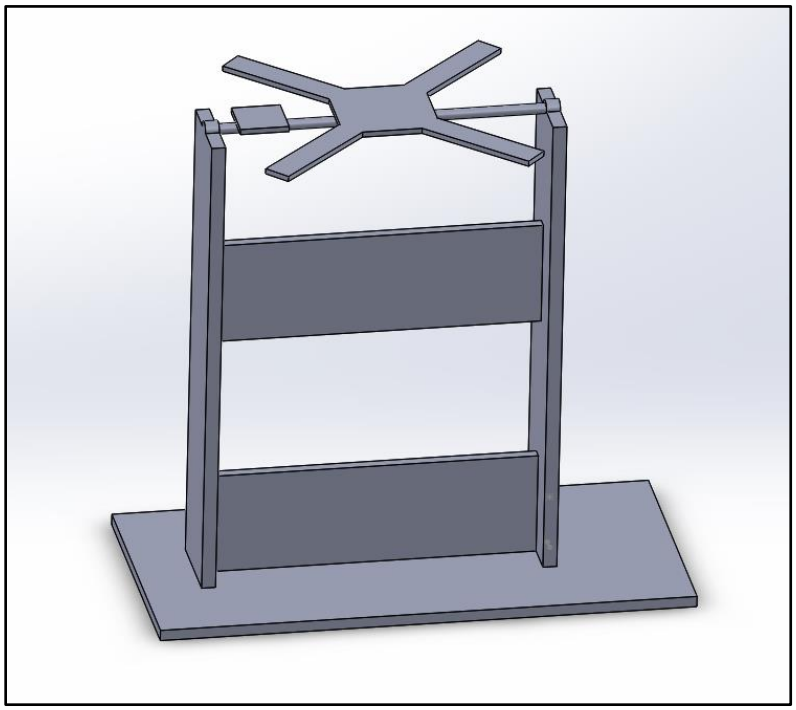


Εικόνα 162. Ο Propeller Balancer

Εικόνα 163. Ισορροπημένη προπέλα

4 Σταθμός ελέγχου και ρύθμισης Quadcopter

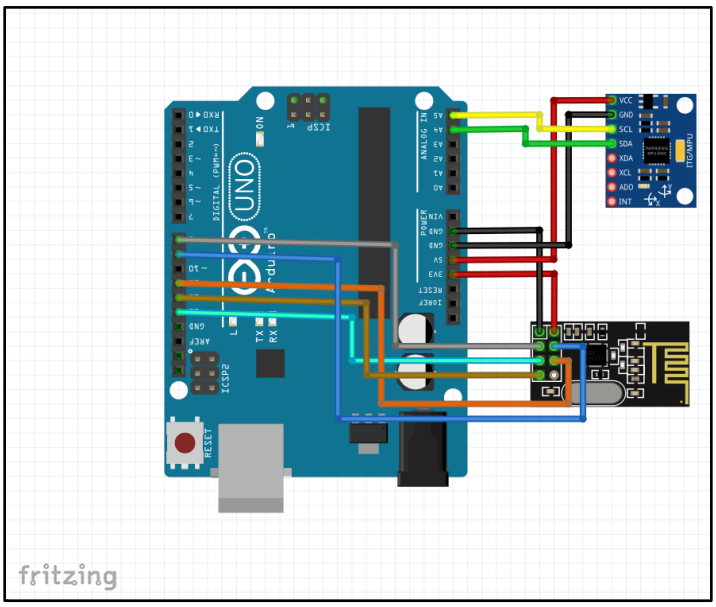
Η ανάγκη ελέγχου και ρύθμισης του Quadcopter σε όσο το δυνατό πιο πραγματικές συνθήκες , οδήγησε στο σχεδιασμό και την κατασκευή ενός σταθμού ελέγχου. Το ζητούμενο είναι να ελεγχθεί η αξιοπιστία του προγράμματος, η λειτουργία του συνόλου της κατασκευής του Quadcopter και να ρυθμιστεί το PID, με γνώμονα πάντα την ασφάλεια του χειριστή και του εξοπλισμού. Στην εικόνα 164 φαίνεται ο σχεδιασμός του σταθμού στο πρόγραμμα SolidWorks.



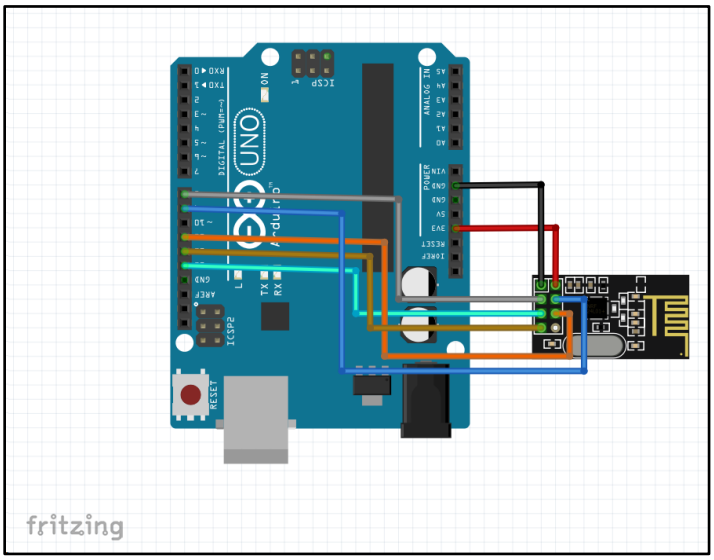
Εικόνα 164. Ο σταθμός ελέγχου στο SolidWorks

Ο σταθμός ελέγχου έχει κατασκευαστεί από ξύλο MDF πάχους 18 mm και 16 mm. Σωλήνας αλουμινίου 15 mm, υλικού 6063 εδράζεται σε δύο bearing κλειστού τύπου τα οποία έχουν ασφαλιστεί στο ξύλινο πλαίσιο. Πάνω στον σωλήνα έχει εφαρμοστεί η ξύλινη βάση που θα φιλοξενήσει το Quadcopter το οποίο δένεται με δεματικά. Ένα δεύτερο μικρό ξύλινο πλαίσιο, πλήρως ευθυγραμμισμένο με το βασικό θα φιλοξενεί ένα IMU (Inertial Measurement Unit) το οποίο σε πραγματικό χρόνο, με την χρήση ενός Arduino Uno, θα στέλνει στον υπολογιστή, ασύρματα τις κλίσεις του τετρακοπτέρου. Η βάση πρέπει να είναι στιβαρή αφού θα αναπτυχθούν δυνάμεις μέχρι και δύο Kg, επίσης πρέπει να έχει βάρος τόσο ώστε να μην εισάγει θόρυβο από τις ταλαντώσεις και κινήσεις κατά την διάρκεια των ελέγχων στο IMU και να έχει όσο το δυνατό λιγότερες τριβές κατά την περιστροφή στον άξονα ελέγχου ώστε να πλησιάσει περισσότερο τις πραγματικές συνθήκες. Ο άξονας περιστροφής δίνει πληροφορίες μόνο σε μία διάσταση στο χώρο. Περιστρέφοντας το Quadcopter κατά 90^{o} πάνω στην βάση θα το ελέγξουμε και στην άλλη διάσταση (Pitch – Roll). Η τροφοδοσία του κυκλώματος επιτυγχάνεται είτε με 5V DC από USB, είτε από μπαταρία 12,6V DC.

Στην εικόνα 165 φαίνεται το ηλεκτρονικό κύκλωμα που συνοδεύει την κατασκευή. Διακρίνεται ο μικροελεγκτής Arduino Uno, το MPU6050 σε ρόλο IMU, το nRF24L01 για την ασύρματη μετάδοση των δεδομένων και οι απαραίτητες συνδέσεις. Το πρόγραμμα του μικροελεγκτή και ο τρόπος λειτουργίας του θα εξηγηθεί στις επόμενες γραμμές.



Εικόνα 165. Το ηλεκτρονικό κύκλωμα του πομπού του σταθμού ελέγχου



Εικόνα 166. Το ηλεκτρονικό κύκλωμα του δέκτη του σταθμού ελέγχου

4.1 Το πρόγραμμα του σταθμού ελέγχου (Transmitter)

Το πρόγραμμα της βάσης ελέγχου είναι σχετικά απλό και θα εξηγηθεί στις επόμενες γραμμές. Το κύκλωμα όπως φαίνεται στην εικόνα 165 αποτελείται από ένα MPU-6050 και ένα nRF24L01 σε συνδεσμολογία IIC και SPI αντίστοιχα.

```
#include <SPI.h>
#include <nRF24L01.h>
#include <RF24.h>
//.................RF24 Object & Data_Package.....................//
RF24 radio(8, 9); // CE, CSN                                       //
const byte address[6] = "00001";                                   //
  struct Data_Package {                                            //
    float roll_tx, zero_timer_tx ; };   //2X4Bytes  = 8/32 max   //
    Data_Package data;                                            //
//.....................................................................//
//Batch1.....................DMP implementation.....................//
#include "MPU6050_6Axis_MotionApps612.h"                           //
#include "Wire.h"                                                  //
#include "I2Cdev.h"                                                //
MPU6050 mpu;                                                       //
bool dmpReady = false;  // set true if DMP init was successful     //
uint8_t mpuIntStatus;   // holds actual interrupt status byte from MPU          //
uint8_t devStatus;      // return status after each device operation           //
uint16_t packetSize;    // expected DMP packet size (default is 42 bytes)      //
uint16_t fifoCount;     // count of all bytes currently in FIFO                //
uint8_t fifoBuffer[64]; // FIFO storage buffer                                 //
Quaternion q;           // [w, x, y, z]         quaternion container           //
VectorFloat gravity;    // [x, y, z]            gravity vector                 //
float ypr[3], yaw, pitch, roll;                                    //
//.....................................................................Batch1//
//.............................General Variables Declaration.....................//
unsigned long  zero_timer ;                                        //
//.....................................................................//
```

Αρχικά δηλώνονται οι βιβλιοθήκες SPI, nRF24L01, RF24, δημιουργείται το object ΄΄radio΄΄ και δημιουργείται ένα πακέτο από δύο μεταβλητές την ΄΄roll_tx΄΄ και την ΄΄zero_timer_tx΄΄, είναι οι μεταβλητές που θα εκπέμπονται από το nRF, έχουν συνολικό μέγεθος 8 bytes(μέγιστο 32 bytes). Ακολουθεί το πακέτο του DMP. Αυτό έχει παρουσιαστεί ξανά σε προηγούμενη ενότητα. Μας ενδιαφέρει μόνο η μεταβλητή ΄΄roll΄΄. Τέλος δηλώνεται η μεταβλητή ΄΄zero timer΄΄ για να μετρά τον χρόνο παράλληλα με την τιμή της γωνίας ΄΄roll΄΄ . Έτσι έχει διαμορφωθεί ένα πακέτο δεδομένων, χρόνου και γωνίας το οποίο τελικά θα εκπέμπεται.

Εικόνα 167. Το πρόγραμμα του σταθμού ελέγχου, μέρος α΄

```
void setup() {

 //Batch2.....DMP implementation.....//

        Wire.begin();                                          //

        Wire.setClock(400000);                                 //

        mpu.initialize();                                      //

        devStatus = mpu.dmpInitialize();                       //

        mpu.setXGyroOffset(116);                               //

        mpu.setYGyroOffset(11);                                //

        mpu.setZGyroOffset(27);                                //

        mpu.setXAccelOffset(-715);                             //

        mpu.setYAccelOffset(1219);                             //

        mpu.setZAccelOffset(1257);                             //

        mpu.CalibrateAccel(6);                                 //

        mpu.CalibrateGyro(6);                                  //

        mpu.PrintActiveOffsets();                              //

        mpu.setDMPEnabled(true);                               //

        dmpReady = true;                                       //

        packetSize = mpu.dmpGetFIFOPacketSize();     //

 //.....Batch2//

 //.....nRF  Setup.....//

 radio.begin();                                                //

 radio.openWritingPipe(address);                               //

 radio.setChannel(5);                                          //

 radio.setPALevel(RF24_PA_MIN);                                //

 radio.setDataRate(RF24_250KBPS);                              //

 radio.setPayloadSize(8);                                      //

 radio.stopListening();                                        //

 //.....//

  }
```

Στο δεύτερο μέρος του προγράμματος στο void setup γίνεται η αρχικοποίηση του MPU-6050 και δηλώνονται τα offset του, όπως έχει προαναφερθεί σε παραπάνω ενότητες. Ακολουθεί η αρχικοποίηση του nRF και ξεκινά το object ΄΄radio΄΄, εγκαθίσταται ο δίαυλος επικοινωνίας στην διεύθυνση ΄΄00001΄΄, ορίζεται η συχνότητα του καναλιού επικοινωνίας - 2400 MHz +5 MHz = 2405 MHz και η στάθμη του επιπέδου εκπομπής σε MIN που αντιστοιχεί σε -18dbm, στην συνέχεια το data rate στα 250 Kbps, το payload που είναι 8 bytes και τελικά το nRF τίθεται ως πομπός .

Εικόνα 168. Το πρόγραμμα του σταθμού ελέγχου, μέρος β΄

```
void loop() {

  zero_timer = micros();

//Batch3.....DMP implementation.....//

 if (mpu.dmpGetCurrentFIFOPacket(fifoBuffer)) {          //

     mpu.dmpGetQuaternion(&q, fifoBuffer);                //

     mpu.dmpGetGravity(&gravity, &q);                     //

     mpu.dmpGetYawPitchRoll(ypr, &q, &gravity);           //

     //yaw = ypr[0] * (180 / M_PI);                       //

     //pitch = ypr[1] * (180 / M_PI) ;                    //

     roll = ypr[2] * (180 / M_PI);                        //

     }                                                    //

//……….....Batch3//

//.....Data Structure to transmit .....//

data.roll_tx = roll ;                                     //

data.zero_timer_tx = zero_timer ;                         //

radio.write(&data, sizeof(Data_Package));                 //

//.....//

}

                                  //Copy – Paste Compilation Verified
```

Στην void loop την ρουτίνα που επαναλαμβάνεται συνεχώς, από το Quaternion παίρνουμε τελικά μόνο την γωνία ΄΄roll΄΄ . Το μικρό ξύλινο πλαίσιο που έχει εγκατασταθεί παράλληλα με το πλαίσιο στήριξης του τετρακόπτερου και φιλοξενεί το MPU-6050 θα δίνει πάντα την γωνία ΄΄roll΄΄, αυτή θα μεταφράζεται σε ΄΄pitch΄΄ ή ΄΄roll΄΄ του τετρακόπτερου ανάλογα με το τρόπο στήριξης του τετρακόπτερου στην βάση .

Τέλος διαμορφώνεται το πακέτο που θα εκπεμφθεί μεταβιβάζοντας στις υπό εκπομπή μεταβλητές τις τιμές των μεταβλητών της γωνίας ΄΄ roll΄΄ και του χρόνου ΄΄ zero_timer΄΄ .

Εικόνα 169. Το πρόγραμμα του σταθμού ελέγχου, μέρος γ΄

4.2 Το πρόγραμμα του σταθμού ελέγχου (Receiver)

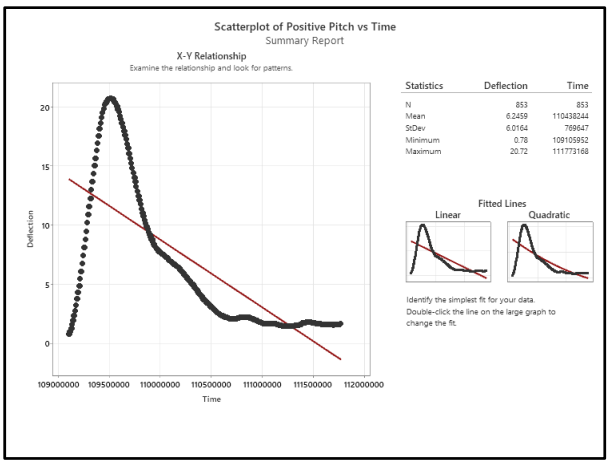
```
#include <SPI.h>
#include <nRF24L01.h>
#include <RF24.h>
//.............RF24 Object & Data_Package.....................//
 RF24 radio(8, 9); // CE, CSN                                  //
 const byte address[6] = "00001";                              //
 struct Data_Package {                                         //
 float roll_tx, zero_timer_tx ;       };                       //
 Data_Package data;                                            //
//.....................................................................//
void setup() {
 Serial.begin(250000);
 //.....................................nRF  Setup.....................//
 radio.begin();                                                //
 radio.openReadingPipe(0, address);                            //
 radio.setChannel(5);                                          //
  radio.setPALevel(RF24_PA_MIN);                               //
 radio.setDataRate(RF24_250KBPS);                              //
 radio.setPayloadSize(8);                                      //
 radio.startListening();                                       //
//.....................................................................//
}
void loop() {
   if (radio.available()) {
    radio.read(&data, sizeof(Data_Package));
   Serial.print(data.zero_timer_tx);
   Serial.print(",");
   Serial.print(data.roll_tx);
   Serial.print(",");
  Serial.println();
 }
 }
                                        //Copy – Paste Compilation Verified
```

Κατά αντιστοιχία με το πρόγραμμα του πομπού αρχικά δηλώνονται οι βιβλιοθήκες που θα χρησιμοποιηθούν και στη συνέχεια ορίζεται το object ΄΄ radio΄΄ . Δημιουργείται το πακέτο ΄΄ data΄΄ που περιέχει τις μεταβλητές που εκπέμπονται . Στο void setup ορίζεται η σειριακή επικοινωνία σε συγκεκριμένο baud rate και αρχικοποιείται το nRF όπως και στην περίπτωση της εκπομπής με μόνη διαφορά ότι τώρα τίθεται σε κατάσταση δέκτη ΄΄ radio.startListening(); ΄΄ .

Στην void loop , αν υπάρχει δίκτυο, το πρόγραμμα διαβάζει το πακέτο και από εκεί εξάγει τα δεδομένα για τις μεταβλητές ΄΄ zero_timer_tx ΄΄ και ΄΄ data_roll_tx ΄΄. Τέλος θα τις τυπώσει στο Serial Monitor.

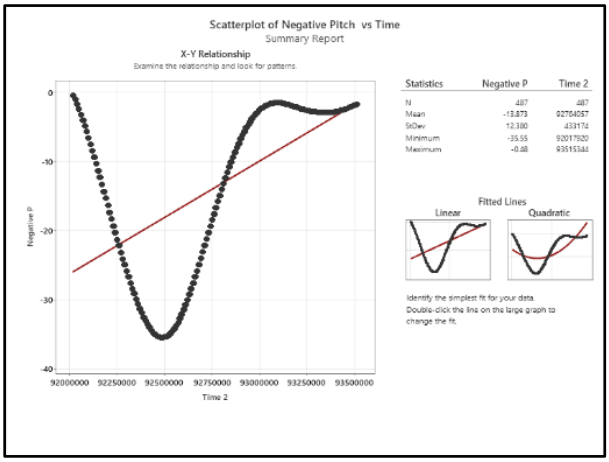
Από το Serial Monitor παίρνουμε τα ζευγάρια των τιμών χρόνου και γωνίας και τις εισάγουμε στο Minitab από όπου προκύπτουν γραφήματα που απεικονίζουν την αντίδραση του τετρακόπτερου σε συγκεκριμένα σενάρια ελεγχόμενης ή βίαιης απόκλισης του τετρακόπτερου. Τέτοια παραδείγματα φαίνονται στις εικόνες 171,172 και 173.

Εικόνα 170. Το πρόγραμμα του σταθμού ελέγχου – receiver



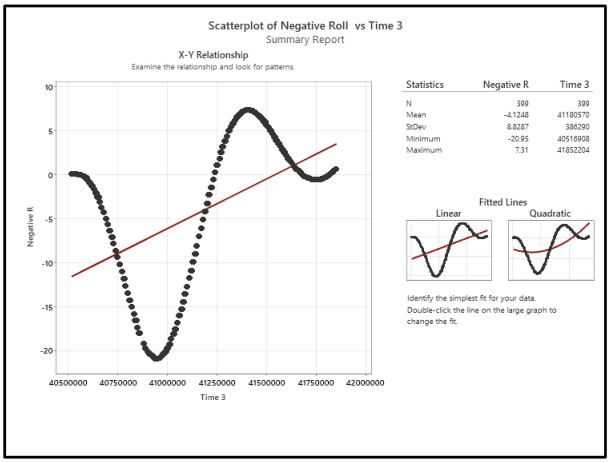
Εικόνα 171. Αντίδραση τετρακόπτερου Positive Pitch vs Time

Μέγιστη απόκλιση 20.72 μοίρες. Πλήθος ζευγαριών απόκλισης – χρόνου, 853 σε συνολικό χρόνο περίπου 3 δευτερολέπτων. Το setpoint είναι 0 μοίρες.



Εικόνα 172. Αντίδραση τετρακόπτερου Negative Pitch vs Time

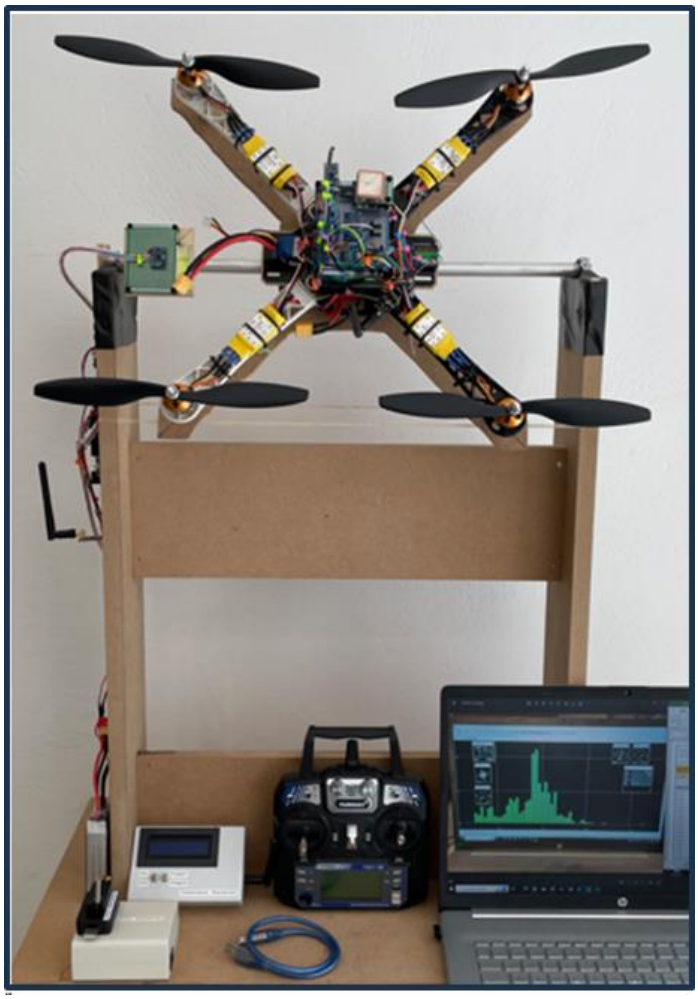
Μέγιστη απόκλιση 35.55 μοίρες. Πλήθος ζευγαριών απόκλισης – χρόνου, 487 σε συνολικό χρόνο περίπου 1.5 δευτερολέπτων. Το setpoint είναι 0 μοίρες



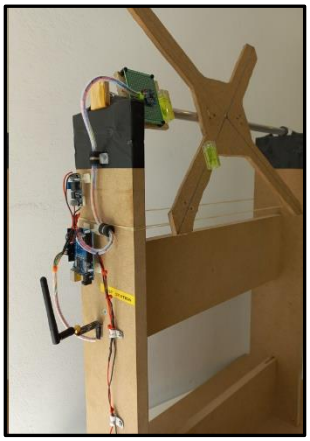
Εικόνα 173. Αντίδραση τετρακόπτερου Negative Roll vs Time

Μέγιστη απόκλιση 20.95 μοίρες. Πλήθος ζευγαριών απόκλισης – χρόνου, 399 σε συνολικό χρόνο περίπου 1.5 δευτερολέπτων. Το setpoint είναι 0 μοίρες

Σε επόμενη ενότητα, με την χρήση του Add -In της Microsoft για το Excel , Data Streamer, θα έχουμε σε πραγματικό χρόνο πληροφορίες για την απόκριση του τετρακόπτερου.



Εικόνα 174. Ο σταθμός ελέγχου

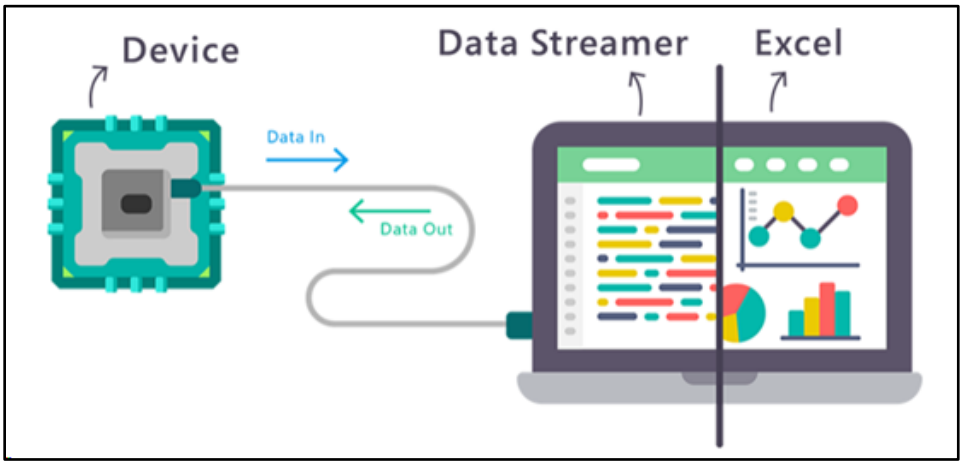


Εικόνα 176. Ο σταθμός ελέγχου, πομπός

Εικόνα 175. Ο σταθμός ελέγχου, δέκτης

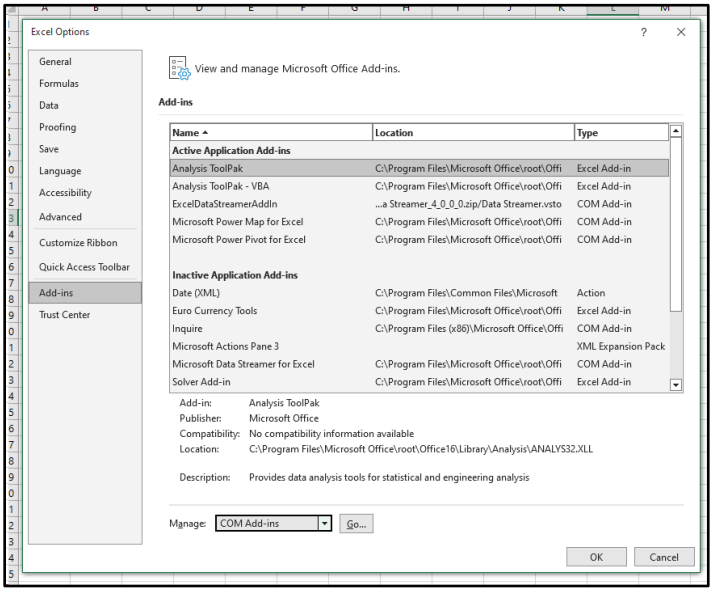
4.3 Excel Data Streamer

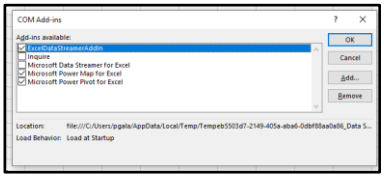
Το Excel Data Streamer είναι ένα Add-In της Microsoft που επιτρέπει στο Excel να επικοινωνεί αμφίδρομα σειριακά με τον μικροελεγκτή. Οι δυνατότητες του Excel στην επεξεργασία και απεικόνιση δεδομένων παρέχουν ένα ιδεατό αποτέλεσμα στην εφαρμογή του ελέγχου της συμπεριφοράς του τετρακόπτερου.



Εικόνα 177. Excel Data Streamer. *https://support.microsoft.com*

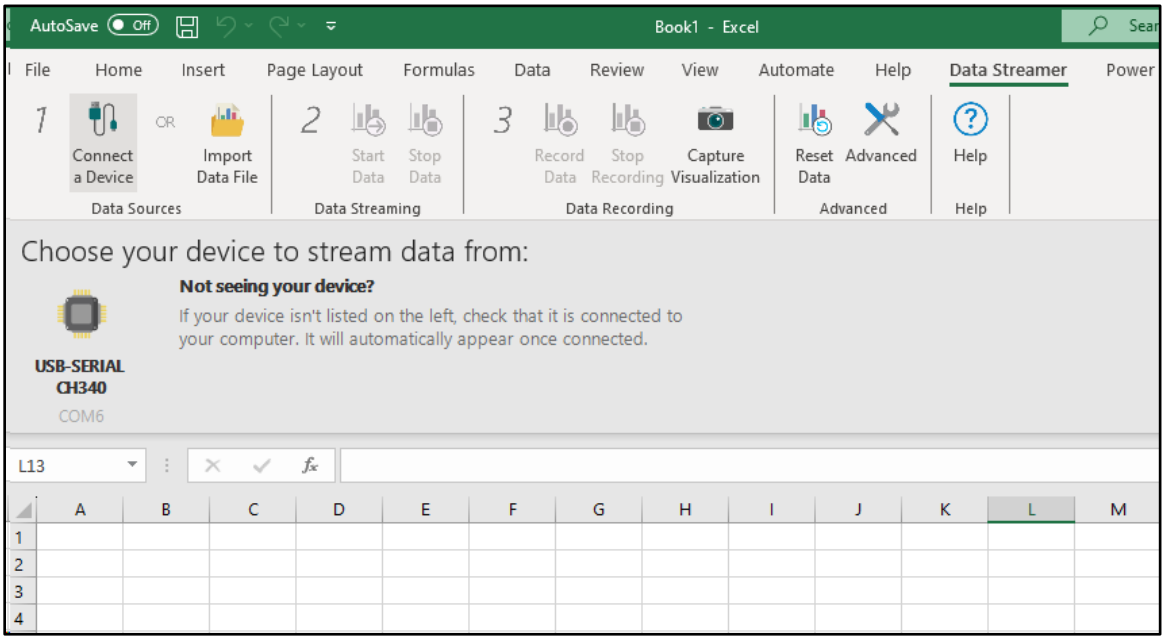
Αρχικά πρέπει να ενεργοποιηθεί το Add- In στο Excel . Για τους κάτοχους του Microsoft 365 το Add-In παρέχεται δωρεάν. Από το menu του Excel : File – Options -Add-ins επιλέγουμε *COM Add-ins* και τσεκάρουμε το *ExcelDataStreamerAddIn* ή το *Microsoft Data Streamer for Excel* :





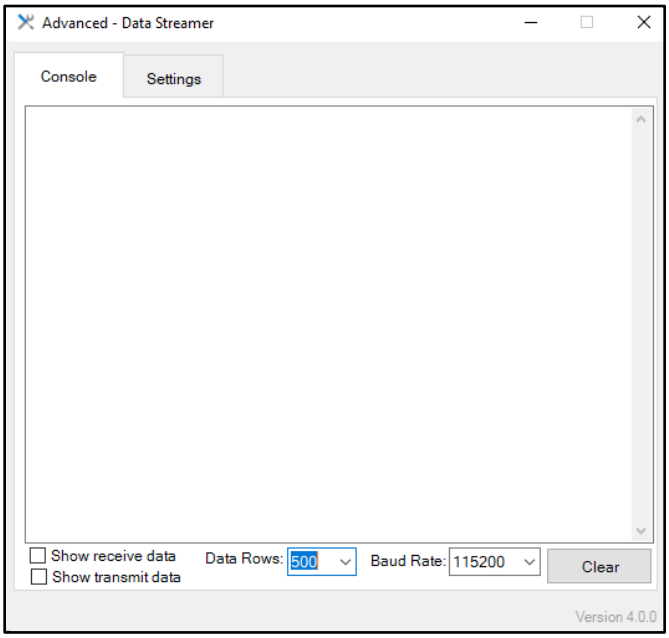
Εικόνα 178. Ενεργοποίηση του Excel Data Streamer Add - In

Στην συνέχεια επιλέγουμε το tab *Data Streamer* και πατάμε *Connect a Device*. Επιλέγουμε τον μικροελεγκτή που έχει συνδεθεί στην USB θύρα.



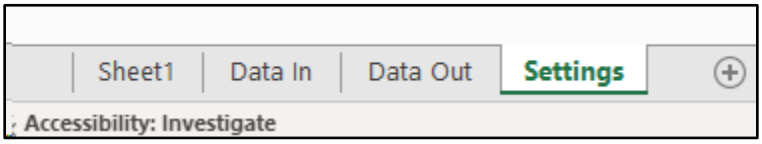
Εικόνα 179. Excel Data Streamer layout

Αφού συνδέθηκε ο μικροελεγκτής, το επόμενο βήμα είναι να πατήσουμε *Advanced* . Το *Baud Rate* πρέπει να συμφωνεί με την ταχύτητα της σειριακής επικοινωνίας του μικροελεγκτή. Στο πεδίο *Data Rows* πρέπει να επιλέξουμε το πλήθος των γραμμών των δεδομένων. Επίσης μπορούμε να επιλέξουμε να φαίνονται στο παράθυρο τα δεδομένα από και προς την σειριακή θύρα.

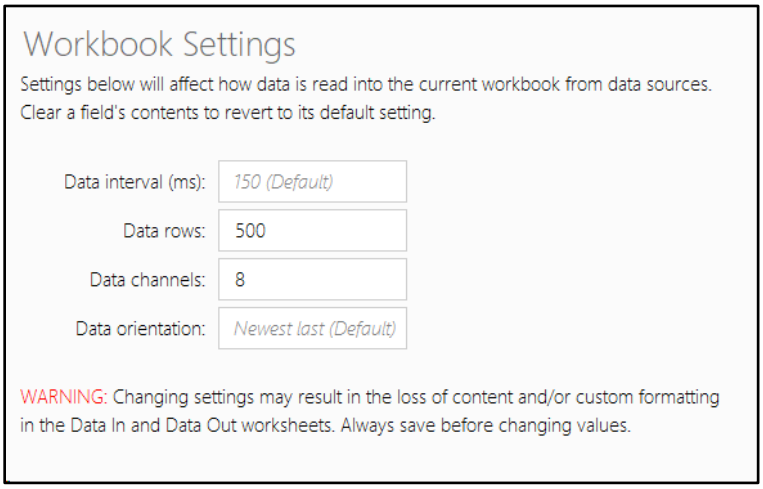


Εικόνα 180. Excel Data Streamer, Advanced Settings

Στο βασικό layout του excel επιλέγουμε tab *Settings* και στο παράθυρο που εμφανίζεται

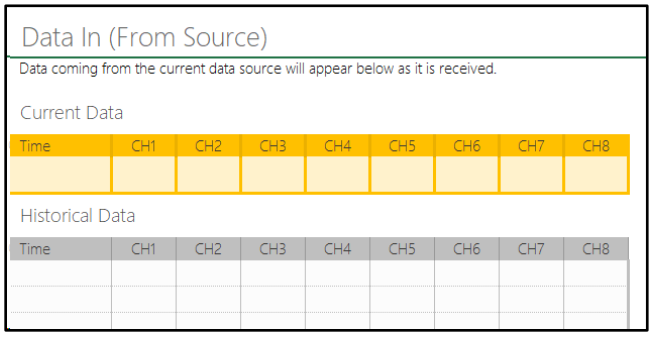


Εικόνα 181. Excel Data Streamer, Tab - Settings

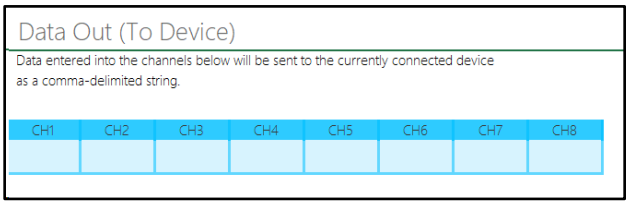


Εικόνα 182. Excel Data Streamer, Workbook Settings

στο πεδίο *Data rows* βάζουμε την ίδια επιλογή με της *Advanced Settings* παραπάνω(Εικόνα 180), ενώ στο πεδίο *Data channels* τον αριθμό των μεταβλητών που θα παρακολουθούμε. Στα υπόλοιπα αφήνουμε τις default τιμές. Στο tab *Data In* παρακολουθούμε τα δεδομένα από τις μεταβλητές ενώ στο tab *Data Out* μπορούμε να στείλουμε data στον μικροελεγκτή .

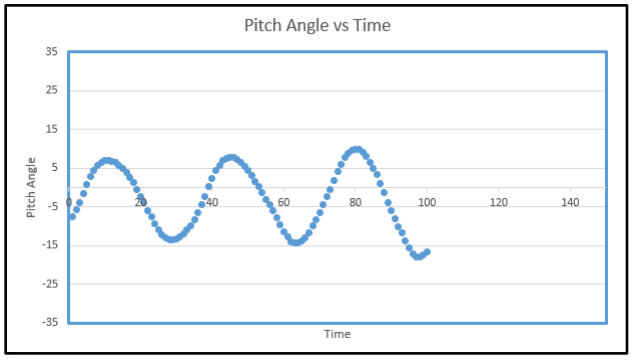


Εικόνα 183. Excel Data Streamer, Data In Tab



Εικόνα 184. Excel Data Streamer, Data Out Tab

Τέλος στο tab *sheet1* μπορούμε να αξιοποιήσουμε όλες τις δυνατότητες του Excel όπως γραφικές παραστάσεις, στατιστικά, πράξεις κ.α. Στην εικόνα 186 που ακολουθεί φαίνονται τα αποτελέσματα σε πραγματικό χρόνο προερχόμενα από τα δεδομένα που εκπέμπονται από το τετρακόπτερο.



Εικόνα 185. Excel Data Streamer, Graph -Pitch Angle vs Time

Data In (From Source)

Data coming from the current data source will appear below as it is received.

Current Data

Time	Throttle	Battery(%)	Temperature(Celcius)	Altitude(m)	Pitch	Roll	Heading	Command
51:08.8	1360	53	22.44	0.29	-6.93	2.53	133.75	1496

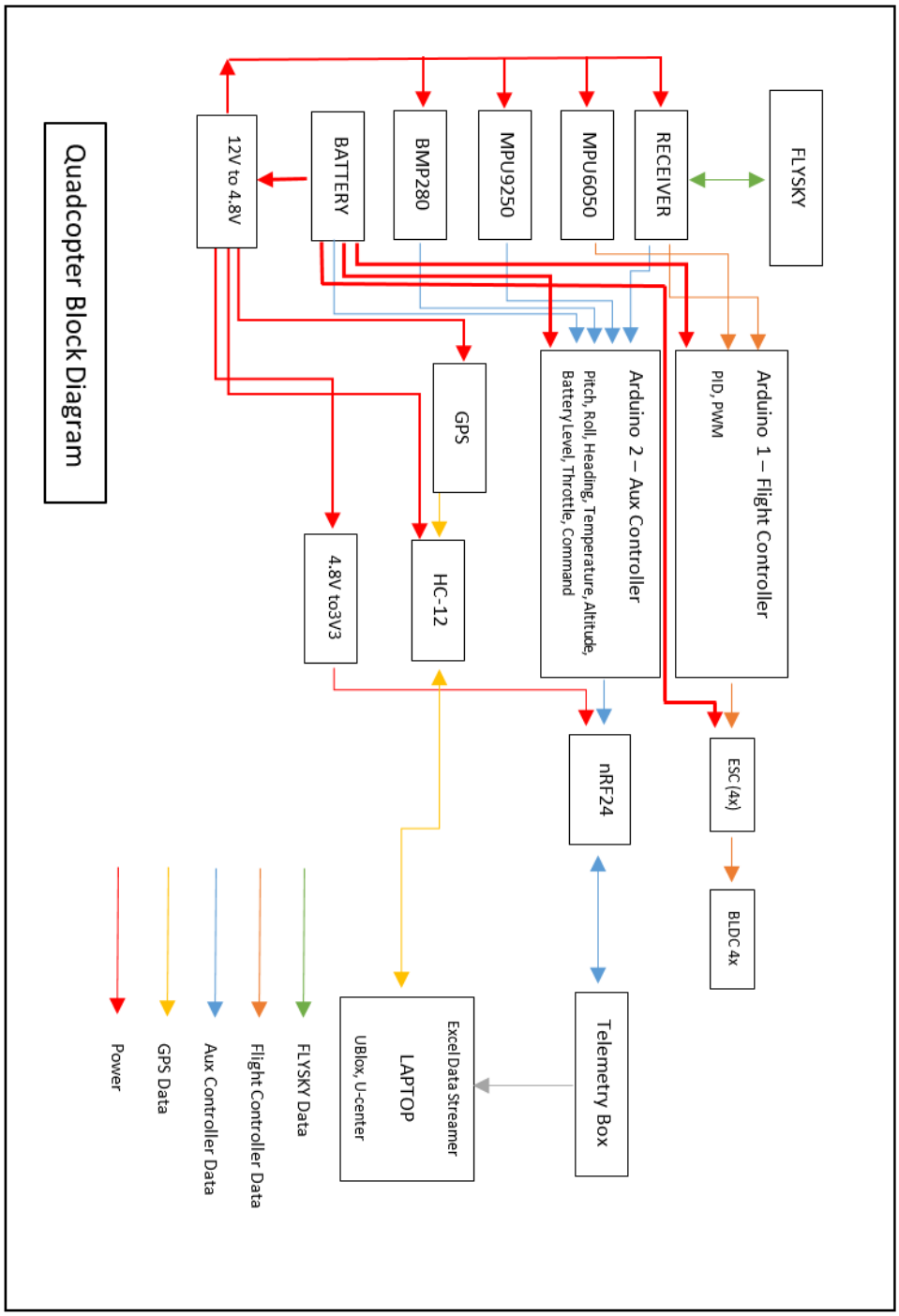
Historical Data

Time	Throttle	Battery(%)	Temperature(Celcius)	Altitude(m)	Pitch	Roll	Heading	Command
50:04.3	1000	67	22.15	0.66	-1.14	2.19	134.34	1496
50:04.5	1000	70	22.15	0.66	-1.38	1.94	134.33	1500
50:04.6	1000	66	22.15	0.69	-1.93	1.51	133.57	1500
50:04.7	1000	70	22.15	0.72	-1.7	1.67	133.66	1500
50:04.9	1000	70	22.15	0.72	-1.59	1.74	133.73	1500
50:05.0	1012	70	22.15	0.63	-1.6	1.79	133.76	1500
50:05.1	1000	70	22.16	0.6	-1.38	1.98	134.23	1500
50:05.3	1000	67	22.16	0.6	-1.15	2.2	134.97	1500
50:05.4	1000	67	22.16	0.57	-1.21	2.13	135.31	1500
50:05.5	1000	70	22.16	0.71	-1.26	2.09	135.62	1508
50:05.6	1000	70	22.16	0.71	-1.29	2.06	135.88	1500
50:05.8	1000	70	22.16	0.77	-1.33	2	136.1	1496
50:05.9	1000	67	22.16	0.71	-1.75	1.63	135.57	1500
50:06.0	1000	70	22.16	0.71	-1.73	1.65	135.47	1500
50:06.2	1000	70	22.16	0.71	-1.73	1.69	135.37	1500
50:06.3	1000	70	22.16	0.66	-1.98	1.5	134.8	1500
50:06.4	1000	67	22.16	0.6	-2.03	1.47	134.37	1496

Εικόνα 186. Excel Data Streamer, Real Time Variables values

5. ΣΧΕΔΙΑΣΗ ΚΑΙ ΚΑΤΑΣΚΕΥΗ ΥΛΙΚΟΥ ΜΕΡΟΥΣ

5.1.1 Block Diagram



Εικόνα 187. Μπλόκ Διάγραμμα Quadcopter

5.1.2 Συνοπτική Επεξήγηση Μπλοκ Διαγράμματος

Τροφοδοσία.

Η βασική πηγή τροφοδοσίας είναι μια 3 Cell Lipo μπαταρία χωρητικότητας 2600mAh. Η μπαταρία αυτή τροφοδοτεί με 12V τους δύο μικροελεγκτές και τα ESC's . Παράλληλα τροφοδοτεί τον πρώτο step down converter ο οποίος υποβαθμίζει τα 12V σε 4V8. Στην συνέχεια αυτή η τάση παρέχεται στα : Rx/Tx Receiver, MPU6050, MPU9250, BMP280, GPS, HC-12 και τέλος στον δεύτερο step down converter που υποβαθμίζει εκ νέου την τάση τροφοδοσίας στα 3V3 . Αυτή είναι η τάση που τροφοδοτεί το nRf24.

Flight Controller.

Ο Flight Controller (Main MCU) έχει εισόδους δεδομένα που προέρχονται από τον δέκτη του τηλεχειρισμού και το βασικό αδρανειακό σύστημα μέτρησης MPU6050. Οι έξοδοι του παρέχουν στους ESC's τα απαιτούμενα σήματα για να οδηγήσουν στη σειρά τα BLDC μοτέρ.

Auxiliary Controller.

Ο Auxiliary Controller (Secondary MCU) έχει εισόδους από τον δέκτη του τηλεχειρισμού, το MPU9250, το BMP280 και την μπαταρία ώστε να μορφοποιήσει το προς εκπομπή πακέτο με τα δεδομένα τηλεμετρίας. Το πακέτο αυτό μέσω του nRf24 εκπέμπεται ασύρματα προς τον δέκτη τηλεμετρίας ο οποίος είτε τα εικονίζει στην οθόνη είτε τα παρέχει, μέσω του Excel Data Streamer, στον υπολογιστή.

GPS.

Το GPS είναι ένα αυτόνομο σύστημα που παρέχει σειριακά τα δεδομένα λήψης στον πομπό HC-12 ο οποίος με τη σειρά του τα εκπέμπει ασύρματα (433 KHz) προς το άλλο HC-12 σε ρόλο δέκτη . Αποκωδικοποιούνται και δίνουν τα απαραίτητα δεδομένα στο Software της U-Blox, U-Center.

MPU6050.

Είναι η βασική αδρανειακή μονάδα μέτρησης και παρέχει στον Flight Controller τα δεδομένα Pitch, Roll, Yaw.

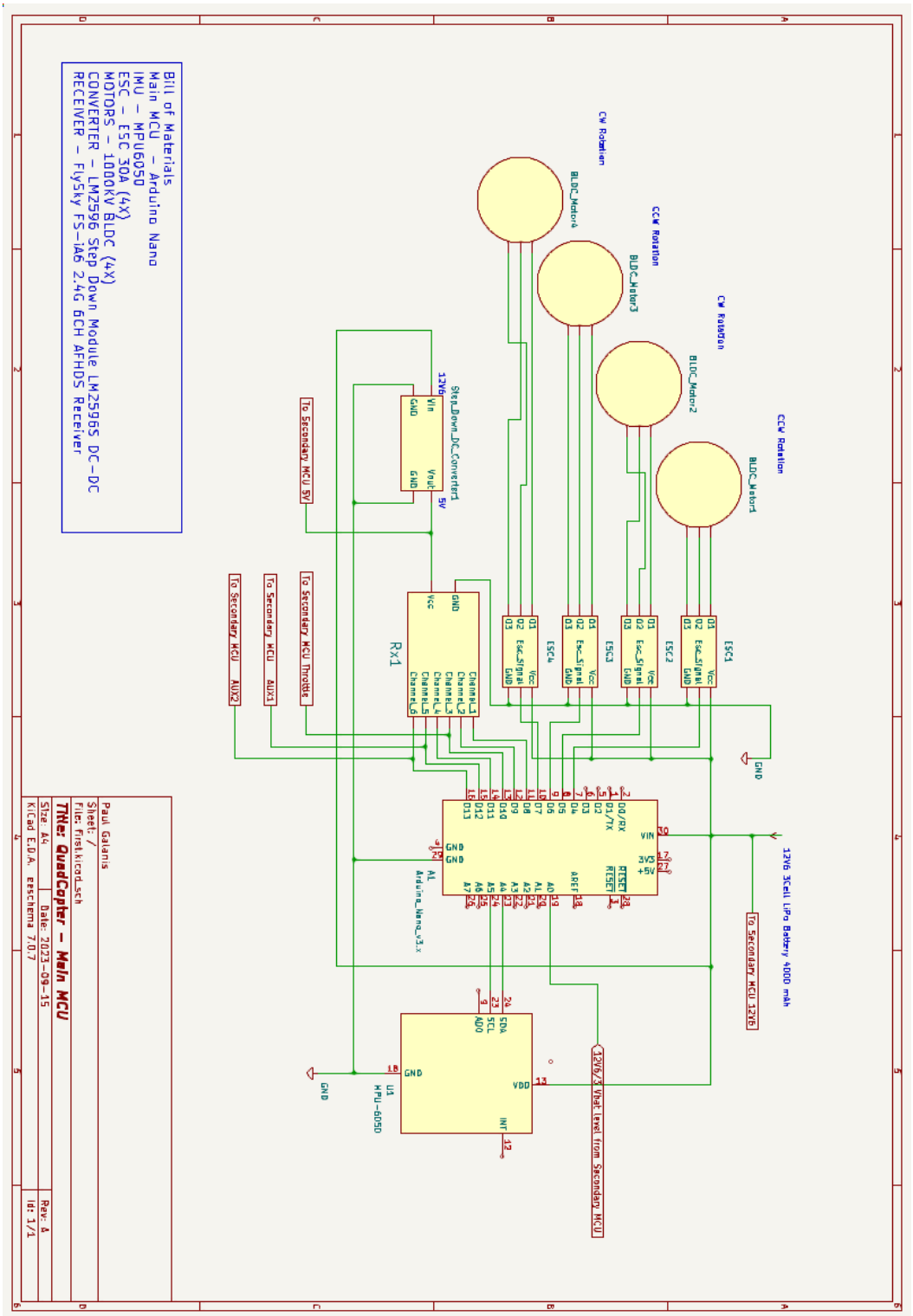
MPU9250.

Είναι η δευτερεύουσα αδρανειακή μονάδα μέτρησης και παρέχει στον Auxiliary Controller δεδομένα Pitch, Roll, Heading προς εκπομπή.

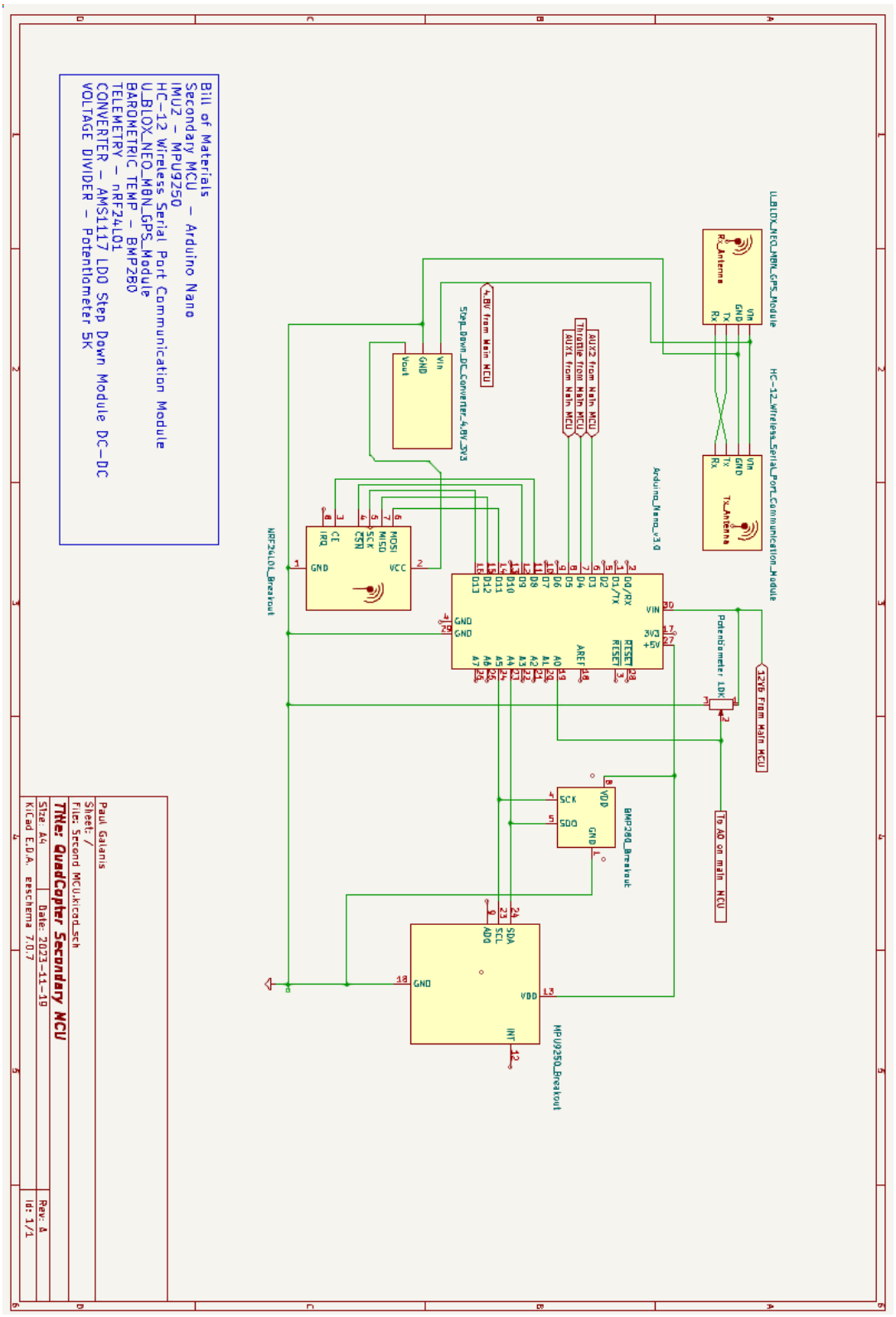
BMP280.

Είναι ο αισθητήρας θερμοκρασίας και βαρομετρικής πίεσης (υψόμετρο) και παρέχει στον Auxiliary Controller τα δεδομένα προς εκπομπή.

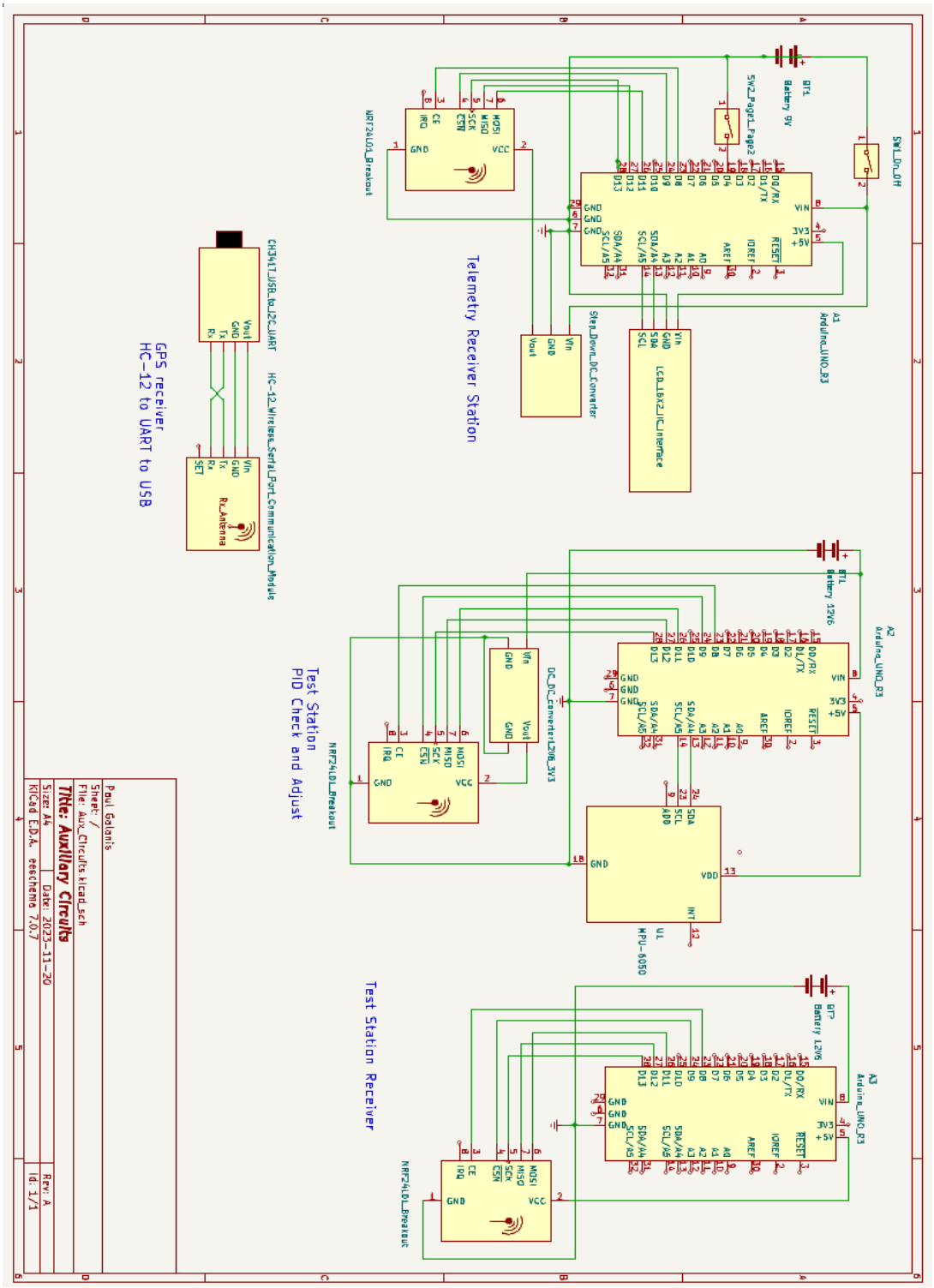
5.2 Ηλεκτρονικά Κυκλώματα



Εικόνα 188. Schematic Diagram - Main MCU



Εικόνα 189. Schematic Diagram – Secondary MCU



Εικόνα 190. Schematic Diagram – Auxiliary Circuits

5.3 Λίστα υλικών και Οικονομικός προϋπολογισμός

X type Quadcopter			
Περιγραφή	Εικόνα	Ποσότητα	Κόστος(€)
F450 Frame		1	13
4X 2212 1000KV 4xprops		4	44
Arduino Nano		2	4
MPU 6050		1	3
MPU9250		1	9
LM2596 Power Converter Step Down Module		1	2
AMS1117		1	1
Trimmer 5K		1	0.5
Dupont Cables		1	2
Prototype PCB Board		2	2
GPS Module NEO M8N		1	8
BMP280		1	2
NRF24L01 PA		1	3
HC-12 Tranceiver		2	8
CH341T		1	3
Lipo Battery 3s 2600 mAh		1	31
Total			135.5

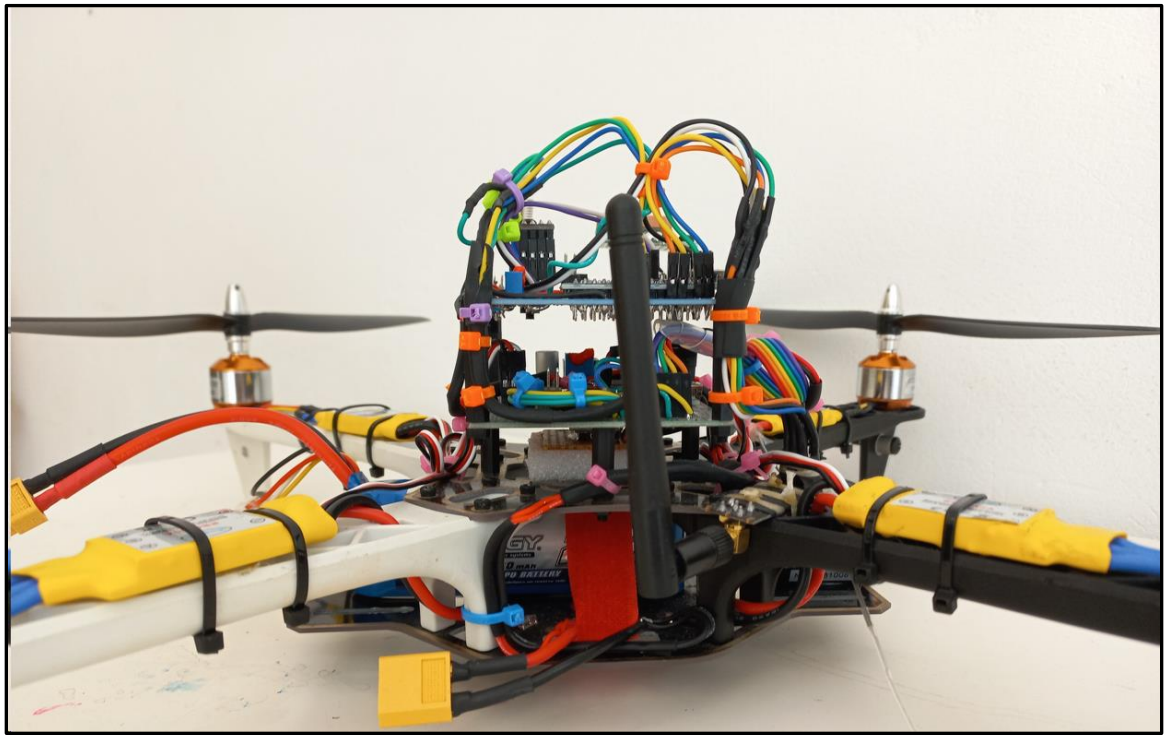
Σταθμός λήψης δεδομένων			
Περιγραφή	Εικόνα	Ποσότητα	Κόστος(€)
LCD 16X2 IIC		1	4
Arduino Uno R3		1	7
Plastic Box		1	5
Mini Switch		2	1
AMS1117		1	1
NRF24L01 PA		1	3
Total			21

Βάση Ελέγχου			
Περιγραφή	Εικόνα	Ποσότητα	Κόστος(€)
Ξυλεία		1	10
Bearings		2	4
Aluminium Tube 6063		1	4
MPU 6050		1	3
Arduino Uno R3		1	7
LM2596 Power		1	2
NRF24L01 PA		1	3
Total			33

Τηλεχειρισμός Flysky FS- i6			
Περιγραφή	Εικόνα	Ποσότητα	Κόστος(€)
Τηλεχειρισμός Flysky FS- i6		1	60
Total			60

Εικόνα 191. Bill of materials

Οι εικόνες παραπάνω παρουσιάζουν τα υλικά που χρησιμοποιήθηκαν για την κατασκευή όπως και το κόστος αυτών. Κρίνεται απαραίτητο να αναφερθεί ότι όλα τα υλικά αγοράστηκαν από προμηθευτές από την Κίνα. Το κόστος αυτών στην Ελληνική αγορά είναι περίπου τριπλάσιο. Αξιοσημείωτο είναι το γεγονός ότι αφενός μεν είναι φθηνότερα, αφετέρου πρέπει να συνυπολογιστεί και ο χρόνος που απαιτείται για να τα πάρει κάποιος στα χέρια του όπως επίσης και το γεγονός ότι πολλές φορές πρέπει να αγοραστούν μεγαλύτερες ποσότητες από τις αναγκαίες είτε λόγω αστοχιών του προμηθευτή είτε λόγω πιθανής καταστροφής του υλικού.

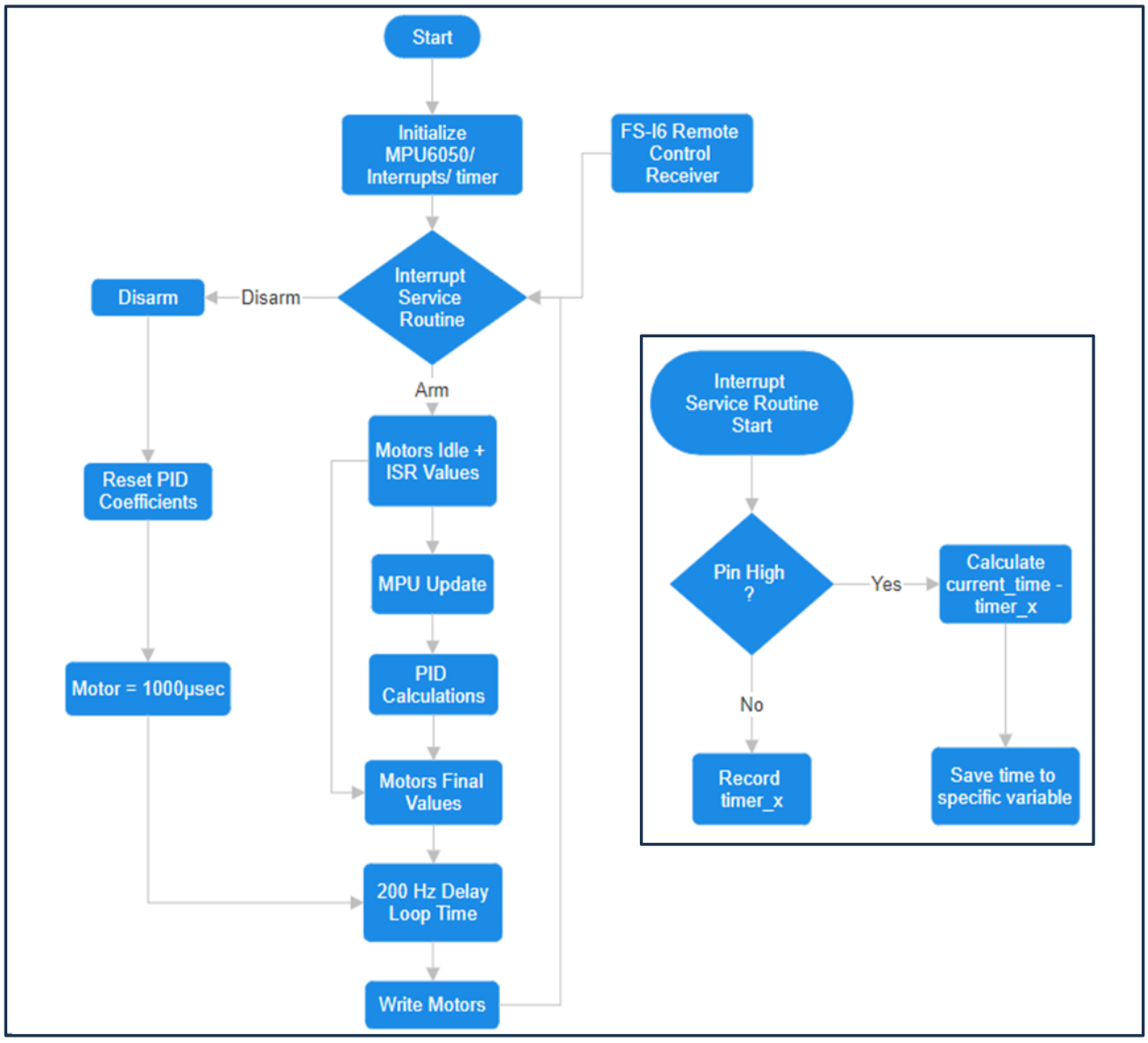


Εικόνα 192. Το Quadcopter

6. ΑΝΑΛΥΣΗ ΚΑΙ ΣΥΓΓΡΑΦΗ ΛΟΓΙΣΜΙΚΟΥ (SOFTWARE)

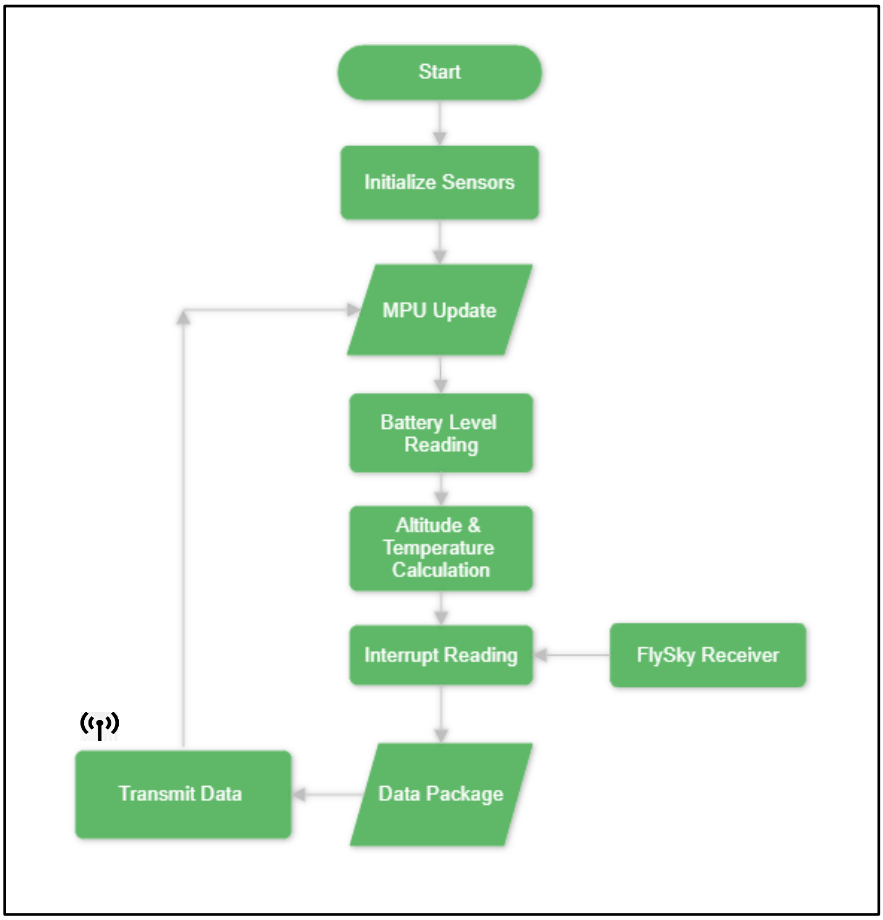
6.1 Διαγράμματα ροής

6.1.1 Διάγραμμα ροής Flight Controller



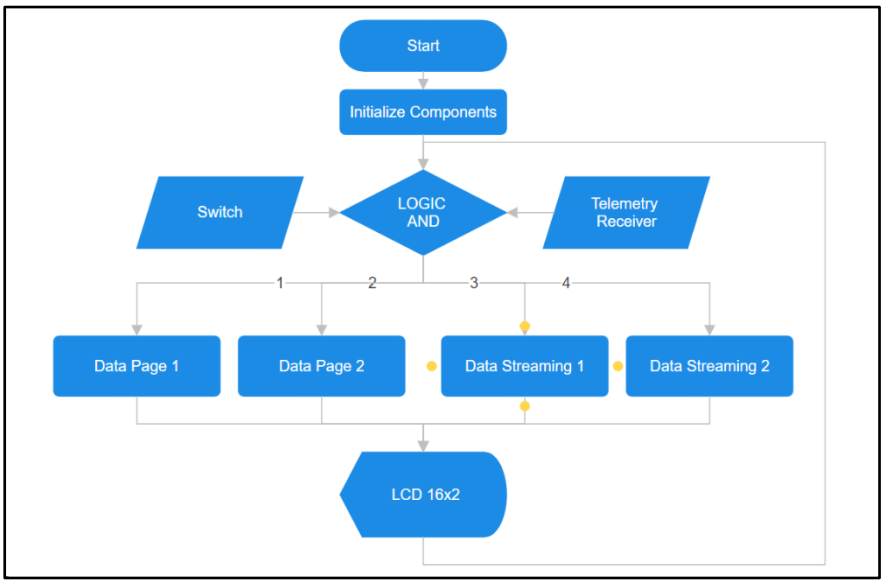
Εικόνα 193. Το διάγραμμα ροής του Flight Controller

6.1.2 Διάγραμμα ροής Secondary MCU



Εικόνα 194. Το διάγραμμα ροής του δεύτερου μικροελεγκτή

6.1.3 Διάγραμμα ροής Telemetry Receiver



Εικόνα 195. Το διάγραμμα ροής του δέκτη τηλεμετρίας

6.2 Κώδικας Προγράμματος

6.2.1 Flight Controller Code

```
//Part1/3************************************DMP implementation*************************************************//
#include "MPU6050_6Axis_MotionApps612.h"                                                                        //
#include "Wire.h"                                                                                               //
#include "I2Cdev.h"                                                                                             //
MPU6050 mpu;                                                                                                    //
bool dmpReady = false;  // set true if DMP init was successful                                                  //
uint8_t mpuIntStatus;   // holds actual interrupt status byte from MPU                                          //
uint8_t devStatus;      // return status after each device operation (0 = success, !0 = Rollerror)              //
uint16_t packetSize;    // expected DMP packet size (default is 42 bytes)                                       //
uint16_t fifoCount;     // count of all bytes currently in FIFO                                                 //
uint8_t fifoBuffer[64]; // FIFO storage buffer                                                                  //
Quaternion q;           // [w, x, y, z]         quaternion container                                           //
VectorFloat gravity;    // [x, y, z]            gravity vector                                                 //
float ypr[3], yaw, pitch, roll;                                                                                 //
//*************************************************************************************************Part1/3//
//Part1*****************************************PID implementation*************************************************//
   //.....................................................................Roll pid imlementation.....................................................//
   float pid_roll_setpoint, pid_roll_error, previous_pid_roll_error, roll_Kp, roll_Ki, roll_Kd, PID_roll_total ;
   float Kp_for_roll = 4.80 ;
   float Ki_for_roll = 0.018;
   float Kd_for_roll = 220.0 ;
   float PID_roll_total_max = 250 ;
   //.....................................................................Pitch pid implementation.....................................................//
   float pid_pitch_setpoint, pid_pitch_error, previous_pid_pitch_error, pitch_Kp, pitch_Ki, pitch_Kd, PID_pitch_total  ;
   float Kp_for_pitch = 4.80 ;
   float Ki_for_pitch =  0.018;
   float Kd_for_pitch = 220.0 ;
   float PID_pitch_total_max = PID_roll_total_max ;
```

```
//.....Yaw pid implementation.....//
    float pid_yaw_setpoint, pid_yaw_error, previous_pid_yaw_error, yaw_Kp, yaw_Ki, yaw_Kd, PID_yaw_total  ;
    float Kp_for_yaw = 6.0;
    float Ki_for_yaw = 0.002;
    float Kd_for_yaw = 0.0;
    float PID_yaw_total_max = 150.0;
//**********************************************************************************************Part1//
int roll_input, pitch_input, throttle_input, yaw_input, throttle, start ;
unsigned long timer_1,timer_2, timer_3, timer_4, motor1, motor2, motor3, motor4, esc1waveform,
esc2waveform, esc3waveform , esc4waveform , current_time, esc_loop_timer, zero_timer, general_timer ;
byte last_channel_1, last_channel_2, last_channel_3, last_channel_4 ;
void setup() {
  //Serial.begin(38400);
//Part2/3**********************************DMP implementation*************************************************//
        Wire.begin();
        Wire.setClock(400000);
        delay(4000);
        mpu.initialize();
        devStatus = mpu.dmpInitialize();
        mpu.setXGyroOffset(129);    // Setting offsets derived from IMU_Zero sketch //
        mpu.setYGyroOffset(-12);    // Different for every MPU6050                  //
        mpu.setZGyroOffset(-48);
        mpu.setXAccelOffset(-2316);
        mpu.setYAccelOffset(868);
        mpu.setZAccelOffset(968);
        mpu.CalibrateAccel(6);      // Offset implementation for Accel and Gyro //
        mpu.CalibrateGyro(6);       //                                          //
        mpu.PrintActiveOffsets();
        mpu.setDMPEnabled(true);
        dmpReady = true;
       packetSize = mpu.dmpGetFIFOPacketSize();
  //**********************************************************************************************Part2/3//
```

```
  PCICR |= (1 << PCIE0);                  //Pin Change Interrupt Control Register E0 PCINT 0 - 7
  PCMSK0 |= (1 << PCINT0);                //Set PCINT0 (digital input 8) to trigger an interrupt on state change,  Roll input
  PCMSK0 |= (1 << PCINT1);                //Set PCINT1 (digital input 9)to trigger an interrupt on state change,   Pitch input
  PCMSK0 |= (1 << PCINT2);                //Set PCINT2 (digital input 10)to trigger an interrupt on state change,  Throttle input
  PCMSK0 |= (1 << PCINT3);                //Set PCINT3 (digital input 11)to trigger an interrupt on state change,  Yaw input
  DDRD |= B11110000;                       // Set 4, 5, 6 ,7  outputs        4=> motor1 CCW , 5=> motor2 CW , 6=> motor3 CCW , 7=> motor4 CW
  zero_timer = micros();
  general_timer = micros();
  start = 0;
  }
void loop() {
    dmp_yaw_pitch_roll() ;    //aver. 2100 micros
    if (throttle_input<=1100 && yaw_input<1050 && yaw_input>1000 ) start=1;
    if (start==1 && throttle_input<=1100 && yaw_input<1520 && yaw_input>1480  ) start =2;
    if (start==2 && throttle_input<=1100 && yaw_input>1950) start=0;
    throttle = throttle_input;
if (start == 2){
    if (throttle >1500) throttle = 1500;
    motor1 = throttle + PID_roll_total + PID_pitch_total - PID_yaw_total ;
    motor2 = throttle + PID_roll_total - PID_pitch_total + PID_yaw_total ;
    motor3 = throttle - PID_roll_total - PID_pitch_total - PID_yaw_total ;
    motor4 = throttle - PID_roll_total + PID_pitch_total + PID_yaw_total ;
    if (motor1 < 1100) motor1 = 1100;
    if (motor2 < 1100) motor2 = 1100;
    if (motor3 < 1100) motor3 = 1100;
    if (motor4 < 1100) motor4 = 1100;
    if(motor1 > 2000)motor1 = 2000;
    if(motor2 > 2000)motor2 = 2000;
    if(motor3 > 2000)motor3 = 2000;
    if(motor4 > 2000)motor4 = 2000;
    }
```

```
    else {
    motor1 = 1000;
    motor2 = 1000;
    motor3 = 1000;
    motor4 = 1000;
    roll_Ki = 0;
    previous_pid_roll_error = 0;
    pitch_Ki = 0;
    previous_pid_pitch_error = 0;
    yaw_Ki = 0;
    previous_pid_yaw_error = 0;
    }
    //if (micros() - zero_timer > 4600)  start = 0 ;
    while(micros() - zero_timer < 5000);                    //program loop av. 4100 fixed 5000 micros   (200Hz)
    zero_timer = micros();
    PORTD |= B11110000;                                     // όλες οι κυματομορφές( esc - motors) HIGH
    esc1waveform = micros() + motor1;
    esc2waveform = micros() + motor2;
    esc3waveform = micros() + motor3;
    esc4waveform = micros() + motor4;
    //general_timer = micros();
      pidcalculation();     // <600 micros
    //if (micros() - general_timer > 600)  start = 0 ;
     while ( (PORTD & B11110000 ) != 0) {                   // μείνε στο loop μέχρι και οι 4 έξοδοι γίνουν LOW
    esc_loop_timer = micros();
    if(esc1waveform <= esc_loop_timer)PORTD &= B11101111;           //μέτρα χρόνο, μόλις γίνει ίσος με τον παραπάνω που υπολογίστηκε γύρνα την έξοδο σε LOW
    if(esc2waveform <= esc_loop_timer)PORTD &= B11011111;
    if(esc3waveform <= esc_loop_timer)PORTD &= B10111111;
    if(esc4waveform <= esc_loop_timer)PORTD &= B01111111;
    }
}
```

```
//Part3/3**********************************************DMP implementation**************************************************//
void dmp_yaw_pitch_roll() {
        if (mpu.dmpGetCurrentFIFOPacket(fifoBuffer)) {
        mpu.dmpGetQuaternion(&q, fifoBuffer);
        mpu.dmpGetGravity(&gravity, &q);
        mpu.dmpGetYawPitchRoll(ypr, &q, &gravity);
        yaw = ypr[0] * (180 / M_PI);
        pitch = -ypr[1] * (180 / M_PI) ;
        roll = ypr[2] * (180 / M_PI);
        }
    }
//*************************************************************************************************************Part3/3//
//************************************************PID CALCULATIONS*************************************************//
void pidcalculation(){
//.....................................................................PID ROLL.....................................................//
   pid_roll_setpoint = map(roll_input,1000, 2000, -10.0, 10.0);
   if (pid_roll_setpoint > -2.0 && pid_roll_setpoint < 2.0) pid_roll_setpoint = 0.0;
   pid_roll_error = roll - pid_roll_setpoint ;
   roll_Kp = pid_roll_error * Kp_for_roll ;
   if (roll_input > 1492  && roll_input <1508 ) roll_Ki += pid_roll_error * Ki_for_roll ;
   else roll_Ki = 0 ;
   roll_Kd = (pid_roll_error - previous_pid_roll_error) * Kd_for_roll;
   PID_roll_total = roll_Kp + roll_Ki + roll_Kd;
   PID_roll_total = constrain(PID_roll_total, -PID_roll_total_max, PID_roll_total_max);
   previous_pid_roll_error = pid_roll_error;
//.....................................................................PID PITCH.....................................................//
   pid_pitch_setpoint = map(pitch_input,1000, 2000, -10.0, 10.0);    //10
   if (pid_pitch_setpoint > -2.0 && pid_pitch_setpoint < 2.0) pid_pitch_setpoint = 0.0;    //dead band to eliminate noise
   pid_pitch_error = pitch - pid_pitch_setpoint ;
   pitch_Kp = pid_pitch_error * Kp_for_pitch ;
   if (pitch_input > 1492  && pitch_input <1508 ) pitch_Ki += pid_pitch_error * Ki_for_pitch ;
   else pitch_Ki = 0 ;
```

```
    pitch_Kd = (pid_pitch_error - previous_pid_pitch_error) * Kd_for_pitch;
    PID_pitch_total = pitch_Kp + pitch_Ki + pitch_Kd;
    PID_pitch_total = constrain(PID_pitch_total, -PID_pitch_total_max, PID_pitch_total_max);
    previous_pid_pitch_error = pid_pitch_error;
//.....PID YAW.....//
    pid_yaw_setpoint = map(yaw_input,1000, 2000, -30.0, 30.0);
    if (pid_yaw_setpoint > -2 && pid_yaw_setpoint < 2) pid_yaw_setpoint = 0.0;    //dead band to eliminate noise
    pid_yaw_error = yaw - pid_yaw_setpoint ;
    yaw_Kp = pid_yaw_error * Kp_for_yaw ;
    yaw_Ki += pid_yaw_error * Ki_for_yaw ;
    yaw_Kd = (pid_yaw_error - previous_pid_yaw_error) * Kd_for_yaw;
    PID_yaw_total = yaw_Kp + yaw_Ki + yaw_Kd;
    PID_yaw_total = constrain(PID_yaw_total, -PID_yaw_total_max, PID_yaw_total_max);
    previous_pid_yaw_error = pid_yaw_error;
//**********//
}
//**********Interrupt Service Routine to track and measure state change and duty cycle of  receiver waveform respectively**********//

ISR(PCINT0_vect){
  current_time = micros();
  if(PINB & B00000001){
    if(last_channel_1 == 0){
      last_channel_1 = 1;
      timer_1 = current_time;
    }
  }
  else if(last_channel_1 == 1){
    last_channel_1 = 0;
    roll_input = current_time - timer_1;
  }
```

```
 if(PINB & B00000010 ){
    if(last_channel_2 == 0){
      last_channel_2 = 1;
      timer_2 = current_time;
    }
  }
  else if(last_channel_2 == 1){
    last_channel_2 = 0;
    pitch_input = current_time - timer_2;
  }
  if(PINB & B00000100 ){
    if(last_channel_3 == 0){
      last_channel_3 = 1;
      timer_3 = current_time;
    }
  }
  else if(last_channel_3 == 1){
    last_channel_3 = 0;
    throttle_input = current_time - timer_3;
 }
  if(PINB & B00001000 ){
    if(last_channel_4 == 0){
      last_channel_4 = 1;
      timer_4 = current_time;
    }
  }
  else if(last_channel_4 == 1){
    last_channel_4 = 0;
    yaw_input = current_time - timer_4;
  }
```

```
  /*  for future use
  //  SCW C
  if(PINB & B00010000 ){
    if(last_channel_5 == 0){
      last_channel_5 = 1;
      timer_5 = current_time;
    }
  }
  else if(last_channel_5 == 1){
    last_channel_5 = 0;
    SWC_input = current_time - timer_5;
  }
  // SWC B
  if(PINB & B00100000 ){
    if(last_channel_6 == 0){
      last_channel_6 = 1;
      timer_6 = current_time;
    }
  }
  else if(last_channel_6 == 1){
    last_channel_6 = 0;
    SWB_input = current_time - timer_6;
  }  for future use   */
}
```

//Copy - Paste Compilation verified

6.2.2 Secondary MCU Code

```
#include "MPU9250.h"
#include <ErriezBMX280.h>
#include <SPI.h>
#include <nRF24L01.h>
#include <RF24.h>
MPU9250 aux_mpu;                                        // MPU 9250 object
ErriezBMX280 bmp280 = ErriezBMX280(0x76);               // BMP 280 object
//*****************************RF24 Object & Data_Package*********************************//
RF24 radio(8, 9); // CE, CSN         `                                                      //
const byte address[6] = "00001";                                                           //
   struct Data_Package {                                                                   //
   float  heading_tx, altitude_tx, temp_tx, throttle_tx, command_tx, percentage_tx, pitch_tx, roll_tx ;  };      //
   Data_Package data;                                                                      //
//**** ***************************************************************************************//
float tmp_pres, tmp_acum, altitude, temp ;                  // BMP 280 variables declaration
float yaw,pitch,roll ;                                      // MPU 9250 variables declaration
unsigned long timer_1, timer_2, current_time, zero_timer ;  // Variables related to the interrupts
byte last_channel_1, last_channel_2 ;                       // Same as above
int  throttle_input, command_input, throttle, command ;     // Same as above
float Vbat_hot, percentage ;                                // Battery percentage related variables declaration
void setup() {
   Wire.begin();
   delay(500);
   pinMode(A0,INPUT);
   PCICR |= (1 << PCIE2);                    //Pin Change Interrupt Control Register E2 PCINT 16 - 23
   PCMSK2 |= (1 << PCINT20);                 //Set PCINT20(digital input 4)to trigger an interrupt on state change,  Throttle input
   PCMSK2 |= (1 << PCINT21);                 //Set PCINT20(digital input 5)to trigger an interrupt on state change,  Command input
   //*********************************MPU 9250 Setup ***********************//
   aux_mpu.setup(0x68);                                                                    //
   aux_mpu.setAccBias(-40.62, 31.50, 62.18);                                               //
   aux_mpu.setGyroBias(1.22, -0.98, 0.28);                                                 //
   aux_mpu.setMagBias(62.11, -76.56, -125.70);                                             //
   aux_mpu.setMagScale(0.92, 0.98, 1.11);                                                  //
   aux_mpu.setMagneticDeclination(5.1);                                                    //
//************************************************************************//
```

```
//*********************************BMP 280 Setup ***************************************//
  bmp280.begin();                                                                        //
  bmp280.setSampling(BMX280_MODE_NORMAL,     // SLEEP, FORCED, NORMAL                  //
                BMX280_SAMPLING_X2,          // Temp:  NONE, X1, X2, X4, X8, X16       //
                BMX280_SAMPLING_X4,          // Press: NONE, X1, X2, X4, X8, X16       //
                BMX280_SAMPLING_NONE,    // Hum:   NONE, X1, X2, X4, X8, X16 (BME280)  //
                BMX280_FILTER_X16,           // OFF, X2, X4, X8, X16                   //
                BMX280_STANDBY_MS_0_5);  // 0_5, 10, 20, 62_5, 125, 250, 500, 1000     //
  delay(200);                                                                            //
  for (int cal = 0; cal < 20 ; cal ++){           //Average of 20 readings loop          //
  tmp_pres = bmp280.readPressure() / 100.0F;                                             //
  delay(50);                                                                             //
  tmp_acum += tmp_pres;                                                                  //
  }                                                                                      //
//**************************************************************************************//
//*************************************************nRF  Setup*****************************************************************//
 radio.begin();                                                                                                                  //
 radio.openWritingPipe(address);              // pipe address 00001                                                              //
 radio.setChannel(100);                       // set channel = 2400 + (number) ,  2400+100 = 2500 Mhz                            //
 radio.setPALevel(RF24_PA_MAX);         // RF24_PA_MIN=-18dBm, RF24_PA_LOW=-12dBm, RF24_PA_HIGH=-6dBM, and RF24_PA_MAX=0dBm //
 radio.setDataRate(RF24_250KBPS);       // RF24_250KBPS for 250kbs, RF24_1MBPS for 1Mbps, or RF24_2MBPS for 2Mbps          //
 radio.stopListening();                       // Transmit only                                                                   //
//*******************************************************************************************************************************//
zero_timer = micros();
}
void loop() {
//*****************************************MPU 9250 values ***********************************//
    if (aux_mpu.update()) {                                                                    //
          yaw = aux_mpu.getYaw();                                                              //
         pitch = aux_mpu.getPitch();                                                           //
         roll = aux_mpu.getRoll();                                                             //
         if (yaw <0.0 ) yaw  = yaw +360.0 ;                                                    //
    }                                                                                          //
//**********************************************************************************************//
```

```
//**********************************Battery Percentage*************************************************//
    Vbat_hot  = analogRead(A0);                                    // 845 => 12.6 Volt,  746  =>11.12 Volt      //
    Vbat_hot = (0.999 * Vbat_hot) + (0.001 * analogRead(A0));      // Complementary filter                      //
    percentage = map(Vbat_hot,845,746,100,15);                     // map Vbat_hot to percentage                //
    if (percentage > 100) percentage  = 100 ;                      //Upper Limit                               //
    if (percentage <15 ) percentage = 15 ;                         // Lower Limit                              //
//*****************************************************************************************************//
//***************************************BMP 280 Altitude & Temperature*********************************//
    static const float pressure_cal = tmp_acum / 20 ;  // Average reference pressure for altitude calculation  //
    altitude = bmp280.readAltitude(pressure_cal);    //  altitude calculation in terms of reference pressure   //
    if ( altitude  < 0.0 ) altitude = 0.0 ;                //  canceling negative values                       //
    temp = bmp280.readTemperature();                 // Temperature reading                                   //
//*****************************************************************************************************//
//**********************throttle & command assignment to variables to be transmitted**********************//
    throttle  = throttle_input;                                                                              //
    command   = command_input;                                                                               //
//*****************************************************************************************************//
//************************************Data Structure to be transmitted*********************************//
    data.heading_tx = yaw ;                                                                                  //
    data.pitch_tx = pitch ;                                                                                  //
    data.roll_tx = roll ;                                                                                    //
    data.altitude_tx = altitude ;                                                                            //
    data.throttle_tx = throttle;                                                                             //
    data.temp_tx = temp ;                                                                                    //
    data.command_tx = command ;                                                                              //
    data.percentage_tx = percentage ;                                                                        //
    radio.write(&data, sizeof(Data_Package));          // Transmit Data Package                              //
//*****************************************************************************************************//
}
```

```
//****************Interrupt Service Routine to track and measure state change and duty cycle of  receiver waveform respectively***************//
ISR(PCINT2_vect){                                                                                                                                //
  current_time = micros();                                                                                                                       //
  if(PIND & B00010000 ){                                                                                                                         //
    if(last_channel_1 == 0){                                                                                                                     //
      last_channel_1 = 1;                                                                                                                        //
      timer_1 = current_time;                                                                                                                    //
    }                                                                                                                                            //
  }                                                                                                                                              //
  else if(last_channel_1 == 1){                                                                                                                  //
    last_channel_1 = 0;                                                                                                                          //
    throttle_input = current_time - timer_1;                  //Throttle                                                                       //
  }                                                                                                                                              //
  if(PIND & B00100000 ){                                                                                                                         //
    if(last_channel_2 == 0){                                                                                                                     //
      last_channel_2 = 1;                                                                                                                        //
      timer_2 = current_time;                                                                                                                    //
    }                                                                                                                                            //
  }                                                                                                                                              //
  else if(last_channel_2 == 1){                                                                                                                  //
    last_channel_2 = 0;                                                                                                                          //
    command_input = current_time - timer_2;                   //Command                                                                        //
  }                                                                                                                                              //
}                                                                                                                                                //
 //*****************************************************End*****************************************************************************//
```

// Copy – Paste Compilation verified//

6.2.3 Telemetry Receiver Code

```
#include <SPI.h>

#include <nRF24L01.h>

#include <RF24.h>

#include <LiquidCrystal_I2C.h>

LiquidCrystal_I2C lcd(0x27, 16, 2);

RF24 radio(8, 9); // CE, CSN

const byte address[6] = "00001";

struct Data_Package {

      float  heading_tx, altitude_tx, temp_tx, throttle_tx, command_tx, percentage_tx,pitch_tx, roll_tx ;

                        };

      Data_Package data;

unsigned long timer ;

int switch_state, switch_command ;

void setup() {

  Serial.begin(250000);

  pinMode(4,INPUT_PULLUP);

  lcd.init(); // initialize the lcd

  lcd.backlight();

  lcd.clear();

  lcd.setCursor(0, 0);

  lcd.print("Telemetry Rcvr");

  lcd.setCursor(0, 1);

  lcd.print("Station  PG.V1.4");

  delay(3000);

radio.begin();

  radio.openReadingPipe(0, address);

  radio.setChannel(100);

  radio.setPALevel(RF24_PA_MAX);          //RF24_PA_MIN=-18dBm, RF24_PA_LOW=-12dBm, RF24_PA_HIGH=-6dBM, and RF24_PA_MAX=0dBm.

  radio.setDataRate(RF24_250KBPS);          //RF24_250KBPS for 250kbs, RF24_1MBPS for 1Mbps, or RF24_2MBPS for 2Mbps

  //radio.setPayloadSize(32);                    // Max 32 bytes int,float 4 bytes each, string one byte per character

  radio.startListening();

  timer = micros();

}
```

```
void loop() {

  if (radio.available()) {

  radio.read(&data, sizeof(Data_Package));

  if (data.throttle_tx <1000 ) data.throttle_tx = 1000;

  switch_state = digitalRead(4);

    if  (data.command_tx >1400 && data.command_tx <1600 && switch_state == LOW) {

       lcd.clear();

       lcd.setCursor(1, 0);

       lcd.print("Data Streaming");

       Serial.print(data.throttle_tx,0);

       Serial.print(",");

       Serial.print(data.percentage_tx,0);

       Serial.print(",");

       Serial.print(data.temp_tx);

       Serial.print(",");

       Serial.print(data.altitude_tx);

       Serial.print(",");

       Serial.print(data.pitch_tx);

       Serial.print(",");

       Serial.print(data.roll_tx);

       Serial.print(",");

       Serial.print(data.heading_tx);

       Serial.print(",");

       Serial.print(data.command_tx);

       Serial.print(",");

       Serial.println();

       delay(100);

        }

    else if   (data.command_tx >1400 && data.command_tx <1600 && switch_state == HIGH) {

            lcd.clear();

            lcd.setCursor(1, 0);

            lcd.print(".....");

            Serial.print(data.pitch_tx);

            Serial.print(",");

            Serial.print(data.roll_tx);
```

```
            Serial.print(",");
            Serial.println();
            }
    else if (switch_state == LOW) {
    lcd.clear();
    lcd.setCursor(0,0);
    lcd.print("T=");
    lcd.print(data.throttle_tx,0);
    lcd.setCursor(9,0);
    lcd.print("V(%)");
    lcd.print(data.percentage_tx,0);
    lcd.setCursor(0,1);
    lcd.print("Tmp:");
    lcd.print(data.temp_tx,1);
    lcd.setCursor(9,1);
    lcd.print("A:");
    lcd.print(data.altitude_tx,1);
    delay(200);
    }
    else {
    lcd.clear();
    lcd.setCursor(0,0);
    lcd.print("H(D):");
    lcd.print(data.heading_tx,0);
    lcd.setCursor(10,0);
    lcd.print("C:");
    lcd.print(data.command_tx);
    lcd.setCursor(0,1);
    lcd.print("P:");
    lcd.print(data.pitch_tx,1);
    lcd.setCursor(9,1);
    lcd.print("R:");
    lcd.print(data.roll_tx,1);
    delay(200);
    }
}
```

```
else {
   lcd.clear();
   lcd.setCursor(1, 0);
   lcd.print("***No Data***");
   lcd.setCursor(0, 1);
   lcd.print("***Check Link***");
   delay(200);
      }
}
```

//// Copy – Paste Compilation verified////

ΑΠΟΤΕΛΕΣΜΑΤΑ

Στην ενότητα αυτή θα αναλυθούν τα αποτελέσματα από την λειτουργία της κατασκευής όσο αφορά στα πτητικά χαρακτηριστικά όσο και στα υπόλοιπες προδιαγραφές που πραγματεύεται η εργασία. Το project υλοποιήθηκε ως επί το πλείστων με την επιλογή χαμηλού κόστους εξαρτημάτων και στις παρακάτω γραμμές θα παρουσιαστούν οι επιδόσεις ή αδυναμίες αυτών.

Frame F450

Το πλαίσιο αυτό είναι μια ρεπλίκα του DJ1 Flame Wheel 450 (F450). Είναι ελαφρύ και στιβαρό, ανταποκρίθηκε πλήρως στις απαιτήσεις της κατασκευής και παρά το γεγονός ότι ταλαιπωρήθηκε αρκετά κράτησε τα περισσότερα από τα αρχικά του χαρακτηριστικά. Το κόστος του βρίσκεται στα 13 περίπου ευρώ , τιμή που το καθιστά best- seller για την ανάπτυξη πρωτότυπων κατασκευών.

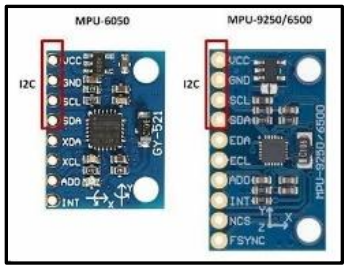
BLDC Motors , ESC's , Props

Τα BLDC μοτέρ της κατασκευής είναι τα 2212, 1000KV τα οποία ταίριαξαν με προπέλες 10 ιντσών, βήματος 4.5^{o} και οδηγήθηκαν από no name κινέζικα ESC's. Το σετ αυτών των εξαρτημάτων ξεπέρασε όλες τις αναμενόμενες προσδοκίες. Αν και λειτούργησαν εξαντλητικά πάνω από 50 ώρες δεν παρουσίασαν σημάδια κόπωσης. Σε συνδυασμό με μία 3Cell Lipo μπαταρία στα 12V5 κατάφεραν thrust περί τα 800 γραμμάρια έκαστο.

Arduino Nano, Uno

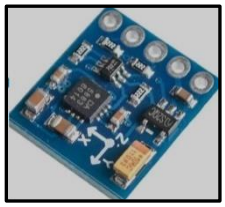
Τα Arduino Nano - Uno χρησιμοποιήθηκαν ως μικροελεγκτές σε 5 διαφορετικές κατασκευές - αποστολές. Ανταποκρίθηκαν πλήρως στις απαιτήσεις της κατασκευής. Το χαμηλό τους κόστος όπως επίσης και η υποστήριξη σε βιβλιοθήκες τα καθιστούν κατάλληλα στις περισσότερες των περιπτώσεων. Είναι σχετικά παρωχημένα και χαμηλά σε συχνότητα λειτουργίας, όμως στην περίπτωση του Flight Controller κατάφεραν μια συχνότητα ανανέωσης 200 Hz που κρίθηκε ικανοποιητική από τα πτητικά αποτελέσματα του τετρακόπτερου.

MPU-6050, MPU-9250



Τα MPU – 6050 και MPU-9250 χρησιμοποιήθηκαν σαν αδρανειακές μονάδες μέτρησης. Το MPU-6050 παρωχημένο πια έχει τον ρόλο του IMU για τον Flight Controller . Έχει πολύ μεγάλη υποστήριξη από βιβλιοθήκες και με την χρήση DMP κατάφερε ικανοποιητικά αποτελέσματα. Με την χρήση του εσωτερικού Digital Low Pass Filter εξασθένησε αρκετά το θόρυβο για το επιταχυνσιόμετρο και το γυροσκόπιο. Το MPU-9250 έρχεται σαν αντικαταστάτης του προηγούμενου και ενσωματώνει ίσως την καλύτερη πυξίδα AK 8963. Κατά την διάρκεια ανάπτυξης της κατασκευής χρησιμοποιήθηκε ως δεύτερη αδρανειακή μονάδα για την αποστολή Pitch , Roll και Heading στον σταθμό τηλεμετρίας. Σίγουρα κρίνεται καλύτερο σε απόδοση από το 6050 και στην επόμενη έκδοση του Flight Controller θα χρησιμοποιηθεί ως βασικό IMU.

Ηλεκτρονική πυξίδα QMC5883L



Η ηλεκτρονική πυξίδα QMC5883L είναι ο αντικαταστάτης της ηλεκτρονικής πυξίδας της Honeywell HMC5883L. Δεν παρουσίασε ικανοποιητικά αποτελέσματα σε σχέση με την ακρίβεια Heading και για αυτό τον λόγο αντικαταστάθηκε από την AK 8963 του MPU-9250.

BMP280

Ο αισθητήρας βαρομετρικής πίεσης και θερμοκρασίας BMP 280 της Bosch παρείχε υψηλής ακρίβειας δεδομένα σχετικά με την θερμοκρασία και την βαρομετρική πίεση περιβάλλοντος παρά το χαμηλό κόστος του (περί τα 2 ευρώ) .

Πομποδέκτες NRF24L01 PA, HC-12

Οι δύο αυτοί πομποδέκτες υλοποίησαν τα ασύρματα δίκτυα δεδομένων . Το NRF24L01 PA στα 2.4 GHz και με ρυθμό διάδοσης δεδομένων 250Kbps και το HC-12 στα 433 MHz και bitrate 9600bps χρησιμοποιήθηκαν για την μετάδοση δεδομένων από το τετρακόπτερο/σταθμό ελέγχου και τον δέκτη GPS στον υπολογιστή αντίστοιχα. Είναι και οι δύο πομποδέκτες χαμηλού κόστους (3 ευρώ και 4 ευρώ αντίστοιχα) και ανταποκρίθηκαν απόλυτα στις απαιτήσεις του project.

GPS Module NEO M8N

Το GPS βασίζεται στο NEO M8N chip της U-Blox. Χρησιμοποιήθηκε σε συνδυασμό με το αντίστοιχο software της εταιρίας, U-Center και παρέχει πλήθος χαρακτηριστικών και δεδομένων GPS σε γραφικό περιβάλλον. Μειονέκτημα μπορεί να αναφερθεί το γεγονός ότι για να απεικονίσει δεδομένα σε χάρτη σε πραγματικό χρόνο πρέπει να γίνει συνδρομή στη Google ώστε να αποκτηθεί το αντίστοιχο Maps JavaScript API Key.

Τηλεχειρισμός Flysky FS- i6

Ο τηλεχειρισμός προϋπήρχε . Είναι ένας εξακάναλος τηλεχειρισμός χαμηλού κόστους (περί τα 60 ευρώ), Η αγορά προσφέρει πλήθος επιλογών με την τιμή κυρίως να καθορίζει τα διαθέσιμα κανάλια , την στάθμη εκπομπής και τα χαρακτηριστικά εκπομπής – λήψης (π.χ. frequency hopping).

Όλα τα υπόλοιπα ηλεκτρονικά εξαρτήματα είναι βασικά και κατά περίπτωση χρησιμοποιήθηκαν για να απεικονίσουν δεδομένα (Liquid Crystal Display) ή για να υποστηρίξουν τροφοδοσίες άλλων εξαρτημάτων(Step Down Converters).

Το τετρακόπτερο σαν project στην ολότητα του απέδωσε ικανοποιητικά. Προέκυψε ένα σκάφος με μέγιστη αναλογία απόδοσης/βάρους περίπου 3 , ικανό να εξυπηρετήσει payload 700 g. σύμφωνα με τις προδιαγραφές . Η μπαταρία που επιλέχθηκε του προσδίδει αυτονομία περίπου 7 με 8 λεπτά ανάλογα με την χρήση στην βασική του διαμόρφωση. Η τηλεμετρία τουλάχιστον στα δεδομένα που επιλέχθηκαν (Throttle , Battery Voltage, Temperature, Altitude , Heading, Pitch , Roll) απέδωσε καλώς και τα δεδομένα μεταδόθηκαν ασύρματα σε μια απόσταση περίπου 400 μέτρων σε ανοιχτό πεδίο.

Ο σταθμός βάσης διαδραμάτισε καταλυτικό ρόλο στην ανάπτυξη του project αφού εκεί αναπτύχθηκε ο κώδικας του flight controller και ελέγχθηκε για την σταθερότητα του, με πρωταρχικό σκοπό την ασφάλεια του χρήστη αλλά και του εξοπλισμού. Τα δεδομένα εκπέμπονταν και αναλύονταν ασύρματα στον υπολογιστή για την εξαγωγή ασφαλών συμπερασμάτων. Το τετρακόπτερο μετά την ρύθμιση του στον σταθμό βάσης πέταξε επιτυχώς με την πρώτη δοκιμή χωρίς καμία έκπληξη. Χρειάζονται μικρό ρυθμίσεις (fine tuning) στο PID ώστε να τελειοποιηθεί η ομαλότητα στην πτήση ανάλογα με το payload και τις συνθήκες του περιβάλλοντος.

ΣΥΜΠΕΡΑΣΜΑΤΑ

Κατά την διάρκεια αποπεράτωσης της διπλωματικής αυτής εργασίας πάμπολλες ήταν οι φορές που οδηγήθηκα σε αδιέξοδο. Με υπομονή και επιμονή κατέληξα σε λύσεις γνωρίζοντας ότι δεν είναι οι βέλτιστες αν και εν τέλει απέδωσαν. Οποιοδήποτε project περνάει από πολλά στάδια βελτίωσης και είναι συνεχώς εξελισσόμενο.

Κατά αυτόν τον τρόπο στην επόμενη έκδοση του τετρακόπτερου ο Flight Controller θα χρησιμοποιεί το MPU-9250 σαν βασική μονάδα IMU, που αποδεδειγμένα φαίνεται να λειτουργεί καλύτερα από τον προκάτοχο του. Παράλληλα πρέπει να αναπτυχθεί ο κώδικας ώστε να μπορεί το τετρακόπτερο να διατηρεί το ύψος του σε κατάσταση hover. Απαιτούνται εξαντλητικές δοκιμές σε διάφορα σενάρια σε σχέση με το payload ή τις καιρικές συνθήκες (π.χ. ριπές ανέμου). Σίγουρα η χρήση ενός τυπωμένου κυκλώματος θα αναβαθμίσει το project μειώνοντας το βάρος του και τον ηλεκτρικό θόρυβο (κατάργηση καλωδίων Dupont). Επίσης η χρήση εκτυπωτή 3D θα αναβαθμίσει αισθητικά το τετρακόπτερο με την χρήση θηκών που θα φιλοξενούν τα διάφορα module.

Ένα drone σαν πλατφόρμα μπορεί να φιλοξενήσει εκατοντάδες ιδέες. Ειλικρινά γράφοντας πιστεύω ότι στα πλαίσια ενός μεταπτυχιακού προγράμματος είναι δύσκολο κάποιος να συλλάβει μια ιδέα που δεν έχει ήδη πραγματοποιηθεί. Αυτό συμβαίνει κυρίως γιατί δεν υπάρχει ο απαιτούμενος χρόνος και οι απαραίτητοι οικονομικοί πόροι σε ατομικό επίπεδο. Η αγορά προσφέρει λύσεις για όλες τις απαιτήσεις με την τιμή κυρίως να καθορίζει τις αποδόσεις των υλικών. Έτσι επιλέγοντας έναν πιο γρήγορο μικροελεγκτή , καλύτερα μοτέρ , καλύτερους ESC’s πιο ακριβούς πομποδέκτες, πιο μεγάλη μπαταρία , το αποτέλεσμα θα είναι δραματικά καλύτερο με μόνο μειονέκτημα το αυξημένο κόστος .

Τελειώνοντας θα ήθελα να επισημάνω το γεγονός ότι κατά την διάρκεια αποπεράτωσης της διπλωματικής εργασίας πολλές φορές ένιωσα ανασφάλεια κατά την διάρκεια των δοκιμών αν και πάντα λάμβανα μέτρα πρόληψης ατυχήματος. Να τονίσω ότι οι Lipo μπαταρίες είναι πολύ δυνατές (η της κατασκευής μπορεί να αποδώσει στιγμιαία 90 A) και ότι τα μοτέρ με τις προπέλες μπορούν να περιστρέφονται άνετα με 6000 - 7000 στροφές ανά λεπτό χαρακτηριστικά που αν μη τι άλλο πρέπει να λάβουμε σοβαρά υπόψη .

ΒΙΒΛΙΟΓΡΑΦΙΑ

[1] K. P. V. a. G. J. Vachtsevanos, Handbook of Unmanned Aerial Vehicles. S, Springer, 2015.

[2] R. F. Stengel, "Relationship between euler-angle rates and body-axis rates," [Online]. Available: http://www.stengel.mycpanel.princeton.edu/Quaternions.pdf.

[3] RavvenLabs, "A tutorial on Euler angles and quaternions," [Online]. Available: .

[4] Michael_J_Caruso, "Applications of magnetic sensors for low cost Compass Systems".

[5] Infineon, "Sensing – Magnetic Compass with Tilt Compensation," [Online]. Available: https://www.infineon.com/dgdl/Infineon-AN2272_PSoC_1_Sensing_Magnetic_Compass_with_Tilt_Compensation-ApplicationNotes-v04_00-EN.pdf?fileId=8ac78c8c7cdc391c017d0731b0d05573. [Accessed June 2023].

[6] Infineon, "https://www.infineon.com," [Online]. Available: https://www.infineon.com/dgdl/Infineon-AN2272_PSoC_1_Sensing_Magnetic_Compass_with_Tilt_Compensation-ApplicationNotes-v04_00-EN.pdf?fileId=8ac78c8c7cdc391c017d0731b0d05573. [Accessed April 2023].

[7] www.pololu.com, "Using LSM303DLH for a tilt compensated electronic compass," [Online]. Available: https://www.pololu.com/file/0J434/LSM303DLH-compass-app-note.pdf.

[8] pololu.com, "AN3192 application note," [Online]. Available: https://www.pololu.com/file/0J434/LSM303DLH-compass-app-note.pdf.

[9] ZurichInstruments, "Principles of pid controllers," July 2023. [Online]. Available: https://www.zhinst.com/sites/default/files/documents/2023-07/zi_whitepaper_principles_of_pid_controllers.pdf.

[10] Q.-G. W. C. C. H. a. T. J. H. K. K. Tan, Advances in PID Control, London: Springer, 1999.

[11] T. Hirzel, "Arduino Docs - Basics of PWM (Pulse Width Modulation)," [Online]. Available: https://docs.arduino.cc/learn/microcontrollers/analog-output/.

[12] N. Z. S. Hylén, "Arduino Docs, Inter -Intergrated Circuit (I2C) Protocol," [Online]. Available: https://docs.arduino.cc/learn/communication/wire/.

[13] ATMEL, "8-bit AVR Microcontroller ATmega328P," 2009.

[14] 1.-1.-5. F. TO, AIRCRAFT WEIGHT AND BALANCE, 2015.

[15] Arduino, "Arduino Uno R3, Product Reference Manual," 2023.

[16] Arduino, "Arduino Nano, Product Reference Manual," 2024.

[17] TDK_Invensense, "MPU-6000 and MPU-6050 product Specification," August 2013. [Online]. Available: https://invensense.tdk.com/wp-content/uploads/2015/02/MPU-6000-Datasheet1.pdf.

[18] M. White, "The MPU6050 explained Programming Robots.," July 2019. [Online]. Available: https://mjwhite8119.github.io/Robots/mpu6050.

[19] Wikipedia, "Electronic speed control," 2023. [Online]. Available: https://en.wikipedia.org/wiki/Electronic_speed_control. [Accessed March 2023].

[20] Wikipedia, "Brushless DC electric motor," 2023. [Online]. Available: https://en.wikipedia.org/wiki/Brushless_DC_electric_motor. [Accessed May 2023].

[21] G. Staff, "How brushless motors work & how to test them," 2022. [Online]. Available: https://www.tytorobotics.com/blogs/articles/how-brushless-motors-work.. [Accessed May 2023].

[22] J. Reid, "C rating for Drone Lipo Battery Packs," 2019. [Online]. Available: https://www.rotordronepro.com/c-rating-drone-lipo-battery-packs. [Accessed May 2023].

[23] FLYSKY, "FS-i6, Digital Proportional Radio Control System - Instruction Manual," 2016.

[24] onsemi, "LM2596 - 3.0 A, step-down switching regulator," 2019. [Online]. Available: https://www.onsemi.com/pdf/datasheet/lm2596-d.pdf.

[25] AdvancedMonolithicSystems, "AMS1117, 1A LOW DROPOUT VOLTAGE REGULATOR".

[26] Qstcorp.com, "QMC5883L datasheet rev. A," [Online]. Available: https://qstcorp.com/upload/pdf/202202/13-52-04%20QMC5883L%20Datasheet%20Rev.%20A(1).pdf.

[27] ST.com, "A new generation, long distance ranging Time-of-Flight sensor," [Online]. Available: https://www.st.com/resource/en/datasheet/vl53l1x.pdf.

[28] NordicSemiconductors, "nRF24L01 Single Chip 2.4 GHz Tranceiver," 2007.

[29] R. Rozee, "HC-12 Wireless Serial Port Communication Module," 2016.

[30] Bosch, "Digital Pressure Sensor BMP280," 2021.

[31] HITACHI, "HD44780U (LCD-II)," 1996.

[32] 30A_ESC, "30A BLDC Electronic Speed Controller".

[33] S. O. Madgwick, "An efficient orientation filter for inertial and inertial/magnetic sensor arrays," 2010.