ΠΑΝΕΠΙΣΤΗΜΙΟ ΔΥΤΙΚΗΣ ΑΤΤΙΚΗΣ

ΣΧΟΛΗ ΜΗΧΑΝΙΚΩΝ

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΒΙΟΪΑΤΡΙΚΗΣ

Κατασκευή ρομποτικού οχήματος με δυνατότητα αναγνώρισης και αποφυγής εμποδίων

Ευθύμιος Κόλλιας
Αριθμός Μητρώου: 48015050

Επιβλέπων Καθηγητής
Παντελεήμων Ασβεστάς, Καθηγητής

Αθήνα 2024

Η Τριμελής Εξεταστική Επιτροπή

Ο Επιβλέπων Καθηγητής		
Παντελεήμων Ασβεστάς	Δημήτριος Γκλώτσος	Σπυρίδων Κωστόπουλος
Καθηγητής	Καθηγητής	Αναπληρωτής Καθηγητής
[ΨΗΦΙΑΚΗ ΥΠΟΓΡΑΦΗ]	[ΨΗΦΙΑΚΗ ΥΠΟΓΡΑΦΗ]	[ΨΗΦΙΑΚΗ ΥΠΟΓΡΑΦΗ]
Digitally signed by PANTELEIMON ASVESTAS Date: 2024.10.12 15:53:16 +03'00'	Digitally signed by DIMITRIOS GKLOTSOS Date: 2024.10.12 17:59:46 +03'00'	Digitally signed by Spyridon Kostopoulos Date: 2024.10.12 18:46:13 +03'00'

ΔΗΛΩΣΗ ΣΥΓΓΡΑΦΕΑ ΔΙΠΛΩΜΑΤΙΚΗΣ ΕΡΓΑΣΙΑΣ

Ο υπογράφων Ευθύμιος Κόλλιας του Γεωργίου, με αριθμό μητρώου 48015050 φοιτητής του Τμήματος Μηχανικών Βιοϊατρικής της Σχολής Μηχανικών του Πανεπιστημίου Δυτικής Αττικής, δηλώνω υπεύθυνα ότι:

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

Παράβαση της ανωτέρω ακαδημαϊκής μου ευθύνης αποτελεί ουσιώδη λόγο για την ανάκληση του διπλώματός μου».

Ημερομηνία

9/9/2024

Ο Δηλών

ΠΕΡΙΛΗΨΗ

Η ραγδαία ανάπτυξη της ρομποτικής και η ενσωμάτωσή της στον ιατρικό χώρο σηματοδοτεί μια νέα εποχή, όπου τεχνολογία και επιστήμη συνυφαίνονται για να υλοποιήσουν ιδέες που κάποτε ανήκαν στη σφαίρα της φαντασίας, προσφέροντας απεριόριστες δυνατότητες στους ασθενείς.

Σε αυτό το πλαίσιο, η παρούσα διπλωματική εργασία στοχεύει στη δημιουργία ενός ρομποτικού οχήματος που μιμείται (ως ένα βαθμό) την ικανότητα ενός αυτόνομου κινούμενου ρομπότ (Autonomous Mobile Robot – AMR) για ομαλή και αποτελεσματική αποφυγή εμποδίων σε περιορισμένους ιατρικούς χώρους. Τα κύρια εξαρτήματα που αξιοποιήθηκαν για την κατασκευή του οχήματος περιλαμβάνουν το Arduino Uno, τον αισθητήρα υπερήχων HC-SR04, καθώς και αισθητήρες υπέρυθρων για την ακριβέστερη αποφυγή εμποδίων. Επιπλέον, ο αρχικός έλεγχος του ρομποτικού οχήματος πραγματοποιείται μέσω ασύρματης επικοινωνίας (Bluetooth), ενώ για τον προγραμματισμό του μικροελεγχτή χρησιμοποιήθηκε το προγραμματιστικό περιβάλλον Arduino IDE.

Έπειτα από το σχεδιασμό, την υλοποίηση και τις τροποποιήσεις που έγιναν, τα αποτελέσματα της κατασκευής είναι ικανοποιητικά, με την λειτουργία του οχήματος να έχει ελεγχθεί τόσο προς την σταθερότητα όσο και την αυτονομία του.

Λέξεις Κλειδιά: αυτόνομο κινούμενο ρομπότ, αναγνώριση εμποδίων, αποφυγή εμποδίων, Arduino Uno, HC-SR04, Bluetooth, Arduino IDE

ABSTRACT

The rapid development of robotics and its integration into the medical field marks a new era, where technology and science intertwine to implement ideas that once belonged to the realm of fantasy, offering unlimited possibilities for patients.
In such context, this thesis aims to create a robotic vehicle that mimics (to some extent) an Autonomous Mobile Robot's - AMRs ability to smoothly and efficiently avoid obstacles in confined medical spaces. The main components utilized to build the vehicle include the Arduino Uno, the HC-SR04 ultrasonic sensor, and infrared sensors for more accurate obstacle avoidance. In addition, the robotic vehicle's starting control is conducted via wireless communication (Bluetooth), while the Arduino IDE programming environment was utilized to program the microcontroller.
Following the design, implementation and modifications made, the results of the construction are satisfactory, with the operation of the vehicle having been tested both towards stability and autonomy.

Keywords: Autonomous Mobile Robot, obstacle sensing, obstacle avoidance, Arduino Uno, HC-SR04, Bluetooth, Arduino IDE

Ευχαριστίες:

Θα ήθελα να ευχαριστήσω θερμά τον επιβλέποντα καθηγητή μου, κύριο Παντελεήμων Ασβεστά για το ενδιαφέρον και τη βοήθεια που μου πρόσφερε για την εκπόνηση της εργασίας καθώς και όλους τους καθηγητές του τμήματος για τις γνώσεις που μοίρασαν σε μένα και τους συμφοιτητές μου και μας οδήγησαν στο σήμερα. Ιδιαίτερα θα ήθελα να ευχαριστήσω την οικογένεια μου και συγκεκριμένα τους γονείς μου για την ψυχολογική και οικονομική υποστήριξη που μου παρείχαν κατά τη διάρκεια εκπόνησης της διπλωματικής μου εργασίας αλλά και κατά τη διάρκεια ολόκληρης της ακαδημαϊκής μου πορείας.

Περιεχόμενα

Κατάλογος εικόνων

Ι ΕΙΣΑΓΩΓΗ ΚΑΙ ΘΕΩΡΗΤΙΚΟ ΥΠΟΒΑΘΡΟ

ΙΙ ΥΛΙΚΑ, ΜΕΘΟΔΟΛΟΓΙΑ ΚΑΙ ΑΠΟΤΕΛΕΣΜΑΤΑ

ΙΙΙ ΣΥΜΠΕΡΑΣΜΑΤΑ, ΜΕΛΛΟΝΤΙΚΕΣ ΕΠΕΚΤΑΣΕΙΣ, ΒΙΒΛΙΟΓΡΑΦΙΑ ΚΑΙ ΠΑΡΑΡΤΗΜΑΤΑ

Κατάλογος Εικόνων

1. Εισαγωγή

1.1 Υπόβαθρο

Σε ένα γενικότερο πλαίσιο, τα όρια της ρομποτικής είναι όχι μόνο ασαφή, αλλά και μεταβαλλόμενα. Ως μια επιστήμη που βασίζεται στην τεχνολογική ανάπτυξη και κυρίως στην ανθρώπινη ικανότητα να εφαρμόζει αυτή την ανάπτυξη σε πρακτικό επίπεδο, η ρομποτική δεν παύει να αποκτά νέες διαστάσεις και να υποβοηθά όλο και πιο ουσιαστικά την καθημερινότητά μας.

Αν και παρατηρούνται διαφοροποιήσεις σχετικά με τον ακριβή ορισμό της, είναι κοινώς αποδεκτό ότι η ρομποτική αποτελεί κλάδο της μηχανοηλεκτρικής επιστήμης. Ως αντικείμενο μελέτης κρίνεται το θεωρητικό πλαίσιο, ο σχεδιασμός, η δημιουργία και ο προγραμματισμός αυτόνομων ή ημιαυτόνομων μηχανών (robot), με σκοπό την εξυπηρέτηση ποικίλων τομέων.

Παρότι η απόλυτη ταξινόμηση των ρομποτικών μηχανημάτων καθίσταται αδύνατη, κυρίως λόγω της πολυπλοκότητας τους τόσο σε μορφή όσο και σε λειτουργία, θα μπορούσαν να χωριστούν σε δύο γενικές κατηγορίες:

- **Βιομηχανικά ρομπότ.** Ως βιομηχανικό ρομπότ ορίζεται από την RIA Robotic Industries Association «ένας επαναπρογραμματίσιμος, πολυλειτουργικός χειριστής σχεδιασμένος να μετακινεί υλικά μέσω ποικίλων προγραμματισμένων κινήσεων για την εκτέλεση διαφόρων εργασιών». Η λειτουργία τέτοιων ρομπότ προορίζεται για εργασία γενικής χρήσης, με μερικώς ειδικευμένο ή και ανιδείκευτο εργατικό δυναμικό. Αποτελούν ιδανικούς βοηθούς σε βιομηχανικούς χώρους λόγω της ικανότητάς τους να εκτελούν αυτοματοποιημένες, επαναλαμβανόμενες και ακριβείς κινήσεις, αντικαθιστώντας έτσι το ανθρώπινο χέρι. Παραδείγματα χρήσης βιομηχανικών ρομπότ μπορούν να παρατηρηθούν σε εργασίες συγκόλλησης, βαφής, μηχανικής κατεργασίας κτλ.
- **Μη βιομηχανικά ρομπότ / Ρομπότ ειδικής χρήσης.** Ρομπότ ειδικής χρήσης θεωρούνται όλα τα ρομποτικά μηχανήματα που είναι σχεδιασμένα για να επιτελούν μια συγκεκριμένη λειτουργία και τυπικά απουσιάζουν από το βιομηχανικό περιβάλλον. Τέτοια ρομπότ μπορούν να χρησιμοποιούνται για την εξερεύνηση του διαστήματος, την εκτέλεση γεωργικών εργασιών και φυσικά, στον χώρο της ιατρικής (ρομποτικά οχήματα, χειρουργικά ρομπότ κτλ) [1].

Μία επιπρόσθετη ταξινόμηση θα μπορούσε να γίνει και για τα κινητά ρομπότ, τα οποία κυρίως διαχωρίζονται με βάση τον τύπο κίνησης που εκτελούν, το περιβάλλον στο οποίο αξιοποιούνται και ποικίλα επιμέρους χαρακτηριστικά. Τρεις είναι οι κατηγορίες κινητών ρομπότ:

- **Τροχήλατα κινητά ρομπότ (WMR).** Εκλαμβάνονται ως ο κλασσικός τύπος κινητού ρομπότ. Αποτελούν έναν ιδιαίτερα διαδεδόμενο τύπο για το είδος τους και παρουσιάζουν πολλά πλεονεκτήματα, καθώς συνδυάζουν απλή δομή και ενεργειακή αποδοτικότητα, ενώ διακρίνονται για την υψηλή ταχύτητα και το χαμηλό κόστος παραγωγής τους.
- **Υποβρύχια ρομπότ (UUV).** Τα ρομπότ αυτά χρησιμοποιούνται κυρίως για την διερεύνηση περιβαλλοντικών ζητημάτων και για την καταπολέμηση της ρύπανσης των υδάτων. Η μεταφορά των δεδομένων στα ρομπότ πραγματοποιείται συνήθως

μέσω καλωδίου, ενώ η ικανότητά τους να λειτουργούν και εξ αποστάσεως τα εντάσσει στην υποκατηγορία των τηλεχειριζόμενων οχημάτων (ROV).

- **Εναέρια ρομπότ (UAV).** Τα εναέρια ρομπότ χαρακτηρίζονται από αυτονομία μεγαλύτερη των υποβρύχιων ρομπότ, δεδομένου ότι η ύπαρξη εξωτερικού καλωδίου για τον έλεγχό τους δεν υφίσταται. Αποτελούν ιδιαίτερα πολύπλοκα ρομπότ, αφού καλούνται να διατηρήσουν ένα αποτελεσματικό σύστημα πλοήγησης ενώ βρίσκονται στον αέρα. Η δυνατότητά τους να διανύουν μακρινές αποστάσεις τα καθιστά ιδανικά για την μεταφορά φορτίων [2].

Τα τελευταία χρόνια έχει καταβληθεί σημαντική προσπάθεια για την αναβάθμιση των αυτόνομων κινητών ρομπότ (Autonomous Mobile Robots ή AMRs) και την προσαρμογή τους στα τεχνολογικά δεδομένα της εποχής. Στην πραγματικότητα, οι τομείς στους οποίους παρατηρείται πρόοδος αφορούν κυρίως την αποτελεσματικότερη λειτουργία τους υπό αντίξοες συνθήκες. Για παράδειγμα, ένα ρομποτικό όχημα μπορεί να προσαρμοστεί σε τυχόν ανωμαλίες του εδάφους, να αναπτύξει μεγάλες ταχύτητες και να εκτελέσει απότομους ελιγμούς, εάν αυτό απαιτείται [3]. Η ικανότητά του να αναγνωρίζει το περιβάλλον στο οποίο τοποθετείται και να λαμβάνει αποφάσεις για την κίνησή του στον χώρο, βασίζεται στο ενσωματωμένο σύστημα προγραμματισμού του.

Τα υψηλά ποσοστά ευελιξίας και προσβασιμότητας που χαρακτηρίζουν τα AMRs τα καθιστούν ιδανικά για την παροχή υπηρεσιών σε ιατρικούς χώρους. Το μέγεθός τους επιτρέπει την αποτελεσματική μεταφορά οποιουδήποτε φορτίου (χαμηλότερου όγκου), ιδιαίτερα στους πολυσύχναστους νοσοκομειακούς διαδρόμους. Επιπρόσθετα, έχουν την δυνατότητα να κινούνται σε στενότερους χώρους από ότι το ιατρικό προσωπικό και να συνεργάζονται μαζί του, εξασφαλίζοντας την φροντίδα των ασθενών και την εκπλήρωση των όποιων επιθυμιών τους. Παρά την διαδεδομένη χρήση των AMRs σε πολλούς τομείς της παραγωγής, δεν έχει αξιοποιηθεί το πλήρες δυναμικό τους και στο ιατρικό περιβάλλον [4].

Αναφορικά με τα αυτόνομα καθοδηγούμενα οχήματα (Autonomous Guided Vehicles ή AGVs), η διαδρομή και οι κινήσεις που μπορούν να πραγματοποιήσουν είναι πιο περιορισμένες σε σύγκριση με τα AMRs. Η χρήση τους είναι ιδιαίτερα συχνή σε διάφορους τομείς της βιομηχανίας, σε εργοστάσια μεταποίησης και στον χειρισμό υλικών γενικότερα [5]. Λειτουργούν αποτελεσματικά σε παραδοσιακά επιχειρηματικά μοντέλα, όπου δεν απαιτείται τόσο η προσαρμογή στο εργασιακό περιβάλλον όσο η επαναλαμβανόμενη κίνηση. Εφόσον χρησιμοποιούνται σε χώρους όπου το περιβάλλον παραγωγής παραμένει σχετικά αμετάβλητο, δύνανται να οδηγήσουν σε βελτιωμένη παραγωγικότητα και μεγαλύτερη ασφάλεια για τους εργάτες [5]. Αν και τα AGVs προηγήθηκαν των AMRs χρονικά, λόγω της αδυναμίας τους να προσαρμόζονται, δεν μπορούν να εφαρμοστούν στους περισσότερους ευέλικτους χώρους εργασίας, γι' αυτό και προτιμούνται τα AMRs.

Φυσικά, τα περισσότερα αυτόνομα ρομπότ χρησιμοποιούν αισθητήρες για την αναγνώριση του περιβάλλοντος και την προσπέραση εμποδίων. Κατασκευάζονται βασιζόμενα στις ανθρώπινες αισθήσεις (όραση, ακοή, θερμοκρασία) και παρέχουν πληροφορίες απαραίτητες για την αποτελεσματική λειτουργία του ρομπότ, όπως απόσταση, ταχύτητα και βάρος [6].

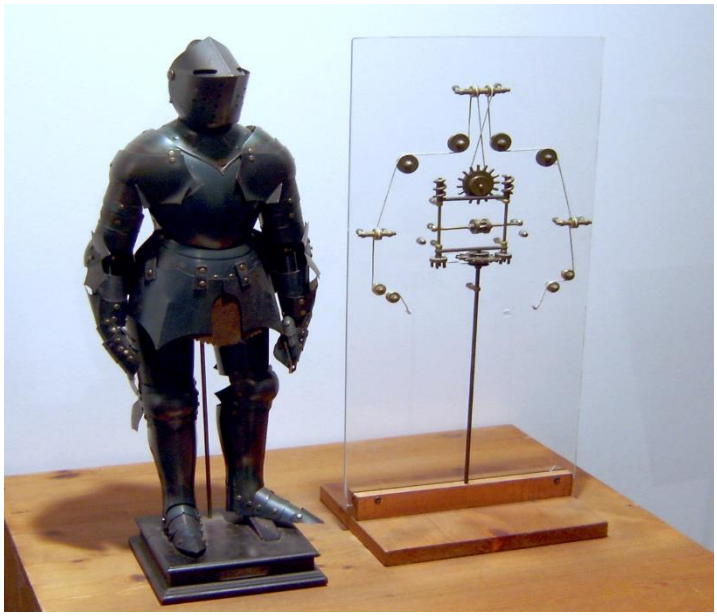
1.2 Ιστορική Αναδρομή

384 – 322 π.Χ.: Στα *Μηχανικά* του Αριστοτέλη, γίνεται αναφορά σε ένα μεγάλο εύρος μηχανισμών που απαρτίζουν τις σημερινές μηχανές, όπως τροχαλίες, γρανάζια και μοχλοί.

287 – 212 π.Χ.: Ο Αρχιμήδης κατασκευάζει τον ατέρμονα κοχλία, έναν μηχανισμό που επιτρέπει την παραγωγή μεγαλύτερης δύναμης από κινητήρες που θα περιοριζόνταν σε ισχύ λόγω του μικρού τους μεγέθους.

1206: Ο Ismail Al-Jazari γράφει στα περσικά το *Βιβλίο της Γνώσης των Ευφυών Μηχανικών Συσκευών*, όπου αναπαριστούνται διάφορα είδη υδροκίνητων και αεροκίνητων μηχανών.

1495: Ο Λεονάρντο Ντα Βίντσι σχεδιάζει και κατασκεύαζει εκείνο που για πολλούς θεωρείται ως το πρώτο ρόμποτ στην ιστορία, με αυτή την πρωταρχική μορφή να αποτελεί έναν ιππότη-ρομπότ με πανωπλία (Εικόνα 1). Μέσα από τροχαλίες και καλώδια, το ρομπότ είχε την δυνατότητα να κάθεται και να κινεί μέρη του σώματός του, με την τεχνική αυτή του Ντα Βίντσι να αξιοποιείται ακόμα στα σημερινά ανθρωποειδή ρομπότ (με ελάχιστες τροποποιήσεις).

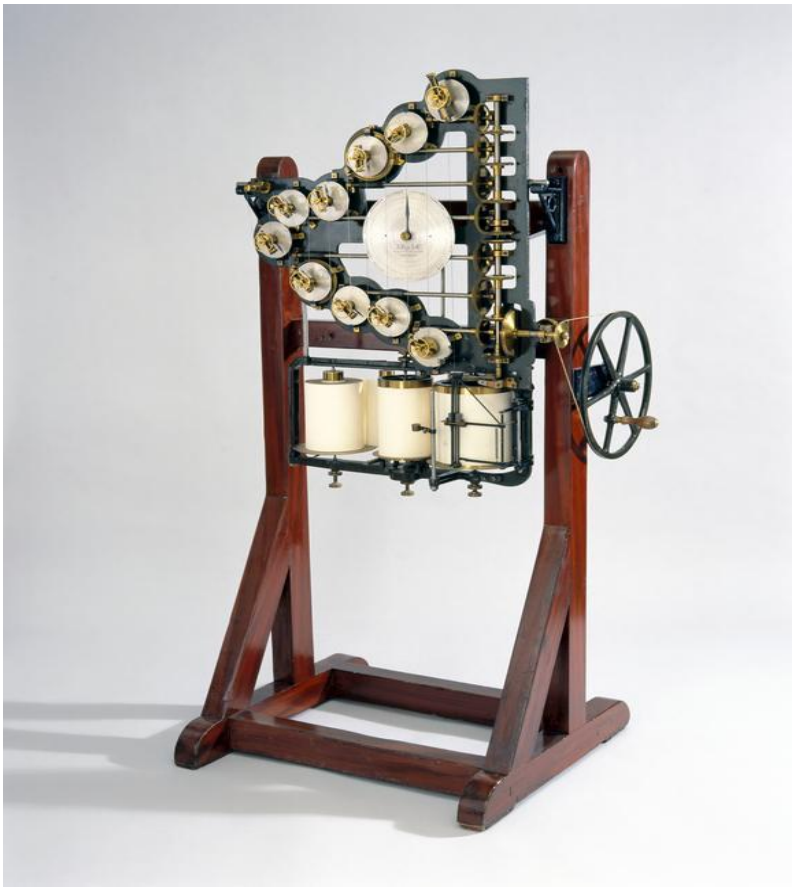


Εικόνα 1. *Ο μηχανικός ιππότης του Leonardo Da Vinci[I]*

1642: Η πρώτη υπολογιστική μηχανή κατασκευάζεται από τον Μπλεζ Πασκάλ, ικανή να πραγματοποιεί μαθηματικούς υπολογισμούς.

1805: Ο Ζοζέφ-Μαρί Ζακάρ δημιουργεί την πρώτη προγραμματίσιμη αυτόματη μηχανή παραγωγής. Αξιοποιώντας έναν παραδοσιακό αργαλειό, μια σειρά από κάρτες με τρύπες ελέγχονταν και παρήγαγαν μοτίβα, με αποτέλεσμα η τρύπα ή η έλλειψη τρύπας να αναγνωρίζεται ως “on” και “off”. Το ίδιο αυτό σύστημα χρησιμοποιείται και για τον προγραμματισμό των σημερινών υπολογιστών.

1877: Στην προσπάθεια του να προβλέπει τις παλίρροιες της θάλασσας, ο ιστιοπλόος Ουίλιαμ Τόμσον (γνωστός και ως Λόρδος Κέλβιν), κατασκευάζει τον πρώτο πρακτικό υπολογιστή (Εικόνα 2). Η μηχανή υπολόγιζε με ακρίβεια όλες τις παλίρροιες του χρόνου μέσα σε τέσσερις ώρες, πραγματοποιώντας έτσι πολύπλοκους υπολογισμούς που θα απαιτούσαν πολύ περισσότερο χρόνο να ολοκληρωθούν με την ανθρώπινη νόηση.



Εικόνα 2. *Η μηχανή πρόβλεψης της παλίρροιας του William Thomson, 1872 [II]*

1921: Ο όρος «ρομπότ» πρωτοεμφανίζεται στο θεατρικό έργο *Τα διεθνή ρομπότ του Ρόσσουμ* του Τσέχου συγγραφέα Κάρελ Τσάπεκ. Σύμφωνα με την πλόκη, ένας κατασκευαστής μηχανικών πλασμάτων προχωρά στην κατασκευή ενός όντος ανώτερου από τον άνθρωπο, με σκοπό την αντικατάσταση των ανθρώπινων εργατών. Ως όντα προορισμένα για την εργασία και χωρίς κανένα ίχνος συναισθήματος, ονομάστηκαν με το τσέχικο "robota", του οποίου η μετάφραση παραπέμπει στην εξαναγκασμένη εργασία. Ο επίλογος του έργου εκφράζει και την δημοφιλή αντίληψη που κυριαρχεί μεταξύ του ευρύτερου πληθυσμού περί ενδεχόμενης υποταγής του ανθρώπου στα ρομπότ. Τα ρομπότ στα οποία αναφέρεται ο συγγραφέας καταλήγουν να εναντιώνονται στους ανωτέρους τους και εν τέλει να κυριαρχούν, εξολοθρεύοντας την ανθρωπότητα [1].

Δεκαετία του 1940: Οι Νόρμπερτ Βίνερ και Τζούλιαν Μπίγκελοου (μαθηματικός και μηχανικός αντίστοιχα) συνεργάζονται για την κατασκευή ενός συστήματος ελέγχου που θα εφαρμοζόταν σε αντιαεροπορικά όπλα. Κατανόησαν ότι η αποτελεσματική λειτουργία τους θα ήταν εφικτή μόνο εάν τα όπλα λάμβαναν κάποιου είδους ανατροφοδότηση σχετικά με την κίνηση που πραγματοποιούσαν τα αεροσκάφη. Σύντομα ο Βίνερ συμπέρανε πως για την λειτουργία των μηχανών, αισθητήρες που εφαρμόζουν διάφορες βιολογικές και κοινωνικές λειτουργίες (όραση), μπορούν να αξιοποιηθούν για την μετάδοση των κατάλληλων πληροφοριών. Οι ιδέες του αποτέλεσαν την βάση για την Κηβερνητική (τομέας μελέτης) και έθεσαν τα θεμέλια για την τεχνητή νοημοσύνη και την σύγχρονη ρομποτική επιστήμη.

1942: Ο Ισαάκ Ασίμωφ συμβάλλει στην γενικότερη ανάδειξη της ρομποτικής μέσα από μια σειρά σύντομων ιστορίων και σε αντίθεση με τον Κάρελ Τσάπεκ, οραματίζεται ένα ρομπότ-βοηθό για το ανθρώπινο γένος. Έτσι, διατυπώνει τρεις νόμους που υποχρεωτικά χαρακτηρίζουν την συμπεριφορά των ρομπότ (εισάγοντας έτσι και τον όρο της Ρομποτικής):

1. Ένα ρομπότ δεν πρέπει να προκαλέσει κακό σε άνθρωπο ή μέσω της αδράνειάς του να επιτρέψει να τραυματιστεί άνθρωπος.
2. Ένα ρομπότ πρέπει να υπακούει στις εντολές των ανθρώπων, εκτός αν αυτό έρχεται σε αντιπαράθεση με τον πρώτο νόμο.
3. Ένα ρομπότ πρέπει να προστατεύει τον εαυτό του, εκτός αν αυτό έρχεται σε αντιπαράθεση με τον πρώτο και τον δεύτερο νόμο [1].

1945: Έπειτα από πολλές αποτυχημένες προσπάθειες, ο υπολογιστής ENIAC (Electronic Numerical Integrator and Computer) δημιουργείται από τον μεταπτυχιακό φοιτητή Τζον Έκερτ και τον καθηγητή φυσικής Τζον Μόχλι. Η επιτυχία της εφεύρεσης ορίζεται μέσα από την ικανότητά της να αποκρυπτογραφήσει τον μυστικό κώδικα Enigma που χρησιμοποιούσαν οι γερμανικές δυνάμεις για την μετάδοση μηνυμάτων κατά τον Β' Παγκόσμιο Πόλεμο. Αντί απλώς να πραγματοποιεί ταχείς μαθηματικούς υπολογισμούς, ο υπολογιστής αυτός μπορούσε να εκτελέσει πολλαπλές εργασίες αναλόγα με το αποτέλεσμα της πράξης, αναβαθμίζοντας έτσι την ικανότητα των υπολογιστών ως προς την επεξεργασία δεδομένων.

1954: Ο Τζόρτζ Ντεβόλ κατασκευάζει το πρώτο ηλεκτρονικά ελεγχόμενο ρομποτικό χέρι.

1974: Το εργαστήριο τεχνητής νοημοσύνης του Στάνφορντ δημιουργεί το πρώτο κινητό ρομπότ. Ο Shakey, όπως ονομάστηκε, αποτελούσε ένα τροχήλατο ρομπότ εφοδιασμένο με μια βιντεοκάμερα και διάφορους αισθητήρες, το οποίο κατόρθωσε να κινηθεί ανεξάρτητα σε έναν μικρό χώρο.

1994: Η ομάδα του Γουίλιαμ «Ρεντ» Γουιτάκερ σχεδιάζει το Dante II (Εικόνα 3), ένα εξάποδο ρομπότ που χρησιμοποιείται για την εξερεύνηση του όρους Spurr, ενός στρωματοηφαιστείου στην Αλάσκα [8].

Εικόνα 3. *Το "Dante II" στο όρος Spurr της Αλάσκας[III]*

1.3 Κίνητρο και στόχοι

Η ιδέα της υλοποίησης ενός «έξυπνου» ρομποτικού οχήματος δεν είναι νέα αλλά προϋπήρχε, ήδη από τη δεκαετία του 1950. Αν και αρχικά τα οχήματα αυτά δεν ήταν σχεδιασμένα για περιβάλλον υγειονομικής περίθαλψης, δεν άργησαν να υιοθετηθούν από τον τομέα της υγείας, όπου έχουν γνωρίσει ευρεία ανάπτυξη σε νεότερα χρόνια λόγω της αυξημένης ζήτησης εύρεσης τρόπων για καλύτερο έλεγχο ιατρικού κόστους, λειτουργικής αποτελεσματικότητας αλλά και βελτιστοποιημένης ροής εργασίας [9,10].

Η εξέλιξη της τεχνολογίας δημιουργεί όλο και υψηλότερα πρότυπα (standards) στη πρόοδο τέτοιων οχημάτων, με την πραγματική αυτονομία να βρίσκεται πιο κοντά από ποτέ. Όμως, για να υπάρξει τέτοια ελευθερία στις κινήσεις του οχήματος, αρχικά θα πρέπει (αν όχι να τελειοποιηθεί) να βελτιστοποιηθεί η χαρτογράφηση του περιβάλλοντος στο οποίο θα βρίσκεται το όχημα, η αλληλεπίδρασή του με αυτό καθώς και η επικοινωνία και απόκριση μαζί του. Η διπλωματική αυτή θα εξετάσει τα ζητήματα σχετικά με τη χρήση ειδικών αισθητήρων για την αποφυγή εμποδίων καθώς και με τη χρήση Bluetooth για την επικοινωνία του χρήστη με το όχημα.

Ο στόχος της διπλωματικής είναι η υλοποίηση ενός ρομποτικού οχήματος το οποίο θα μπορεί να εντοπίζει και να αποφεύγει αντικείμενα-εμπόδια, με την εκκίνηση του οχήματος να πραγματοποιείται με τη χρήση ασύρματης επικοινωνίας μέσω smartphone. Η νέα κατεύθυνση θα γίνεται χωρίς την επιρροή του χρήστη, αφού θα καθορίζεται από το ίδιο το όχημα μέσω ειδικών αισθητήρων.

Τα ερωτήματα που θα πρέπει να απαντηθούν είναι τα εξής:

- Πόσο ομαλά γίνεται ο εντοπισμός και η αποφυγή των εμποδίων από το όχημα;
- Πόσο σύντομο είναι το διάστημα που μεσολαβεί από τη στιγμή που δίνεται η φωνητική οδηγία από τον χρήστη μέχρι τη στιγμή που ξεκινά να κινείται το όχημα;
- Πέρα από την αρχική λήψη οδηγιών, κατάφερε το όχημα να κινηθεί αυτόνομα σε ένα άγνωστο για αυτό περιβάλλον, χωρίς περαιτέρω οδηγίες;
- Πόσο σταθερή είναι η λειτουργία του οχήματος;

2. Θεωρητικό υπόβαθρο

Σε αυτό το κεφάλαιο παρέχεται απαραίτητο υπόβαθρο που βοηθά στη κατανόηση των υλικών και της μεθοδολογίας που χρησιμοποιήθηκαν για τη παρούσα διπλωματική. Στο 2.1 γίνεται μια εισαγωγή στη κατασκευή του οχήματος, στα υλικά και στους αλγόριθμους που χρησιμοποιούνται ευρέως και στις εφαρμογές του στο βιοϊατρικό πεδίο. Στο 2.2 γίνεται μια σφαιρική αναφορά πάνω στο Arduino, στο 2.3 πάνω στους κινητήρες και τον έλεγχο τους, στο 2.4 πάνω στους αισθητήρες, στο 2.5 παρέχεται υπόβαθρο πάνω στο περιβάλλον ανάπτυξης προγράμματος και στο 2.6 στους τύπους επικοινωνίας που χρησιμοποιούνται.

2.1 Εισαγωγή

Η ρομποτική είναι ένας διεπιστημονικός τομέας που συνδυάζει τη μηχανολογία, την ηλεκτρολογία, την επιστήμη των υπολογιστών και την τεχνητή νοημοσύνη για τον σχεδιασμό, την κατασκευή και τον προγραμματισμό μηχανών ικανών να εκτελούν εργασίες αυτόνομα ή ημιαυτόνομα. Τα ρομπότ σχεδιάζονται για να αλληλεπιδρούν με το περιβάλλον, να αντιλαμβάνονται τυχόν αλλαγές και να ενεργούν ανάλογα, λαμβάνοντας συχνά αποφάσεις με βάση τα δεδομένα από διάφορους αισθητήρες. Τα συστήματα αυτά μπορεί να κυμαίνονται από απλές μηχανές που ακολουθούν προκαθορισμένες οδηγίες έως πολύπλοκα ρομπότ που χρησιμοποιούν τεχνητή νοημοσύνη για να μαθαίνουν και να προσαρμόζονται στο περιβάλλον τους.

Στη ρομποτική, τα σημαντικά στοιχεία είναι δύο: η μηχανική δομή (η οποία περιλαμβάνει κινητήρες και ενεργοποιητές για να καταστεί δυνατή η κίνηση) και το σύστημα ελέγχου (το οποίο αποτελείται από αισθητήρες, μικροελεγκτές και λογισμικό για την καθοδήγηση των ενεργειών). Στην περίπτωση ενός οχήματος αποφυγής εμποδίων, όπως αυτό που θα υλοποιηθεί, το μηχανικό σύστημα του ρομπότ αποτελείται από κινητήρες συνεχούς ρεύματος που κινούν τους τροχούς, ενώ το σύστημα ελέγχου καθοδηγείται από ένα Arduino, το οποίο επεξεργάζεται δεδομένα από αισθητήρες όπως υπερήχων και υπέρυθρων για την ανίχνευση και την αποφυγή εμποδίων. Η ενσωμάτωση αυτών των εξαρτημάτων επιτρέπει στο ρομπότ να αντιλαμβάνεται το περιβάλλον του, να λαμβάνει αποφάσεις σε πραγματικό χρόνο και να προσαρμόζει τις κινήσεις του ανάλογα [11].

Στα συστήματα αποφυγής εμποδίων, οι αισθητήρες παίζουν καθοριστικό ρόλο στην αντίληψη του περιβάλλοντος. Οι αισθητήρες υπερήχων χρησιμοποιούνται ευρέως για τη μέτρηση της απόστασης με την εκπομπή ηχητικών κυμάτων και τον υπολογισμό του χρόνου που απαιτείται για την επιστροφή τους μετά την πρόσκρουσή τους σε ένα αντικείμενο και συνδυάζονται συχνά με έναν σερβοκινητήρα, έτσι ώστε το ρομπότ να σαρώνει μια ευρύτερη περιοχή μπροστά του. Επιπλέον, οι αισθητήρες υπέρυθρης ακτινοβολίας (IR) χρησιμοποιούνται ευρέως για την ανίχνευση αντικειμένων σε μικρότερες αποστάσεις μετρώντας το ανακλώμενο φως υπέρυθρης ακτινοβολίας και συχνά τοποθετούνται στις πλευρές του οχήματος για ευρύτερη κάλυψη. Η επιτυχής συνεργασία διαφόρων ειδών αισθητήρων διασφαλίζει τον έγκυρο εντοπισμό οποιουδήποτε εμποδίου σε οποιαδήποτε κατεύθυνση [12].

Το σημαντικότερο κομμάτι ενός συστήματος αποφυγής εμποδίων βρίσκεται στους αλγορίθμους που επεξεργάζονται τις εισόδους των αισθητήρων και αποφασίζουν τις επόμενες ενέργειες του. Μια συνηθισμένη αλγοριθμική προσέγγιση περιλαμβάνει την ανάγνωση δεδομένων απόστασης από τον αισθητήρα υπερήχων και τον προσδιορισμό εάν ένα εμπόδιο βρίσκεται εντός μιας προκαθορισμένης εμβέλειας. Εάν εντοπιστεί αντικείμενο, το ρομπότ σταματά και το σύστημα ελέγχου μπορεί να ξεκινήσει μια ακολουθία ελιγμών με βάση τα δεδομένα που λαμβάνει από τον αισθητήρα. Το ρομπότ λαμβάνει συνήθως αποφάσεις με βάση τιμές κατωφλίου που έχουν προκαθοριστεί, τόσο από τους αισθητήρες υπερήχων όσο και από τους IR αισθητήρες. Με την ενσωμάτωση των δεδομένων των αισθητήρων και τη συνεχή ενημέρωση της κατάστασης του ρομπότ, επιτυγχάνεται ανταπόκριση σε πραγματικό χρόνο, εξασφαλίζοντας ομαλή και αποτελεσματική πλοήγηση. Αυτή η αλγοριθμική

προσέγγιση χρησιμοποιείται ευρέως στην αυτόνομη ρομποτική για εργασίες όπως ο σχεδιασμός διαδρομής και η δυναμική αποφυγή εμποδίων [13].

Η ρομποτική, ιδίως τα συστήματα που βασίζονται σε αισθητήρες, έχει σημαντικές εφαρμογές στο πεδίο της βιοϊατρικής. Τα αυτόνομα ρομποτικά συστήματα μπορούν να βοηθήσουν σε περιβάλλοντα υγειονομικής περίθαλψης, όπως ρομποτικά αναπηρικά αμαξίδια που περιφέρονται γύρω από εμπόδια στα νοσοκομεία ή βοηθητικά ρομπότ που βοηθούν ηλικιωμένα ή ανάπηρα άτομα να κινούνται ανεξάρτητα στα σπίτια τους.

Συγκεκριμένα, η διπλωματική εμπνεύστηκε από οχήματα που υπάρχουν σε νοσοκομεία και κλινικές που διαθέτουν τεχνολογία χαρτογράφησης και αποφυγής εμποδίων, με την χρήση τους να εξυπηρετεί στη μεταφορά φαρμάκων, αναλώσιμων, φαγητών και ρούχων, στο καθαρισμό πατωμάτων αλλά και στην απολύμανση-αντιμετώπιση ιών και βακτηρίων μέσω υπεριώδους ακτινοβολίας. Η άνοδος στη χρησιμότητα τους προέκυψε από την εξέλιξη στη τεχνολογία των αισθητήρων, την ανάγκη για μικρότερο φόρτο εργασίας του προσωπικού και την ανάγκη για ελαχιστοποίηση των νοσοκομειακών λοιμώξεων, μειώνοντας την ανθρώπινη επαφή σε ιατρικούς χώρους. Καθώς η βιοϊατρική ρομποτική εξελίσσεται, η ενσωμάτωση της πλοήγησης με βάση αισθητήρες συνεχίζει να διασφαλίζει την ακρίβεια και την ασφάλεια στις ιατρικές εφαρμογές [9, 10, 14].

2.2 Arduino

Βασισμένο στη χρήση υλικού (Hardware) και λογισμικού (Software), το Arduino αποτελεί μια ευρέως διαδεδομένη πλατφόρμα ηλεκτρονικών «ανοιχτού κώδικα» (open-source) και μετεξέλιξη της πλατφόρμας Wiring. Η ιδέα για την υλοποίησή του προήλθε από μια ομάδα φοιτητών στα πλαίσια ενός μεταπτυχιακού προγράμματος σπουδών στην Ιβρέα της Ιταλίας (τους Massimo Banzi, David Cuartielles, Tom Igoe, Gianluca Martino, και David Mellis). Σκοπός τους ήταν η κατασκευή ενός εργαλείου για αρχάριους, προχωρημένους αλλά και επαγγελματίες, το οποίο θα τους παρείχε την ευκαιρία δημιουργίας projects εκμάθησης ηλεκτρονικών συστημάτων, ή ακόμη και συσκευών που αλληλεπιδρούν με το περιβάλλον τους [16] .

Όπως προαναφέρθηκε, το Arduino αποτελείται από δύο μέρη. Πρακτικά, είναι μια προγραμματίσιμη ηλεκτρονική πλακέτα η οποία διαθέτει ενσωματωμένο μικροελεγκτή (ATmega της Atmel), συνοδεύομενο από αναλογικές και ψηφιακές εισόδους και εξόδους. Για το προγραμματιστικό σκέλος, το Arduino προσφέρει στο χρήστη ένα ενσωματωμένο περιβάλλον ανάπτυξης, το Arduino IDE, το οποίο αποτελεί συνέχεια του περιβάλλοντος ανάπτυξης Wiring. Με τη σειρά του, το Wiring βασίστηκε στο λογισμικό Processing, ένα προγραμματιστικό περιβάλλον βασισμένο στη γλώσσα Java. Εν ολίγοις, το Arduino επιτρέπει στο χρήστη να συνδέσει πάνω του πολλαπλές μονάδες εισόδου αλλά και εξόδου, να προγραμματίσει τον μικροελεγκτή προκειμένου να δέχεται δεδομένα από αυτές, να τις επεξεργάζεται και εν τέλει να στέλνει εντολές στις εξόδους του, λειτουργώντας έτσι σαν ένας μικροσκοπικός υπολογιστής.

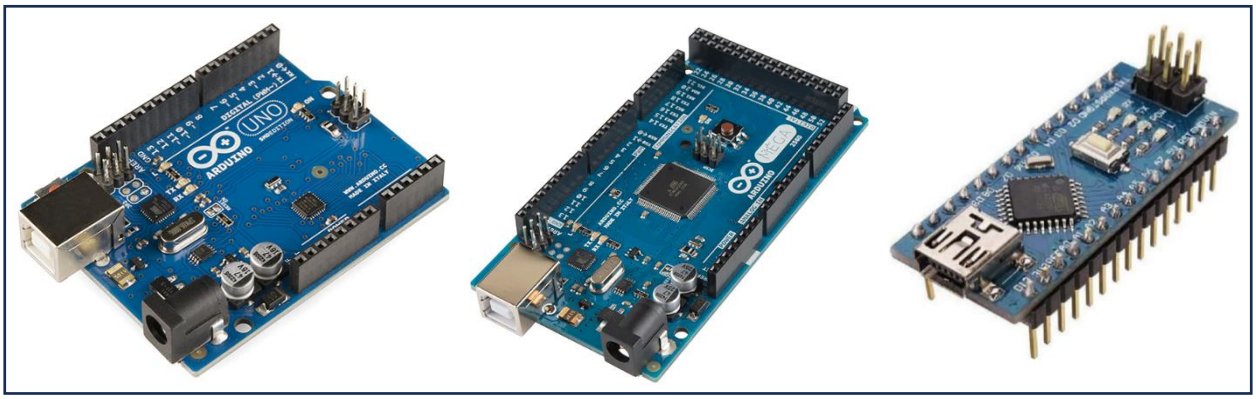
Φυσικά, δεν είναι η μόνη πλατφόρμα που κυκλοφορεί στην αγορά. Αρκετά χαρακτηριστικά της όμως την έχουν ξεχωρίσει απέναντι σε άλλες:

- Είναι αρκετά φθηνή, π.χ. το Arduino Uno κοστολογείται ≈ 25€ (Σεπτέμβριος 2023). Ακόμη και σε περίπτωση βλάβης, τα εκάστοτε εξαρτήματα είναι φθηνά και εύκολα αντικαταστήσιμα.
- Το περιβάλλον της μπορεί να «τρέξει» σε πολλαπλά λειτουργικά συστήματα, όπως τα Windows, Macintosh και Linux.
- Το υλικό και το λογισμικό είναι open-source, κάτι που δίνει τη δυνατότητα στο χρήστη να τα επεκτείνει ή να αναπτύξει όπως επιθυμεί, ή ακόμα και να

αγοράσει όλα τα εξαρτήματα μόνος του και να δημιουργήσει μια δική του πλατφόρμα.

- Το προγραμματιστικό της περιβάλλον είναι σχεδιασμένο έτσι ώστε να είναι απλό και κατανοητό σε αρχάριους.
- Διαθέτει μια σημαντικά ενεργή κοινότητα από χρήστες, οι οποίοι δύνανται να λύσουν οποιοδήποτε πρόβλημα ή ερώτημα προκύψει [15].

Τα τελευταία χρόνια έχουν αναπτυχθεί πολλές διαφορετικές εκδόσεις του Arduino, το καθένα με το δικό του μικροελεγκτή (Εικόνα 4). Κάποια από τα χαρακτηριστικά με τα οποία διαφοροποιούνται μεταξύ τους είναι ο αριθμός των εισόδων και εξόδων, η επεξεργαστική ταχύτητα, η τάση λειτουργίας και ο συντελεστής μορφής [16].



Εικόνα 4. *Από αριστερά προς τα δεξιά: Arduino UNO, Arduino Mega 2560 και Arduino Nano[IV]*

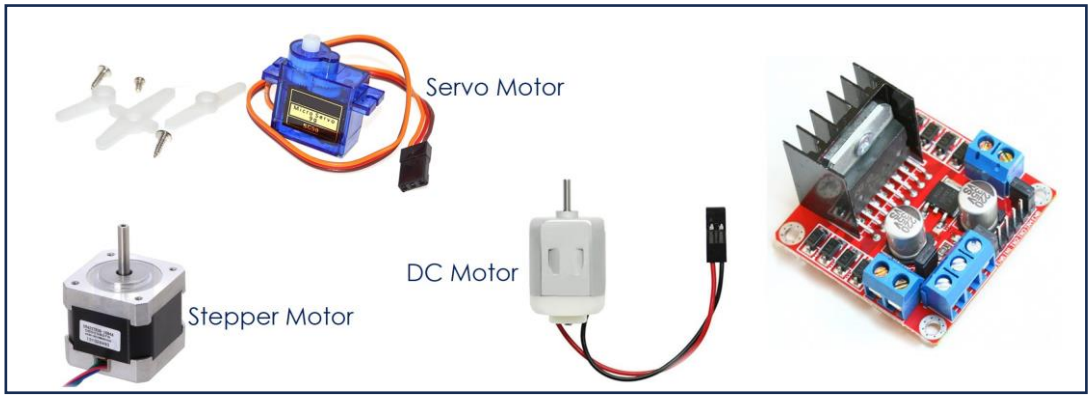
2.3 Κινητήρες και έλεγχος

Στο κόσμο της ρομποτικής και συγκεκριμένα των ρομποτικών οχημάτων, οι ηλεκτρικοί κινητήρες αποτελούν έναν από τους πιο διαδεδομένους ενεργοποιητές, υπεύθυνοι για την παραγωγή κίνησης, τον έλεγχο και την ταχύτητα των οχημάτων, καθιστώντας τους ως ευέλικτα εργαλεία σε πλήθος εφαρμογών. Οι 3 βασικές κατηγορίες κινητήρων είναι οι εξής:

- Κινητήρας συνεχούς τάσης (**DC motor**) (Εικόνα 5): Ο πιο συνηθισμένος κινητήρας σε εφαρμογές ρομποτικών οχημάτων. Αφού τροφοδοτηθεί και ενεργοποιηθεί, αρχίζει και περιστρέφεται, χωρίς να υπάρξει αλλαγή της κατεύθυνσης του ρεύματος λόγω ενός μεταγωγέα εκ κατασκευής του. Η αλλαγή της πολικότητας οδηγεί σε αλλαγή της κατευθύνσης της περιστροφής και η ταχύτητα είναι ανάλογη της τάσης που εφαρμόζεται.
- Σερβοκινητήρας (**Servomotor**) (Εικόνα 5): Χρησιμοποιείται σε ένα ευρύ φάσμα εφαρμογών της ρομποτικής, όπως σε ρομποτικά χέρια, ρομποτικούς βραχίονες και εφαρμογές προσδιορισμού θέσης μέσω ανιχνευτή, λόγω της ακρίβειας που προσφέρει στον έλεγχο της ταχύτητας και της γωνιακής ή γραμμικής θέσης. Έχει την ιδιότητα να περιστρέφει έναν άξονα έως 90°, 180° ή και 270° αναλόγα με το είδος του σερβοκινητήρα, αλλά με κατάλληλη μετατροπή μπορούν να ολοκληρώσουν περιστροφή ακόμη και 360°.
- Βηματικός κινητήρας (**Stepper motor**) (Εικόνα 5): Χρησιμοποιείται σε εφαρμογές που απαιτούν χαμηλή ταχύτητα και υψηλή ακρίβεια, όπως 3D εκτυπωτές και μηχανήματα ιατρικής απεικόνισης. Έχει τη δυνατότητα να περιστρέφεται 360° με συνεχή τρόπο κατά έναν συγκεκριμένο αριθμό μοιρών (βήμα) [17][18].

Ο έλεγχος των κινητήρων αποτελεί ένα σημαντικό μέρος στο σχεδιασμό και την υλοποίηση ρομποτικών οχημάτων, κάτι το οποίο συνήθως επιτυγχάνεται με τη χρήση ενός Arduino. Η συχνότητα με την οποία ‘’συνεργάζονται’’ είναι μεγάλη, κάτι το οποίο αντικατοπτρίζεται στη πληθώρα εφαρμογών που έχουν χρησιμοποιηθεί μαζί.

Στις περισσότερες από αυτές κρίνεται ακόμη απαραίτητη η χρήση και ενός οδηγού ελέγχου κινητήρων (Εικόνα 5), μιας προέκτασης ουσιαστικά του Arduino, η οποία δίνει στο χρήστη πλήρη έλεγχο της ταχύτητας και της περιστροφής των κινητήρων. Ταυτόχρονα, η εξωτερική τροφοδοσία του οδηγού προστατεύει το Arduino από τυχόν βλάβες καθώς οι κινητήρες, συγκεκριμένα οι κινητήρες συνεχούς τάσης, καταναλώνουν υψηλές τιμές ρεύματος με αποτέλεσμα να υπάρχει κίνδυνος βλάβης των ακίδων ή και ακόμη να καεί η πλακέτα [16].



Εικόνα 5. *Από αριστερά προς τα δεξιά:Βηματικός κινητήρας, σερβοκινητήρας, κινητήρας συνεχούς τάσης και οδηγός ελέγχου κινητήρων L298N[V]*

2.4 Αισθητήρες

Όλες οι ρομποτικές διατάξεις είναι συστήματα που έχουν κατασκευαστεί για να αντιλαμβάνονται το περιβάλλον τους, να εκτελούν υπολογισμούς και να λαμβάνουν αποφάσεις. Μια τέτοια διάταξη κρίνεται απαραίτητο να χρησιμοποιεί ρομποτικούς αισθητήρες, καθώς χωρίς αυτούς δε θα μπορούσε να γνωρίζει τι συμβαίνει γύρω της.

Οι αισθητήρες είναι συσκευές (συσκευές εισόδου συγκεκριμένα), οι οποίες είναι σχεδιασμένες να λαμβάνουν πληροφορίες από το περιβάλλον γύρω τους και να τις μετατρέπουν σε δεδομένα τα οποία μπορεί να ‘’διαβάζει’’ ένα ρομπότ. Υπάρχουν πολλοί τύποι αισθητήρων, όπως θερμοκρασίας, υγρασίας, ορατού φωτός, κίνησης κλπ., αλλά σε project ρομποτικών οχημάτων, όπως το παρών, συχνά χρησιμοποιούνται αισθητήρες απόστασης.

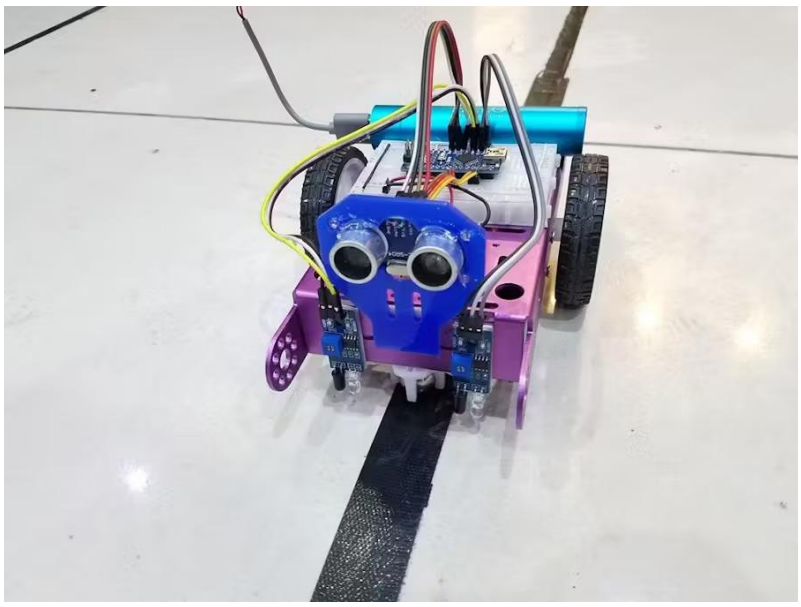
Η ικανότητα ενός ρομποτικού οχήματος να ανιχνεύει ένα αντικείμενο χωρίς να χρειαστεί να το αγγίξει αλλά και να προσδιορίζει τη θέση του από την απόσταση που έχει από τον αισθητήρα κρίνεται αρκετά χρήσιμη. Ένας αισθητήρας απόστασης συνήθως χρησιμοποιεί κύματα φωτός ή ηχητικά κύματα τα οποία ανακλώνται πάνω στο αντικείμενο ώστε να μετρηθεί η απόσταση του από τον αισθητήρα [17]. Δύο διαδεδομένοι τύποι αισθητήρων απόστασης είναι οι εξής:

- Αισθητήρας υπερήχων (**Ultrasonic Sensor**) (Εικόνα 6): Ο συγκεκριμένος αισθητήρας μετράει την απόσταση με ένα αντικείμενο χρησιμοποιώντας υπερηχητικά κύματα υψηλής συχνότητας, της τάξης των 40 kHz. Διαθέτει έναν ηλεκτροακουστικό μορφοτροπέα, ο οποίος συνήθως λειτουργεί ως εκπομπός αλλά και ως δέκτης του κύματος (κάποιοι αισθητήρες υπερήχων διαθέτουν και δεύτερο μορφοτροπέα που

λειτουργεί ως δέκτης). Η απόσταση υπολογίζεται μέσω του κυκλώματος του αισθητήρα, μετρώντας το χρόνο μεταξύ του κύματος και της ηχού. Τέτοιου είδους αισθητήρες αποτελούν καλή επιλογή για ανίχνευση αντικειμένων σε μεσαία ή μεγάλη απόσταση. Σε ρομποτικές διατάξεις με βάση τον μικροελεγκτή Arduino ένας πολύ δημοφιλής αισθητήρας υπερήχων είναι ο HC-SR04. Διαθέτει δύο μορφοτροπείς υπερήχων που δρουν ως εκπομπός και ως δέκτης αντίστοιχα [19].

- Αισθητήρας υπέρυθρων (**IR Sensor**) (Εικόνα 6): Αποτελείται από έναν εκπομπό (IR LED), ο οποίος εκπέμπει υπέρυθρο φως, και από μία φωτοδίοδο υπέρυθρων (IR Photodiode), η οποία λειτουργεί ως δέκτης υπέρυθρου φωτός. Με λίγα λόγια, η αρχή λειτουργίας του αισθητήρα είναι ένας οπτοπλέκτορας, δηλαδή ο συνδυασμός του εκπομπού και του δέκτη. Ο αισθητήρας μπορεί κ μετρά την απόσταση από μια επιφάνεια, από το χρόνο που κάνει το εκπεμπόμενο υπέρυθρο φως να επιστρέψει και εντοπιστεί από το δέκτη. Τέτοιου είδους αισθητήρες αποτελούν καλή επιλογή για ανίχνευση αντικειμένων σε μικρή ή μεσαία απόσταση. Χρησιμοποιούνται συχνά σε εφαρμογές ρομποτικών οχημάτων με δυνατότητα αποφυγής εμποδίων αλλά και σε εφαρμογές ρομποτικών οχημάτων προγραμματισμένα να ακολουθούν μια γραμμή (Εικόνα 7) [18, 19, 20].

Εικόνα 6. *Από αριστερά προς τα δεξιά: Αισθητήρας υπερήχων HC-SR04 και αισθητήρας υπέρυθρων[VI]*

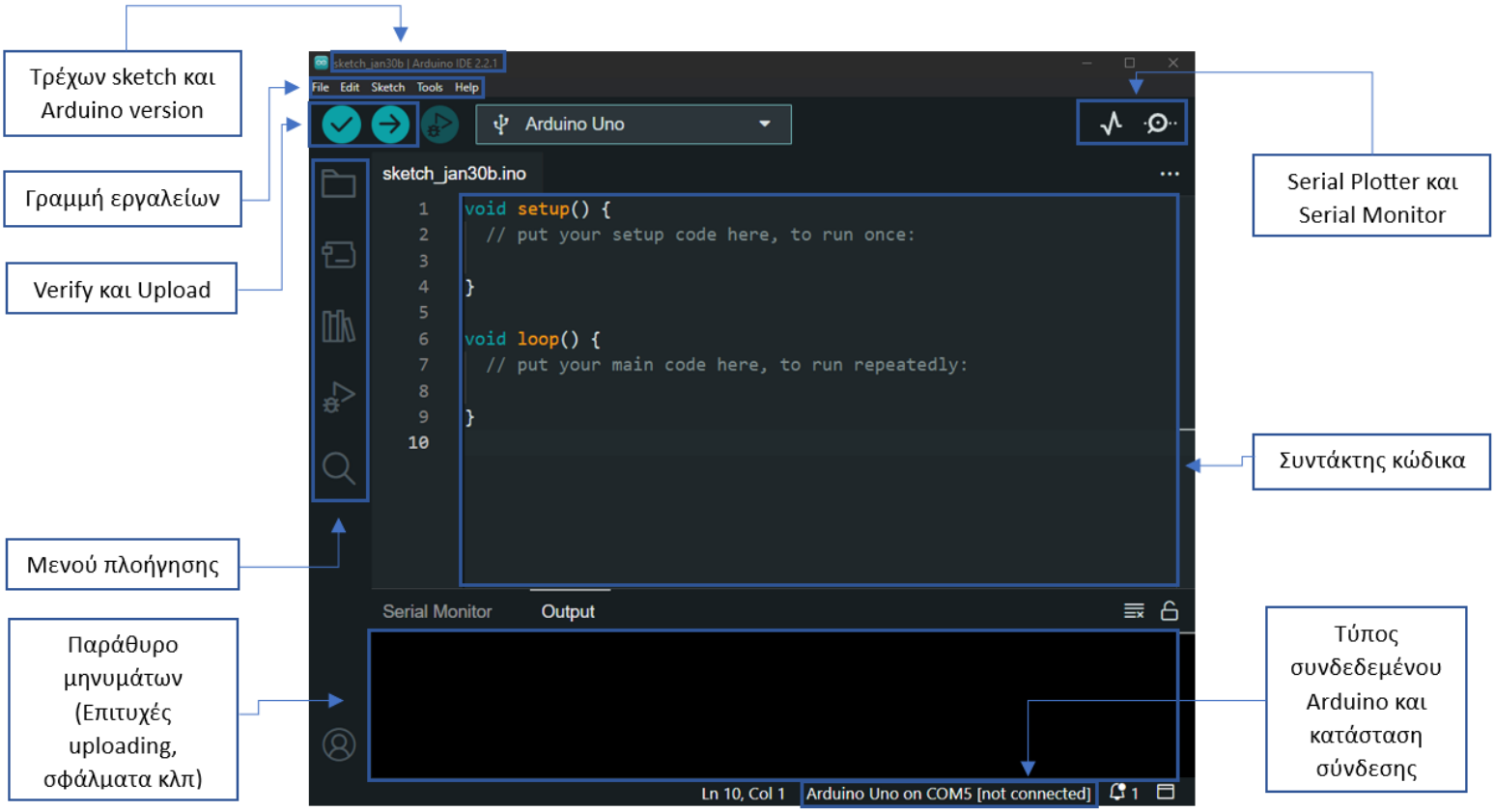


Εικόνα 7. *Εφαρμογή ρομποτικού οχήματος ακολούθησης γραμμής με χρήση HC-SR04 και IR Sensors[VII]*

2.5 Περιβάλλον ανάπτυξης προγράμματος

Όπως αναφέρθηκε και στο 2.1, το Arduino διαθέτει ενσωματωμένο περιβάλλον ανάπτυξης προγράμματος, το Arduino IDE (Εικόνα 8), το οποίο και θα χρησιμοποιηθεί στη παρούσα διπλωματική. Βέβαια, ο χρήστης έχει την επιλογή να χρησιμοποιήσει οποιοδήποτε περιβάλλον ανάπτυξης επιθυμεί και δεν περιορίζεται από το Arduino, αντιθέτως μπορεί να προγραμματιστεί με οποιαδήποτε γλώσσα μέσω μεταγλωττιστών (compilers) που θα παράγουν δυαδικό κώδικα [21].

Το Arduino IDE έχει σχεδιαστεί για να λειτουργεί σε πολλαπλά λειτουργικά συστήματα και είναι φιλόξενο προς το χρήστη, είτε έμπειρο στο προγραμματισμό είτε άπειρο. Έχει γραφτεί στη γλώσσα Java καθώς βασίζεται στο περιβάλλον ανάπτυξης Processing και διαθέτει βιβλιοθήκες οι οποίες είναι γραμμένες σε γλώσσες C και C++. Κατά τη διάρκεια της διαδικασίας της μεταμόρφωσης ενός κομματιού κώδικα (sketch) στο Arduino, ο κώδικας μεταφράζεται σε C και μεταφέρεται στο μεταγλωττιστή *avr-gcc*, ο οποίος εκτελεί την τελική μετάφραση σε δυαδικό κώδικα, δηλαδή γλώσσα την οποία μπορεί να αναγνωρίσει ο μικροελεγκτής [15] .



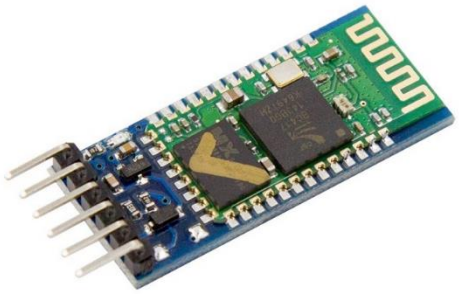
Εικόνα 8. *Περιβάλλον ανάπτυξης Arduino IDE (2.2.1 version).*

2.6 Επικοινωνία

Στη παρούσα διπλωματική χρησιμοποιούνται δύο τύποι επικοινωνίας, η σειριακή και η ασύρματη επικοινωνία.

Η σειριακή επικοινωνία είναι η διαδικασία κατά την οποία πραγματοποιείται αποστολή δεδομένων ενός δυαδικού αριθμού (ή bit) τη φορά, είτε μέσω ενός αγωγού, όπως ένα καλώδιο, είτε μεταξύ τμημάτων μιας υπολογιστικής μονάδας ή μεταξύ 2 υπολογιστικών μονάδων [21]. Η χρήση της στη διπλωματική είναι για τη σύνδεση του Arduino UNO με σταθερό υπολογιστή μέσω USB έτσι ώστε να προγραμματιστεί κατάλληλα αλλά και για αποσφαλμάτωση (debugging) χρησιμοποιώντας το Serial Monitor του Arduino IDE [19].

Η ασύρματη επικοινωνία είναι η διαδικασία κατά την οποία πραγματοποιείται αποστολή δεδομένων χωρίς τη χρήση κάποιου αγωγού ή συνεχούς μέσου, αλλά είτε συνήθως μέσω ραδιοκυμάτων είτε λιγότερο συχνά μέσω φαινόμενων ηλεκτρομαγνητισμού. Το Bluetooth είναι μια τεχνολογία ασύρματης επικοινωνίας που χρησιμοποιεί ραδιοκύματα χαμηλής ενέργειας, ιδανικό για μικρές αποστάσεις και ικανό για επικοινωνία μεταξύ δύο συσκευών, ανεξάρτητα από τον κατασκευαστή τους [21]. Στη διπλωματική θα χρησιμοποιηθεί το HC-05 (Εικόνα 9), ένα περιφερειακό (module) του Arduino το οποίο αξιοποιεί τη τεχνολογία Bluetooth και θα συνδέσει το ρομποτικό όχημα με το χρήστη, συγκεκριμένα το HC-05 θα συνδέσει το Arduino UNO με το smartphone του χρήστη.



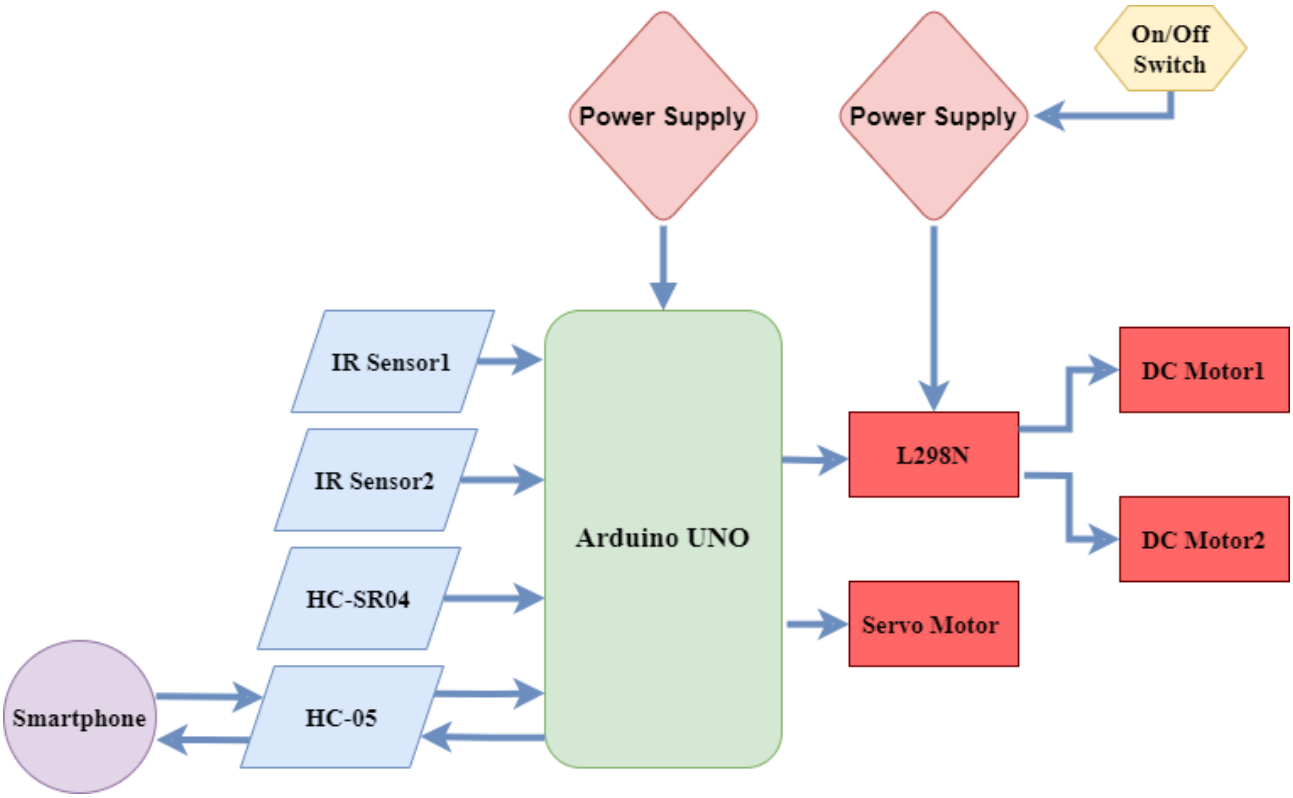
Εικόνα 9. *Bluetooth module HC-05[VIII]*

3. Σχεδιασμός και ανάπτυξη του ρομποτικού οχήματος

Σε αυτό το κεφάλαιο, στο 3.1 παρέχεται ένα διάγραμμα μπλοκ του οχήματος, στα 3.2 και 3.3 παρέχεται μια λίστα του Hardware και του Software που χρησιμοποιήθηκαν, στο 3.4 περιγράφεται ο σχεδιασμός του οχήματος, στο 3.5 η απόκτηση των υλικών, στο 3.6 η συναρμολόγηση του και τέλος στο 3.7 η συνδεσμολογία του.

3.1 Διάγραμμα μπλοκ οχήματος

Μέσω ενός διαγράμματος μπλοκ παρουσιάζεται η γενική αρχιτεκτονική του ρομποτικού οχήματος (Εικόνα 10).



Εικόνα 10. *Διάγραμμα μπλοκ ρομποτικού οχήματος*

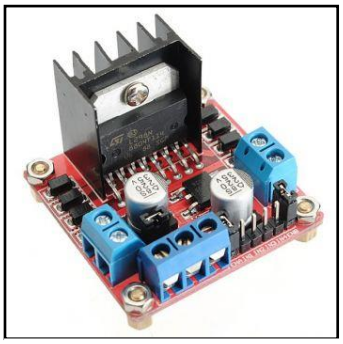
3.2 Hardware

Arduino UNO R3

Εικόνα 11. *Arduino UNO R3 SMD[IX]*

- Μικροελεγχτής ATPmega328
- Τάση εισόδου (συνιστάται) 7-12 V
- 14 ψηφιακές είσοδοι/έξοδοι (εκ των οποίων 6 μπορούν να λειτουργήσουν ως έξοδοι PWM)
- 6 αναλογικές είσοδοι
- Ταχύτητα ρολογιού 16 MHz
- Flash Memory 32 KB

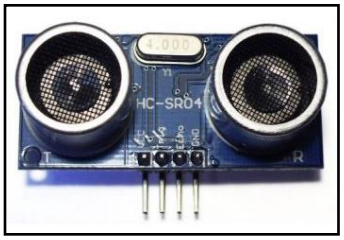
L298N



Εικόνα 12. *Οδηγός κινητήρων L298N[X]*

- Οδηγός κινητήρα διπλής κατεύθυνσης
- Τροφοδοσία DC 5-35 V
- 4 ακροδέκτες ελέγχου κατεύθυνσης
- 2 ακροδέκτες ελέγχου ταχύτητας

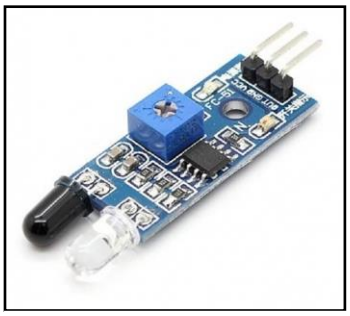
HC-SR04



Εικόνα 13. *Αισθητήρας Υπερήχων HC-SR04[XI]*

- Εύρος εντοπισμού 2-400 cm
- Τάση εισόδου 5 V
- 4 ακροδέκτες (TRIG, ECHO, Vcc, GND)
- Γωνία μέτρησης 15°

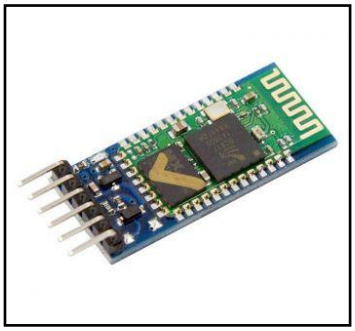
Infrared (IR) Sensor



- Εύρος εντοπισμού 2~30 cm
- Τάση εισόδου 3.3-5 V
- 3 ακροδέκτες (OUT, Vcc, GND)
- Γωνία μέτρησης 35°
- IR Sensor x2

Εικόνα 14. *Αισθητήρας υπέρυθρων αποφυγής εμποδίων[XII]*

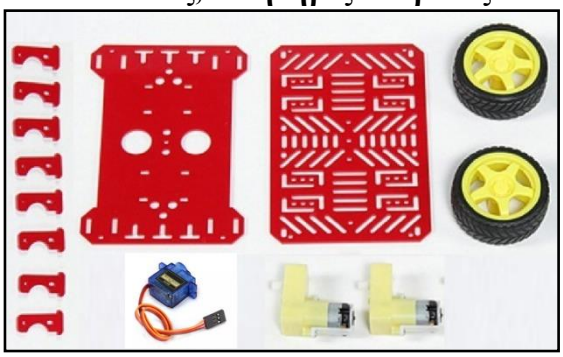
HC-05



- Τάση εισόδου 3.6-6 V
- 6 ακροδέκτες (EN, Vcc, GND, TXD, RXD, State)
- Εύρος έως και 10m

Εικόνα 15. *Bluetooth module[XIII]*

Σκελετός, κινητήρες και ρόδες



- 2 ακρυλικά φύλλα 175x109x3 mm
- 2 ρόδες 67x26 mm
- 2 DC κινητήρες τάσης 3 V
- Σερβοκινητήρας τάσης 4.2-6 V
- 8 ακρυλικές συνδέσεις φύλλων

Εικόνα 16. *Κιτ ρομποτικού οχήματος[XIV]*

Πίνακας 1: Πρόσθετο υλικό

Εξαρτήματα	Ποσότητα	Περιγραφή
Breadboard	1	170 οπών, για καλύτερη οργάνωση και διευκόλυνση στη συνδεσμολογία
Πυκνωτές	2	0.1 μF πριν από τους DC κινητήρες για μείωση θορύβου και εξομάλυνση διακυμάνσεων τάσης
Μπαταρίες	9	6xAA των 1.5V και 3x18650 των 3.7V
Βίδες	18	Στήριξη εξαρτημάτων
Παξιμάδια	10	Στήριξη εξαρτημάτων
Καλώδια		Σύνδεση εξαρτημάτων
Ετικέτες		Σήμανση καλωδίων
Μπαταριοθήκη	2	6xAA (σύνδεση με DC jack) 3x18650 με διακόπτη on/off
Ταινία διπλής όψης	1	Αφρώδης ταινία διπλής όψης για στήριξη εξαρτημάτων στο σκελετό
Βάση HC-SR04	1	Στήριξη του HC-SR04
Περιστρεφόμενες ρόδες	2	Στήριξη σκελετού και καλύτερη ικανότητα προς ελιγμό

3.3 Software

- **Arduino IDE**, όπως περιγράφεται στην ενότητα **2.4**.

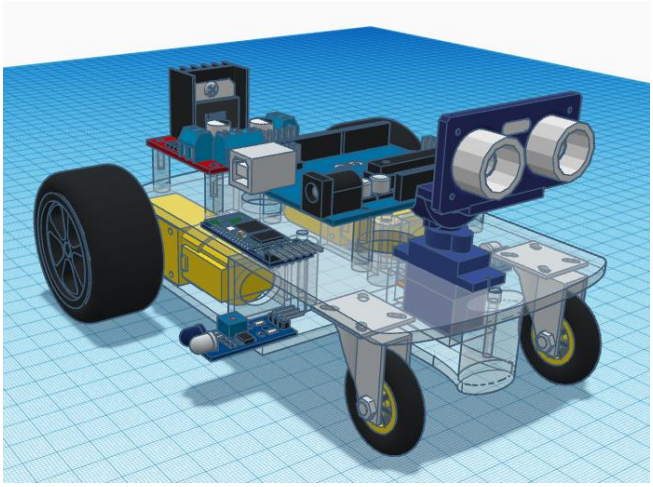
- **MIT App Inventor** είναι μια οπτική γλώσσα προγραμματισμού υψηλού επιπέδου βασισμένη σε block (δημιουργία προγραμμάτων με γραφικό χειρισμό). Βασίζεται πάνω στις γλώσσες Java, Swift, Objective-C, Kawa, Scheme, JavaScript και HTML και αποτελείται από ένα γραφικό περιβάλλον χρήστη (GUI), επιτρέποντας στο χρήστη να δημιουργήσει εφαρμογές για λειτουργικά Android και iOS [21].

- **Tinkercad** είναι ένα δωρεάν διαδικτυακό πρόγραμμα 3D σχεδιασμού. Βασίζεται στα WEB και Javascript και χρησιμοποιείται για τη δημιουργία 3D μοντέλων τα οποία μπορούν να εκτυπωθούν τρισδιάστατα αλλά και για την προσωμοίωση ηλεκτρονικών κυκλωμάτων, με τη χρήση των ενσωματωμένων βιβλιοθηκών που διαθέτει [21].

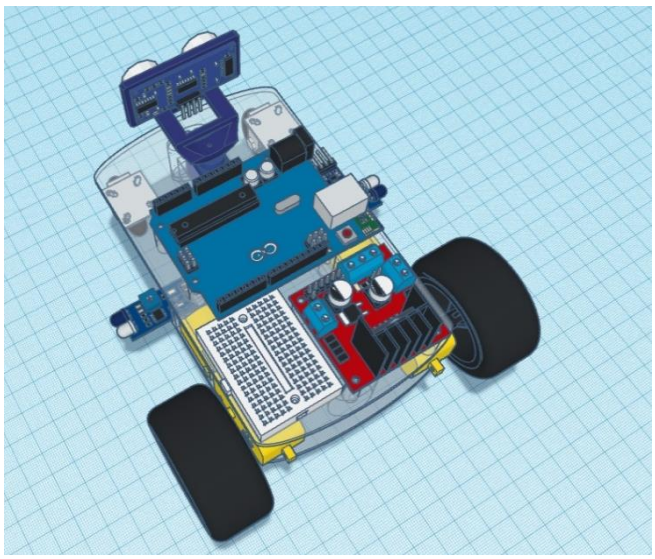
- **Fritzing** είναι μια εφαρμογή ανοιχτού κώδικα, η οποία επιτρέπει στο χρήστη να χρησιμοποιεί λογισμικό CAD για σχεδιασμό και προσωμοίωση ηλεκτρονικών κυκλωμάτων. Εμπνεύστηκε από το προγραμματιστικό περιβάλλον του Processing και τον μικροελεγχτή Arduino.Βασίζεται στη γλώσσα C++ [21].

3.4 Σχεδιασμός

Η ιδέα για τον σχεδιασμό του ρομποτικού οχήματος προϋπήρχε, ήδη απο τις πρώτες δύο εβδομάδες της διπλωματικής. Στην υλοποίησή της συνέβαλλε η έρευνα πάνω στο συγκεκριμένο κομμάτι, διαβάζοντας επιστημονικά άρθρα και παρακολουθώντας βίντεο ανάλογων project. Στη συνέχεια, αξιοποιήθηκε η πλατφόρμα 3D σχεδιασμού Tinkercad για την αρχική μοντελοποίηση του σχεδίου που επιλέχθηκε (Εικόνα 17) (Εικόνα 18).



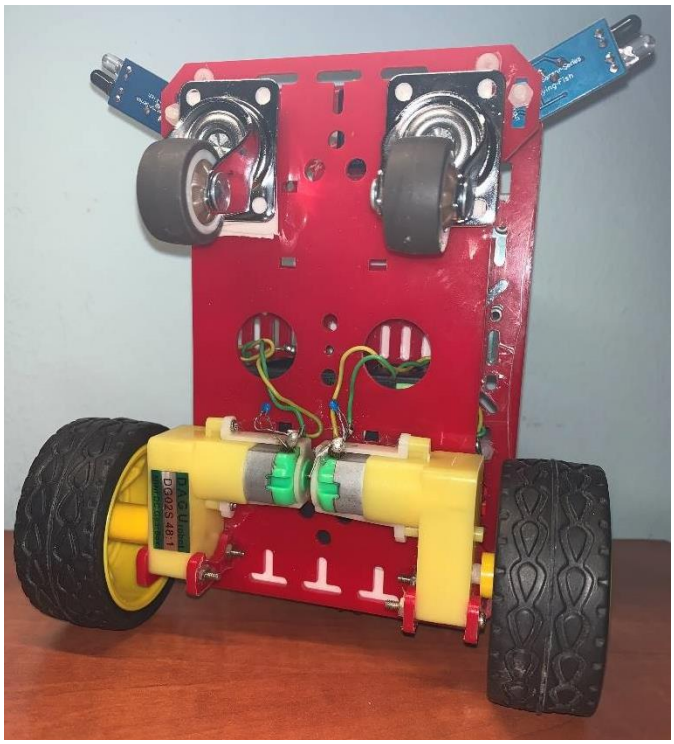
Εικόνα 17*. Πρόσοψη αρχικής 3D μοντελοποίησης ρομποτικού οχήματος μέσω Tinkercad*



Εικόνα 18. *Κάτοψη αρχικής 3D μοντελοποίησης ρομποτικού οχήματος μέσω Tinkercad*

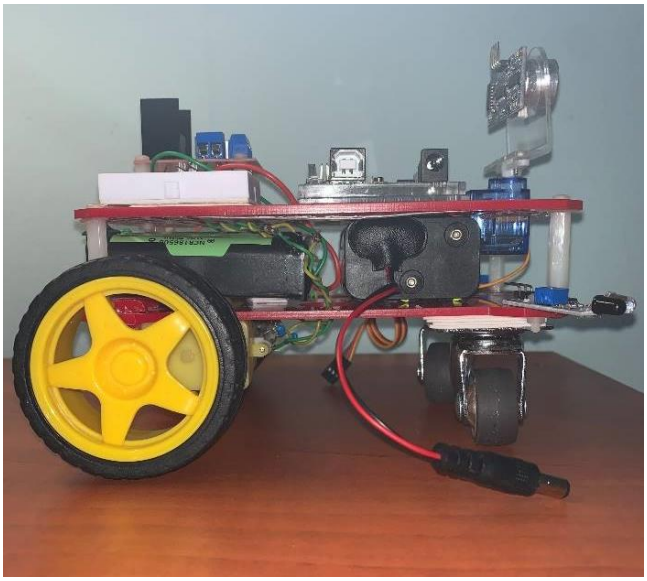
3.5 Συναρμολόγηση

Αρχικά έγινε ένωση των δύο τμημάτων από τα οποία συγκροτείται το σασί του οχήματος, μέσω πλαστικών διαχωριστικών που του προσφέρουν στήριξη. Στη συνέχεια, στο κάτω τμήμα (Εικόνα 19) βιδώθηκαν τα DC motors και τοποθετήθηκαν οι περιστρεφόμενες ρόδες με τη βοήθεια της αφρώδους ταινίας για να βρίσκονται στο σωστό ύψος και να είναι το φύλλο παράλληλο με το έδαφος .



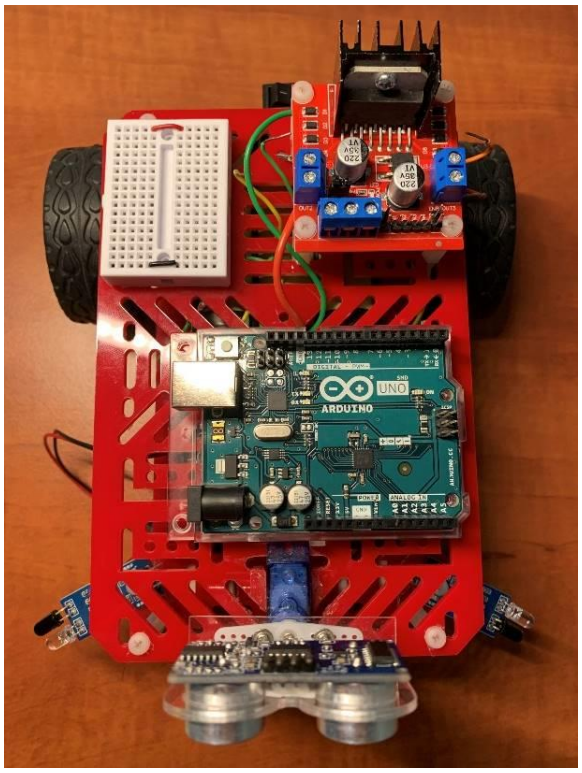
Εικόνα 19. *Κάτω τμήμα οχήματος*

Ανάμεσα στα δύο τμήματα του οχήματος (Εικόνα 20) τοποθετήθηκαν μια μπαταριοθήκη με 6 μπαταρίες των 1.5V, μια μπαταριοθήκη με 3 επαναφορτιζόμενες μπαταρίες των 3.7V και τα IR sensors, στα οποία έγινε αλλαγή θέσης σε σύγκριση με το αρχικό 3D μοντέλο για βελτίωση της έγκυρης αναγνώρισης εμποδίου και του ελιγμού του οχήματος. Οι συγκεκριμένοι αισθητήρες διαθέτουν ποτενσιόμετρα τα οποία με τη χρήση ενός κατσαβιδιού ρυθμίζουν την μέγιστη απόσταση αναγνώρισης εμποδίου. Για τη διπλωματική οι αισθητήρες έχουν ρυθμιστεί στα 10-12cm.



Εικόνα 20. *Ανάμεσα στα δύο τμήματα του οχήματος*

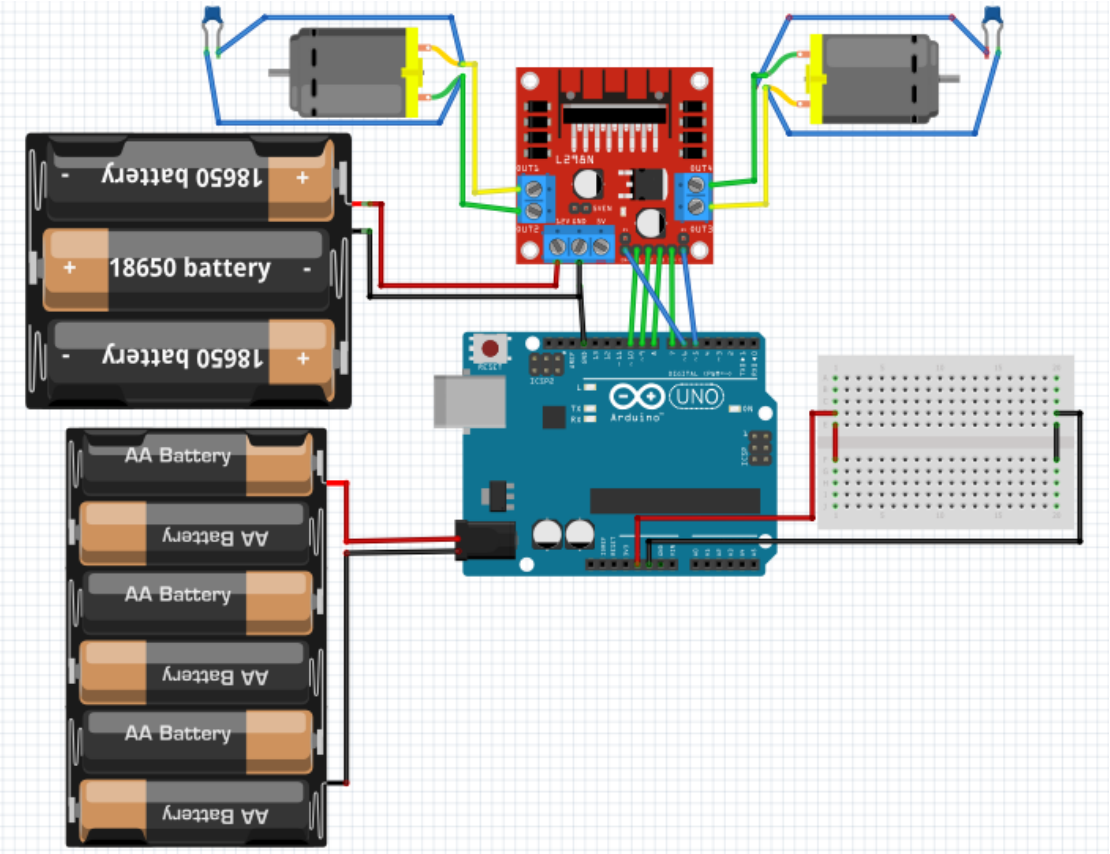
Για το πάνω τμήμα (Εικόνα 21), αρχικά ανοίχτηκε τρύπα με τη χρήση κολλητήριου για τη τοποθέτηση του σερβοκινητήρα. Στη συνέχεια χρησιμοποιήθηκε αφρώδης ταινία για την τοποθέτηση του Arduino UNO και του mini breadboard, πλαστικά διαχωριστικά και βίδες για το L298N και βίδες και αφρώδης ταινία για την τοποθέτηση του HC-SR04 και της βάσης του πάνω στο σερβοκινητήρα. Επίσης πραγματοποιήθηκε εναλλαγή των breadboard και L298N σε σύγκριση με το αρχικό σχέδιο για καλύτερη διαχείριση καλωδίων.



Εικόνα 21. *Πάνω τμήμα οχήματος*

3.6 Συνδεσμολογία

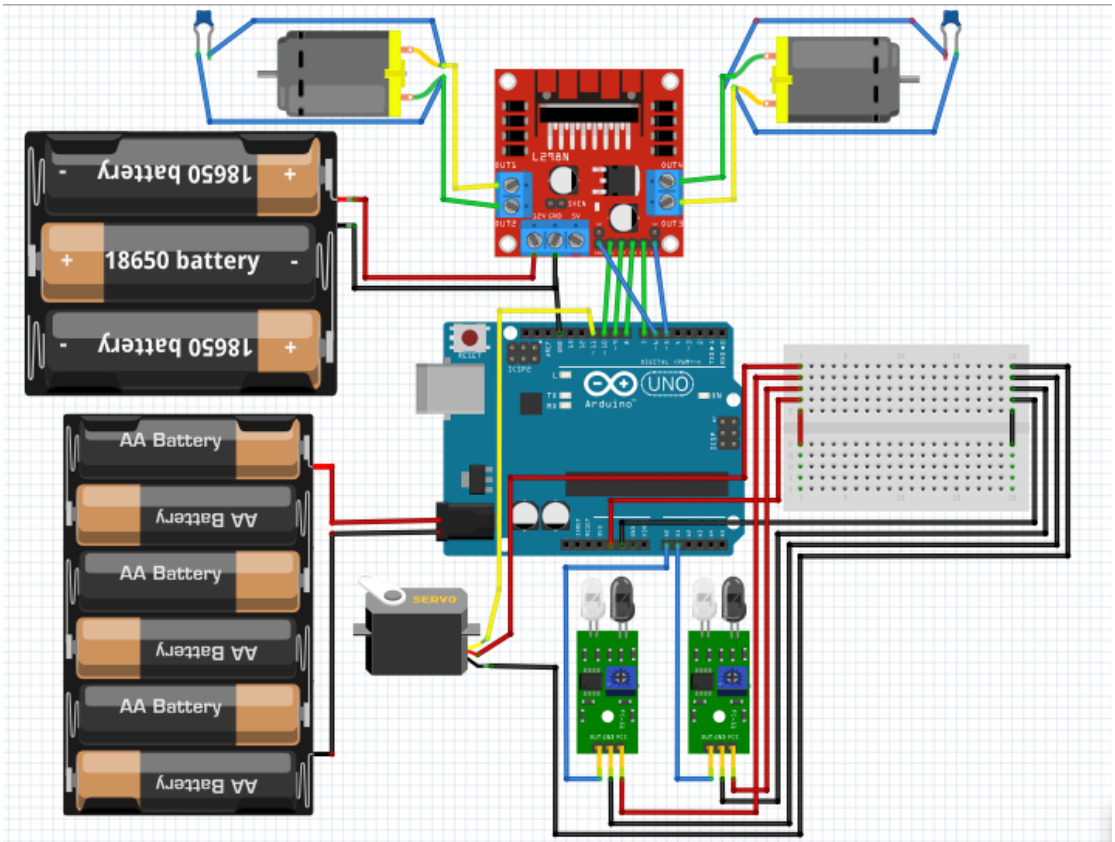
Αρχικά χρησιμοποιήθηκε ένα κολλητήρι, αλοιφή συγκόλλησης και ίνα συγκόλλησης για να ενωθούν δύο πυκνωτές των 0.1 μF με τους DC κινητήρες αντίστοιχα για μείωση θορύβου και εξομάλυνση τυχόν διακυμάνσεων τάσης, καθώς και καλώδια σαν επεκτάσεις. Στη συνέχεια τα καλώδια του δεξιού κινητήρα συνδέθηκαν με τα OUT1 και OUT2 αντίστοιχα του L298N και τα καλώδια του αριστερού με τα OUT3 και OUT4 αντίστοιχα. Ακολούθησε η σύνδεση της μπαταριοθήκης με το L298N (κόκκινο καλώδιο στη θύρα +12V, μαύρο καλώδιο στη θύρα GND και από αυτήν καλώδιο που συνδέεται με θύρα GND στο Arduino UNO), της μπαταριοθήκης με το Arduino μέσω της θύρας DC jack, ενός καλωδίου τροφοδοσίας +5V από τη θύρα 5V του Arduino στο breadboard για τη τροφοδοσία των εξαρτημάτων και ενός καλωδίου Voltage REF από μια θύρα GND του Arduino στο mini breadboard για να υπάρχει μια κοινή γείωση με όλα τα εξαρτήματα που θα συνδεθούν στη συνέχεια. Τέλος, συνδέθηκαν τα ENA, IN1, IN2, IN3, IN4 και ENB με τις θύρες 6,10,9,8,7 και 5 αντίστοιχα (προτιμήθηκαν οι 5 και 6 για τα ENA και ENB αντίστοιχα καθώς είναι έξοδοι PWM ίδιας συχνότητας 980Hz) (Εικόνα 22).



Εικόνα 22. *Συνδεσμολογία DC motors και L298N με Arduino*

Ακολούθησε η σύνδεση του σερβοκινητήρα και των αισθητήρων υπέρυθρων με το Arduino (Εικόνα 23). Ο σερβοκινητήρας έχει τρεις ακροδέκτες: τροφοδοσίας, γείωσης και σήματος ελέγχου. Οι δύο πρώτοι συνδέθηκαν με το breadboard σε υποδοχές με +5V και GND αντίστοιχα. Ο ακροδέκτης σήματος ελέγχου συνδέθηκε στη θύρα 11 του Arduino καθώς ο σερβοκινητήρας χρειάζεται τη τεχνική PWM για να λειτουργήσει σωστά. Στη συνέχεια συνδέθηκαν τα IR sensors τα οποία και αυτά έχουν τρεις ακροδέκτες: τροφοδοσίας, γείωσης και εξόδου. Οι δύο πρώτοι και των δύο αισθητήρων συνδέθηκαν στο breadboard

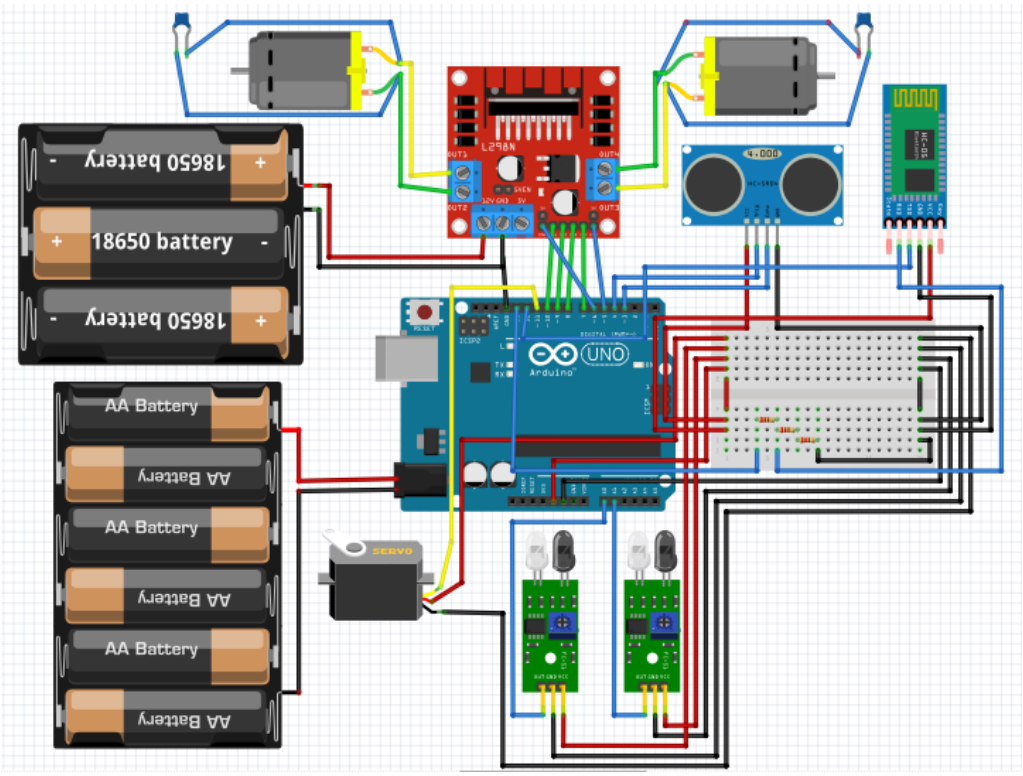
(+5V και GND αντίστοιχα), ο ακροδέκτης εξόδου του αριστερού αισθητήρα συνδέθηκε στην αναλογική θύρα A0 και ο αντίστοιχος του δεξιού αισθητήρα στην αναλογική θύρα A1.



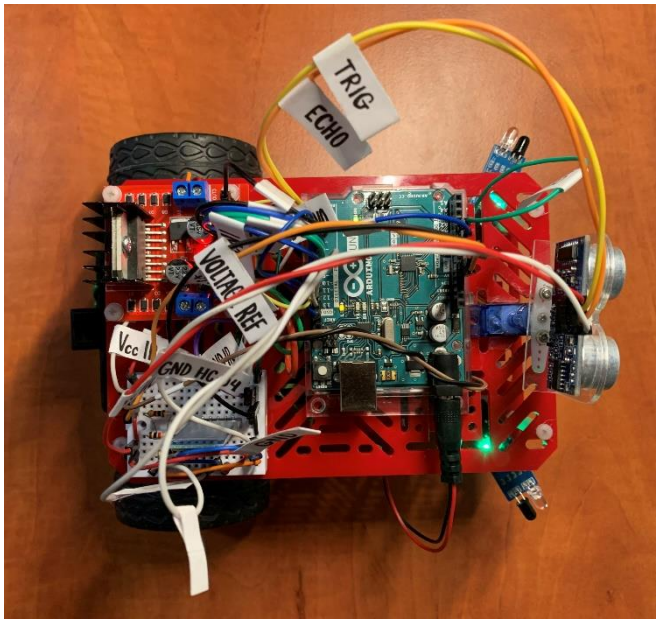
Εικόνα 23. *Συνδεσμολογία Servo motor και IR sensors με Arduino*

Τέλος, ακολούθησε η σύνδεση του αισθητήρα υπέρυχων HC-SR04 και του Bluetooth module HC-05 (Εικόνα 24). Για το HC-SR04, οι ακροδέκτες της τροφοδοσίας και της γείωσης συνδέθηκαν στο breadboard (+5V και GND αντίστοιχα) και οι ακροδέκτες των TRIG και ECHO στις θύρες 4 και 3 αντίστοιχα. Σχετικά με το HC-05, το Arduino UNO λειτουργεί στα 0V και 5V λογικά επίπεδα τάσης. Το HC-05 όμως λειτουργεί στα 0V και 3.3V λογικά επίπεδα τάσης. Από τα datasheets των ATMega328 και HC-05 προκύπτει πως θα πρέπει να τοποθετηθεί διαιρέτης τάσης μεταξύ της θύρας που θα οριστεί ως TX στο Arduino και του RX του HC-05 για να μετατραπούν τα 5V σε λογική 3.3V καθώς υπάρχει κίνδυνος να πάθει ζημιά το HC-05. Η συνδεσμολογία ακολούθησε ως εξής: από τους έξι ακροδέκτες χρησιμοποιήθηκαν οι τέσσερις (Vcc, GND,RX,TX). Τα Vcc και GND συνδέθηκαν στο breadboard (+5V και GND αντίστοιχα). Ο ακροδέκτης TX (πομπός) του HC-05 συνδέθηκε με τη θύρα 12 του Arduino που λειτουργεί ως RX (δέκτης) του Arduino. Ο ακροδέκτης RX του HC-05 συνδέθηκε με τη θύρα 13 του Arduino που λειτουργεί ως TX του Arduino, μέσω διαιρέτη τάσης. Το Arduino UNO ενώ διαθέτει ξεχωριστές θύρες RX και TX, προτιμήθηκε να χρησιμοποιηθεί, όπως θα φανεί και στο επόμενο κεφάλαιο, η βιβλιοθήκη SoftwareSerial και οι θύρες 12 και 13 καθώς κατά τη διάρκεια του uploading κώδικα αν οι θύρες RX και TX του Arduino είναι απασχολημένες, το uploading θα είναι ανεπιτυχές και θα πρέπει να αποσυνδεθούν οποιοιδήποτε ακροδέκτες για να είναι επιτυχές.

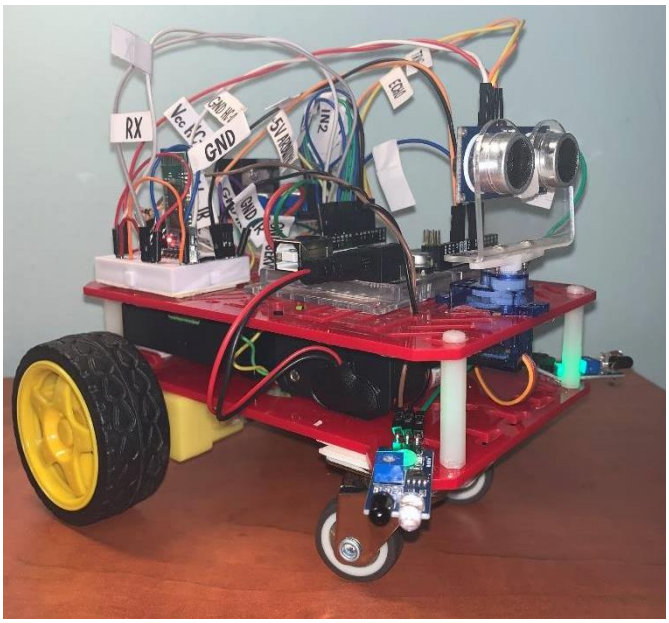
Αφού συνδέθηκαν και τα τελευταία εξαρτήματα, η κατασκευή του ρομποτικού οχήματος έχει ολοκληρωθεί (Εικόνα 25) (Εικόνα 26).



Εικόνα 24. *Τελική συνδεσμολογία ρομποτικού οχήματος*



Εικόνα 25. Κάτοψη ολοκληρωμένου ρομποτικού οχήματος



Εικόνα 26. Πλάγια όψη ολοκληρωμένου ρομποτικού οχήματος

4. Μεθοδολογία και προγραμματισμός

Σε αυτό το κεφάλαιο περιγράφεται η αρχή λειτουργίας του οχήματος και τα βήματα που ακολουθήθηκαν για να προγραμματιστεί.

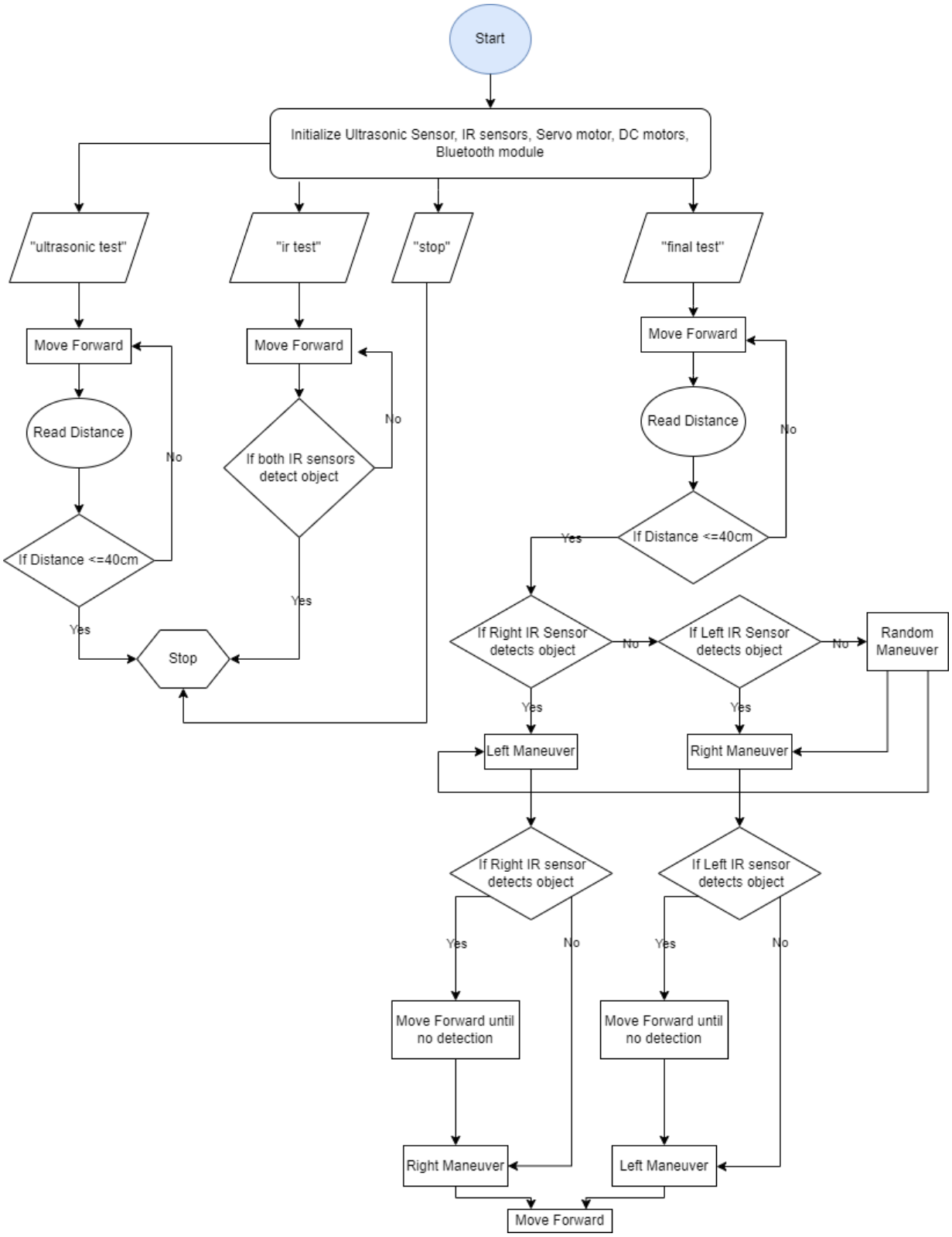
4.1 Αρχή λειτουργίας

Η αρχή λειτουργίας του ρομποτικού οχήματος έχει σχεδιαστεί να είναι η εξής:

- Αρχικά συνδέεται το DC jack της μπαταριοθήκης 6xAA με το Arduino και ενεργοποιείται ο διακόπτης on/off της μπαταριοθήκης 3x18650 του L298N.
- Στη συνέχεια ανοίγεται η εφαρμογή του MIT App Inventor σε smartphone και πραγματοποιείται σύνδεση με το Arduino μέσω του HC-05. Το όχημα είναι σε κατάσταση ετοιμότητας, προετοιμασμένο να δεχτεί φωνητικές εντολές από το χρήστη.
- Ο αριθμός των φωνητικών εντολών που μπορεί να δοθεί είναι τέσσερις (‘’ultrasonic test’’, ir test, final test και stop).
- Αν δοθεί η φωνητική εντολή ‘’ultrasonic test’’, το όχημα ξεκινά να κινείται με ευθεία πορεία. Αν ο αισθητήρας υπερήχων εντοπίζει εμπόδιο στα 40cm, το όχημα θα σταματήσει ακαριαία.
- Αν δοθεί η φωνητική εντολή ‘’ir test’’, το όχημα ξεκινά να κινείται με ευθεία πορεία. Αν και οι δύο αισθητήρες υπερύθρων εντοπίσουν εμπόδιο, το όχημα θα σταματήσει ακαριαία.
- Αν δοθεί η φωνητική εντολή ‘’final test’’, το όχημα ξεκινά να κινείται με ευθεία πορεία. Αν ο αισθητήρας υπερήχων εντοπίζει εμπόδιο στα 40cm, συμβαίνει το εξής:
    - Αν υπάρχει κάτι στα δεξιά ή αριστερά του οχήματος (π.χ. τοίχος) τη στιγμή που εντοπίζεται εμπόδιο στα 40cm, γίνεται εντοπισμός από τον εκάστοτε αισθητήρα υπερύθρων και το όχημα πραγματοποιεί ελιγμό στην αντίθετη κατεύθυνση με ελαττωμένη ταχύτητα.
    - Αν δεν υπάρχει κάτι στα δεξιά ή αριστερά του οχήματος, το όχημα πραγματοποιεί τυχαίο ελιγμό.
- Μόλις ολοκληρωθεί ο ελιγμός, συμβαίνει το εξής:
    - Ανάλογα τον ελιγμό, γίνεται εντοπισμός από τον αντίθετο αισθητήρα υπερύθρων (π.χ. αν αριστερός ελιγμός, γίνεται εντοπισμός από τον δεξιό IR sensor). Όσο ο αισθητήρας παραμένει ενεργοποιημένος , δηλαδή όσο εντοπίζει το εμπόδιο, το όχημα θα κινείται ευθεία με ελαττωμένη ταχύτητα. Μόλις ο αισθητήρας απενεργοποιηθεί, δηλαδή σταματήσει να εντοπίζει το εμπόδιο, θα πραγματοποιήσει αντίθετο ελιγμό.
    - Αν ο αισθητήρας υπερήχων δεν εντοπίσει το εμπόδιο, τότε το όχημα πραγματοποιεί κατευθείαν αντίθετο ελιγμό.
- Αφού ολοκληρωθεί και ο δεύτερος ελιγμός, τότε το όχημα θα έχει πραγματοποιήσει επιτυχή αποφυγή εμποδίου και θα έχει ευθυγραμμίσει τη πορεία του.
- Τέλος, αν δοθεί η φωνητική εντολή ‘’stop’’, το όχημα θα σταματήσει ακαριαία.

4.2 Διάγραμμα ροής προγράμματος

Ακολουθεί διάγραμμα ροής του προγράμματος (Εικόνα 27), το οποίο αναπαριστά βηματικά την αρχή λειτουργίας του για καλύτερη κατανόηση του αλγορίθμου.



Εικόνα 27. *Διάγραμμα ροής προγράμματος*

4.3 Προγραμματισμός

Όπως αναφέρθηκε και στο υποκεφάλαιο 2.4, θα χρησιμοποιηθεί το προγραμματιστικό περιβάλλον Arduino IDE για το προγραμματισμό του οχήματος καθώς και το MIT App Inventor για τη δημιουργία εφαρμογής σε λειτουργικό Android. Ο μικροελεγκτής αφού συνδέεται στον υπολογιστή με USB, επιλέγεται το Verify όπου κάνει Compile (μεταγλώττιση) το sketch και στη συνέχεια επιλέγεται το Upload για να γίνει uploading.

4.3.1 Βιβλιοθήκες

Θα χρησιμοποιηθούν συνολικά τρεις βιβλιοθήκες, με τις δύο να χρειάζεται να ληφθούν μέσω του Arduino ώστε να μπορέσει να τις κάνει Compile. Οι βιβλιοθήκες είναι οι εξής:

- **Newping.h**: βιβλιοθήκη για τον αισθητήρα υπερήχων η οποία μεγιστοποιεί τις δυνατότητες του, χρειάζεται να ληφθεί
- **Servo.h**: βιβλιοθήκη για το σερβοκινητήρα
- **SoftwareSerial.h**: βιβλιοθήκη για επικοινωνία μέσω Bluetooth, χρειάζεται να ληφθεί

Μέσω της εντολής **#include** συμπεριλαμβάνονται οι βιβλιοθήκες στο sketch:

```
#include <NewPing.h>          //  NewPing library for maximising ultrasonic sensor capabilities
#include <Servo.h>            //  Servo library for controlling the servo motor
#include <SoftwareSerial.h>   //  SoftwareSerial library for Bluetooth communication
```

Εικόνα 28. *Βιβλιοθήκες που χρησιμοποιούνται*

4.3.2 Καθορισμός και αρχικοποίηση

Μέσω της εντολής **#define** δίνονται ονόματα σε τιμές σταθεράς πριν γίνουν compile. (Εικόνα 29). Δηλαδή όταν χρησιμοποιούνται στο πρόγραμμα, τα ονόματα αντικαθιστούνται από τις τιμές τους λίγο πριν το compile, προσφέροντας ευκολότερη τροποποίηση και διάβασμα του αλγορίθμου. Συγκεκριμένα:

- Τα Digital Pins **6**,**10**,**9**,**8**,**7** και **5** του Arduino αντικαθιστούνται με τα ονόματα **ENA**, **IN1**, **IN2**, **IN3**, **IN4** και **ENB** αντίστοιχα
- Τα Digital Pins **11**,**4** και **3** του Arduino αντικαθιστούνται με τα ονόματα **SERVO_PIN**, **TRIG_PIN** και **ECHO_PIN** αντίστοιχα
- Τα Analog Pins **A0** και **A1** του Arduino αντικαθιστούνται με τα ονόματα **LEFT_IR_PIN** και **RIGHT_IR_PIN** αντίστοιχα
- Οι σταθερές τιμές **400**, **40** και **500** αντικαθιστούνται με τα ονόματα **MAX_DISTANCE**, **OBST_DISTANCE** και **IR_THRESHOLD** αντίστοιχα

```
// Define pin numbers
#define ENA 6
#define IN1 10
#define IN2 9
#define IN3 8
#define IN4 7
#define ENB 5
#define SERVO_PIN 11
#define TRIG_PIN 4
#define ECHO_PIN 3
#define LEFT_IR_PIN A0
#define RIGHT_IR_PIN A1

// Define constants
#define MAX_DISTANCE 400          // Maximum distance for ultrasonic sensor in cm
#define OBST_DISTANCE 40          // Distance threshold to detect obstacle in cm
#define IR_THRESHOLD 500          // Threshold value for IR sensors
```

Εικόνα 29. *Εντολή* ***#define*** *και χρήση της*

Απαραίτητη κρίνεται η δημιουργία των αντικειμένων '**myServo'** (της κλάσης Servo), '**Sonar'** (της κλάσης NewPing) και '**mySerial'** (της κλάσης SoftwareSerial).(Εικόνα 30) Οι θύρες **12** και **13** του Arduino τοποθετούνται σε () έτσι ώστε να δηλωθεί πως θα καταλαμβάνονται από τα RX και TX αντίστοιχα του HC-05. Μέσω αυτών ξεκλειδώνονται μέθοδοι και λειτουργίες που παρέχονται από τις τρεις βιβλιοθήκες που χρησιμοποιούνται, και θα φανούν χρήσιμα στη συνέχεια.

```
// Objects creation
Servo myServo;                    // Create a Servo object
NewPing sonar(TRIG_PIN, ECHO_PIN, MAX_DISTANCE);  // Create a NewPing object
SoftwareSerial mySerial(12, 13); // RX, TX - Create a SoftwareSerial object for Bluetooth
```

Εικόνα 30. *Δημιουργία αντικειμένων*

Με τη χρήση του '**enum'**, ενός τύπου δεδομένων που αποτελείται από ένα σύνολο ονομασμένων τιμών, δημιουργούνται δύο τύποι δεδομένων:

- Το 'enum' ορίζει το τύπο δεδομένων '**Commands'** με προκαθορισμένες εντολές που μπορεί να λάβει το όχημα, και σε κάθε μία προσάπτεται μια αριθμητική σταθερά (CMD_START → 0, CMD_STOP → 1 και ούτω καθεξής) (Εικόνα 31)
- Το 'enum' ορίζει το τύπο δεδομένων '**State'** όπου περιέχει τις καταστάσεις στις οποίες θα βρεθεί το όχημα, και σε κάθε μία προσάπτεται μια αριθμητική σταθερά, όπως και στο '**Commands'** (Εικόνα 32)

```
// Define commands enum
enum Commands {
  CMD_START,
  CMD_STOP,
  CMD_ULTRASONIC_TEST,
  CMD_IR_TEST,
  CMD_INVALID
};
```

Εικόνα 31. enum Commands

```
// Define states for non-blocking delay management
enum State {
  MOVE_FORWARD,
  MOVE_FORWARD_WITH_NEWSPEED,
  AVOID_OBSTACLE,
  MANEUVER_LEFT,
  MANEUVER_RIGHT,
  CHECK_LEFT,
  CHECK_RIGHT,
  MOVE_FORWARD_WHILE_CHECKING_LEFT,
  MOVE_FORWARD_WHILE_CHECKING_RIGHT,
  CORRECT_LEFT,
  CORRECT_RIGHT,
  ALIGN_LEFT,
  ALIGN_RIGHT,
  ULTRASONIC_TEST,
  IR_TEST
};
```

Εικόνα 32. enum State

Στη συνέχεια γίνεται η αρχικοποίηση κάποιων μεταβλητών (Εικόνα 33):

- Μέσω της '**unsigned long'**, όπου χρησιμοποιείται για την αποθήκευση αριθμών από το 0 μέχρι και το 4,294,967,295, δημιουργούνται δύο μεταβλητές, η '**unsigned long previousMillis = 0;**', η οποία χρησιμοποιείται για να αποθηκεύει τον χρόνο όταν εκτελέστηκε η τελευταία ενέργεια, και η '**unsigned long stateStartMillis = 0;'**, η οποία χρησιμοποιείται για να αποθηκεύει τον χρόνο όταν ξεκινάει μια κατάσταση.
- Μέσω της '**bool'**, ενός τύπου δεδομένων που παίρνει μόνο τις τιμές '**true'** ή '**false'** και χρησιμοποιείται μόνο για τιμές που μπορούν να βρίσκονται σε δύο μόνο καταστάσεις. Στον αλγόριθμο χρησιμοποιείται η '**bool robotStarted = false;**', δηλαδή δημιουργείται η μεταβλητή '**robotStarted**' όπου θα μπορεί να πάρει μόνο τις τιμές '**true'** ή '**false'** και αρχικοποιείται με τη τιμή '**false**'. Η τιμή αυτή επιδεικνύει πως το όχημα δεν έχει ξεκινήσει ακόμη να κινείται.
- Μέσω της '**State**' αρχικοποιείται η μεταβλητή κατάσταση '**currentState**' στη τιμή '**MOVE_FORWARD**'.
- Μέσω της '**int**' και της '**const int**', όπου χρησιμοποιούνται για να αρχικοποιήσουν τις μεταβλητές '**currentSpeed**', '**newspeed**' και '**minSpeed**' στις τιμές 58,45 και 35 αντίστοιχα. Η διαφορά της '**const**' είναι πως μετατρέπει την '**int**' σε ''read-only'' και δεν επιτρέπει την αλλαγή της τιμής της.

```
// Initialize variables
unsigned long previousMillis = 0;  // Store the last time an action was performed
bool robotStarted = false;         // Track whether the robot is started
State currentState = MOVE_FORWARD;  // Initialize the current state to MOVE_FORWARD
unsigned long stateStartMillis = 0; // Store the time when the state started

// Speed variables
int currentSpeed = 58;
int newspeed = 45;
const int minSpeed = 35;
```

Εικόνα 33. *Αρχικοποίηση μεταβλητών*

Τέλος, δημιουργείται η συνάρτηση **setup()** (Εικόνα 34), όπου καλείται πάντα λίγο πριν ξεκινήσει το sketch και θα 'τρέξει' μόνο μια φορά, είτε όταν δίνεται τροφοδοσία στο Arduino, είτε όταν επανεκκινείται. Μέσω αυτής αρχικοποιούνται pins (θύρες) του Arduino, μεταβλητές βιβλιοθηκών και γενικότερα μεταβλητές. Η **void** που τοποθετήθηκε μπροστά από τη **setup()** χρησιμοποιείται αποκλειστικά σε δηλώσεις συναρτήσεων.

Στη **setup()** του αλγορίθμου αρχικά γίνεται χρήση της **pinMode()**, μιας συνάρτησης που διαμορφώνει ένα pin σε είσοδο ή εξοδο, στα pins του L298N και στο pin του σερβοκινητήρα και τις διαμορφώνει ως εξόδους, καθώς και των **Serial.begin(9600)** και **mySerial.begin(9600)**, εντολές που ξεκινούν την σειριακή επικοινωνία και την SoftwareSerial επικοινωνία αντίστοιχα σε 9600 baud (bits ανά δευτερόλεπτο).

Λόγω της χρήσης των βιβλιοθηκών **NewPing** και **SoftwareSerial**, τα **TRIG**, **ECHO**, **RX** και **TX** δε χρειάζεται να διαμορφωθούν μέσω της **pinMode()** καθώς έχουν ήδη διαμορφωθεί, και συγκεκριμένα:

- Το **TRIG** pin έχει οριστεί ως έξοδος καθώς στέλνει παλμούς
- Το **ECHO** pin έχει οριστεί ως είσοδος καθώς ''ακούει'' την ηχώ των παλμών
- Το **RX** pin έχει οριστεί ως είσοδος καθώς λαμβάνει δεδομένα και
- Το **TX** pin έχει οριστεί ως έξοδος καθώς εκπέμπει δεδομένα

Η **Serial.println** χρησιμοποιείται για να σταλεί μήνυμα προς το χρήστη πως το Bluetooth είναι έτοιμο για να πραγματοποιηθεί σύνδεση.

```
void setup() {
  pinMode(ENA, OUTPUT);           // Set ENA as output
  pinMode(IN1, OUTPUT);           // Set IN1 as output
  pinMode(IN2, OUTPUT);           // Set IN2 as output
  pinMode(IN3, OUTPUT);           // Set IN3 as output
  pinMode(IN4, OUTPUT);           // Set IN4 as output
  pinMode(ENB, OUTPUT);           // Set ENB as output
  pinMode(SERVO_PIN, OUTPUT);     // Set SERVO_PIN as output
  myServo.attach(SERVO_PIN);      // Attach the servo to SERVO_PIN
  Serial.begin(9600);             // Start the Serial communication at 9600 baud rate
  mySerial.begin(9600);           // Start the SoftwareSerial communication at 9600 baud rate
  Serial.println("Bluetooth is ready for connection. Connect to HC-05 with 1234 as pairing key!");
}
```

Εικόνα 34. *Συνάρτηση void setup()*

4.3.3 Συναρτήσεις κίνησης

Για τη κίνηση του οχήματος, τα IN1, IN2 και ENA είναι υπεύθυνα για τον δεξιό κινητήρα ενώ τα IN3, IN4 και ENB για τον αριστερό. δημιουργούνται οι εξής συναρτήσεις:

- Η **moveForward(int speed)** (Εικόνα 35)**,** που διαμορφώνει το όχημα να κινηθεί ευθεία. Χρησιμοποιούνται οι εντολές: **digitalWrite()**, η οποία καθορίζει αν ένας ακροδέκτης θα έχει τη τιμή HIGH ή LOW, και η **analogWrite(),** η οποία 'γράφει' μια αναλογική τιμή σε έναν ακροδέκτη. Στη συνάρτηση αυτή, η **digitalWrite()** καθορίζει τα IN1 και IN3 σε HIGH και τα IN2 και IN4 σε LOW ενώ η **analogWrite()** καθορίζει τη περιστροφική ταχύτητα των κινητήρων μέσω του σήματος PWM, χρησιμοποιώντας τη μεταβλητή **int speed**.

```
void moveForward(int speed) {
  digitalWrite(IN1, HIGH);
  digitalWrite(IN2, LOW);
  digitalWrite(IN3, HIGH);
  digitalWrite(IN4, LOW);
  analogWrite(ENA, speed);
  analogWrite(ENB, speed);
}
```

Εικόνα 35. *Συνάρτηση* ***void moveForward(int speed)***

- Η **maneuverLeft()** (Εικόνα 36), που διαμορφώνει το όχημα να κάνει ελιγμό προς τα αριστερά. Για να γίνει σωστά ο ελιγμός, τα IN1, IN3 διαμορφώνονται σε HIGH, τα IN2,IN4 σε LOW, το ENA σε 55 και το ENB σε 35.

```
void maneuverLeft() {
  digitalWrite(IN1, HIGH);
  digitalWrite(IN2, LOW);
  digitalWrite(IN3, HIGH);
  digitalWrite(IN4, LOW);
  analogWrite(ENA, 55);
  analogWrite(ENB, 35);
}
```

Εικόνα 36. *Συνάρτηση* ***maneuverLeft()***

- Η **maneuverRight()** (Εικόνα 37), που διαμορφώνει το όχημα να κάνει ελιγμό προς τα δεξιά. Για να γίνει σωστά ο ελιγμός, τα IN1, IN3 διαμορφώνονται σε HIGH, τα IN2,IN4 σε LOW, το ENA σε 35 και το ENB σε 55.

```
void maneuverRight() {
  digitalWrite(IN1, HIGH);
  digitalWrite(IN2, LOW);
  digitalWrite(IN3, HIGH);
  digitalWrite(IN4, LOW);
  analogWrite(ENA, 35);
  analogWrite(ENB, 55);
}
```

Εικόνα 37. *Συνάρτηση* ***maneuverRight()***

- Η **stopMotors()** (Εικόνα 38), που διαμορφώνει το όχημα να σταματήσει. Τα IN1, IN2, IN3, IN4 διαμορφώνονται σε LOW και τα ENA, ENB σε 0.

```
void stopMotors() {
  digitalWrite(IN1, LOW);
  digitalWrite(IN2, LOW);
  digitalWrite(IN3, LOW);
  digitalWrite(IN4, LOW);
  analogWrite(ENA, 0);
  analogWrite(ENB, 0);
}
```

Εικόνα 38. *Συνάρτηση* ***stopMotors()***

- Η **speedUpToSpeed(int targetSpeed)** (Εικόνα 39), που διαμορφώνει το όχημα να επιταχύνει. Η μεταβλητή **targetSpeed** είναι η αποβλεπόμενη ταχύτητα και θα μπορεί να είναι μια από τις τρεις μεταβλητές '**currentSpeed**' ή '**newspeed**'. Χρησιμοποιείται ένας βρόχος **while** ο οποίος θα επαναλαμβάνεται μέχρι να επιτευχθεί η έκφραση **currentSpeed < targetSpeed**. Κάθε φορά που επαναλαμβάνεται ο βρόχος, η ταχύτητα θα αυξάνεται +1 μέσω της **currentSpeed += 1;** με delay 10ms, μέχρι να επιτευχθεί η ταχύτητα στόχος.

```
void speedUpToSpeed(int targetSpeed) {
  while (currentSpeed < targetSpeed) {
    currentSpeed += 1;
    moveForward(currentSpeed);
    delay(10);  // Delay adjustment for smoother or quicker transitions
  }
}
```

Εικόνα 39. *Συνάρτηση* ***speedUpToSpeed(int targetSpeed)***

- Η **slowDownToSpeed(int targetSpeed)** (Εικόνα 40), που διαμορφώνει το όχημα ώστε να ελαττώσει τη ταχύτητα του. Ισχύει το ίδιο με την **speedUpToSpeed(int targetSpeed)** αλλά αλλάζει η έκφραση του βρόχου (**currentSpeed > targetSpeed**). Κάθε φορά που επαναλαμβάνεται ο βρόχος, η ταχύτητα θα μειώνεται -1 μέσω της **currentSpeed -= 1;** με delay 10ms, μέχρι να επιτευχθεί η ταχύτητα στόχος.

```
void slowDownToSpeed(int targetSpeed) {
  while (currentSpeed > targetSpeed) {
    currentSpeed -= 1;
    moveForward(currentSpeed);
    delay(10);  // Delay adjustment for smoother or quicker transitions
  }
}
```

Εικόνα 40. *Συνάρτηση* ***slowDownToSpeed(int targetSpeed)***

4.3.4 Συναρτήσεις αισθητήρων και σερβοκινητήρα

Για τον αισθητήρα υπερήχων, αφού έχει γίνει η δημιουργία του αντικειμένου '**sonar**', δημιουργείται η συνάρτηση **readUltrasonicDistance()** (Εικόνα 41). Μέσω της NewPing, χρησιμοποιείται η εντολή **sonar.ping()**, η οποία στέλνει ηχητικό παλμό και λαμβάνει σαν δεδομένα το χρόνο που έκανε να επιστρέψει ο παλμός. Στη συνέχεια υπολογίζεται η απόσταση μέσω της πράξης **distance = uS / US_ROUNDTRIP_CM;** και σαν **return** λαμβάνεται η απόσταση του εμποδίου που εντοπίστηκε.

```
int readUltrasonicDistance() {
  unsigned int uS = sonar.ping();       // Send a ping and get the ping time in microseconds
  int distance = uS / US_ROUNDTRIP_CM;  // Convert the ping time to distance in cm
  return distance;                      // Return the distance
}
```

Εικόνα 41. *Συνάρτηση* ***readUltrasonicDistance()***

Για τους αισθητήρες υπερύθρων, δημιουργείται η συνάρτηση **readIRSensor(int pin)** (Εικόνα 42), όπου στη θέση της μεταβλητής pin θα τοποθετείται είτε RIGHT_IR_PIN είτε LEFT_IR_PIN. Μέσω της analogRead(pin), διαβάζεται η τιμή του επιλεγμένου αισθητήρα στο καθορισμένο αναλογικό ακροδέκτη και επιστρέφει τη τιμή αυτή.

```
int readIRSensor(int pin) {
  return analogRead(pin);              // Read the analog value from the specified IR sensor pin
}
```

Εικόνα 42. *Συνάρτηση* ***readIRSensor(int pin)***

Για το σερβοκινητήρα, δημιουργείται η συνάρτηση **moveServo(int angle)** (Εικόνα 43), όπου στη θέση της μεταβλητής angle θα τοποθετείται σε μοίρες η γωνία που θα κάνει ο σεβοκινητήρας. Το ίδιο ισχύει και για την εντολή **myServo.write(angle)**, η οποία εκτελεί τη στροφή του σερβοκινητήρα, ανάλογα τη γωνία.

```
void moveServo(int angle) {
  myServo.write(angle);  // Move the servo to the specified angle
  delay(100);   // Wait for the servo to reach the desired position
}
```

Εικόνα 43. *Συνάρτηση* ***moveServo(int angle)***

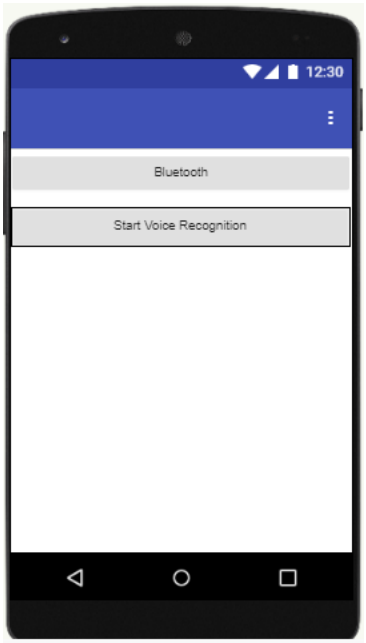
4.3.5 Συναρτήσεις και δημιουργία εφαρμογής ασύρματης επικοινωνίας

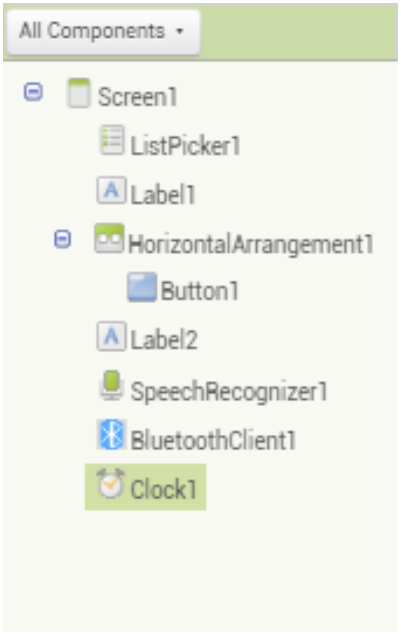
Πριν δημιουργηθούν οι συναρτήσεις, θα πρέπει να δημιουργηθεί μια εφαρμογή ασύρματης επικοινωνίας για να μπορέσει να γίνει ικανή η επικοινωνία του χρήστη με το HC-05 και αυτό με τη σειρά του με το Arduino. Για τη δημιουργία της χρησιμοποιείται το MIT App Inventor και ένα smartphone λειτουργικού Android, καθώς το HC-05 δεν γίνεται να συνδεθεί με smartphone λειτουργικού iOS λόγω διαφορετικού λειτουργικού Bluetooth.

Τα βήματα είναι τα εξής:

- Projects→Start new project→Voice_recognition (Όνομα project)
- Από το User Interface του Designer επιλέγονται και ‘’σέρνονται’’ στην οθόνη του smartphone (Viewer) (Εικόνα 44) (Εικόνα 45) τα εξής:
  - ListPicker1 (Width → Fill parent, Text → ‘’Bluetooth’’)
  - Label1 (Width → Fill parent, Text → ‘’ ‘’)
  - HorizontalArrangement1 (Width → Fill parent)
  - Button1 (Width → Fill parent, Text → ‘Start Voice Recognition)
  - Label2 (Width → Fill parent, Text → ‘’ ‘’)
  - SpeechRecognizer1
  - BluetoothClient1
  - Clock1
- Αφού γίνει η εναλλαγή στο Blocks Editor, ‘’σέρνονται’’ τα κατάλληλα blocks στο Viewer. (Εικόνα 46)

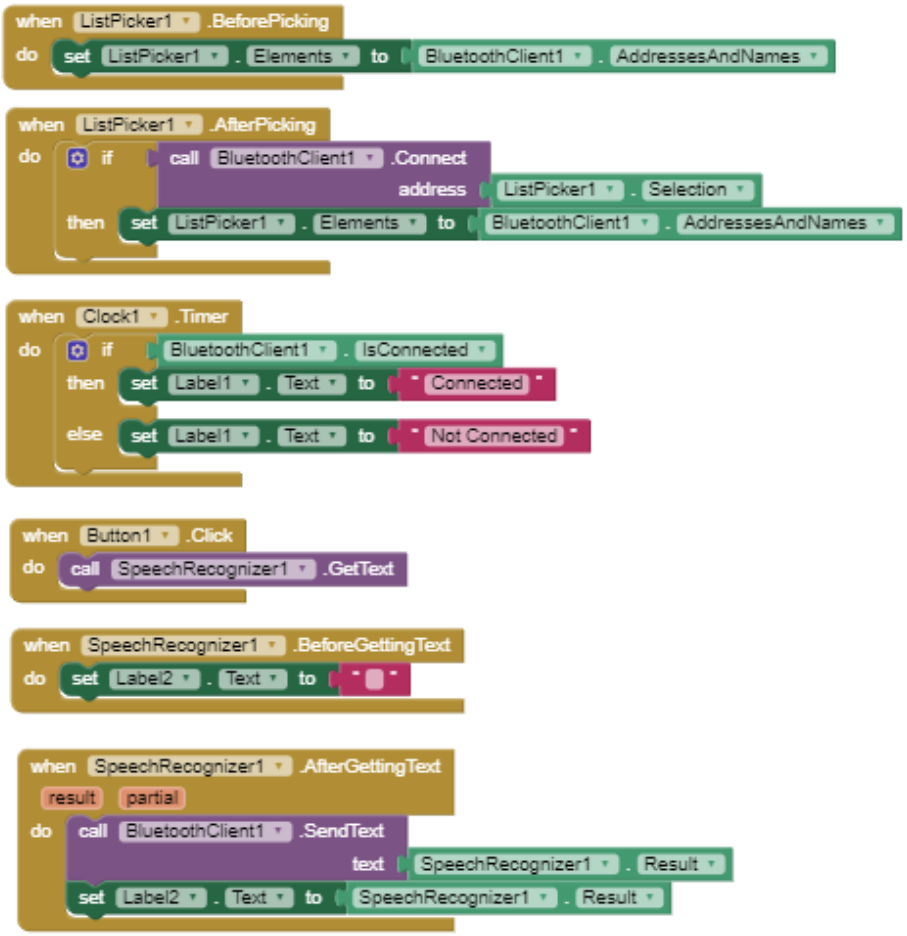
Η αρχή λειτουργίας της εφαρμογής είναι η εξής: μόλις πατηθεί το ‘’Bluetooth’’, εμφανίζονται οι συσκευές με τις οποίες έχει ήδη γίνει pairing από τις τις ρυθμίσεις του smartphone. Αφού επιλεχθεί το HC-05, κάτω από το ‘’Bluetooth’’ θα εμφανιστεί μήνυμα ‘’Connected’’ (σε οποιαδήποτε άλλη περίπτωση θα εμφανιστεί μήνυμα ‘’Not Connected’’). Στη συνέχεια επιλέγεται το ‘’Start Voice Recognition’’ και ο χρήστης δίνει μια φωνητική οδηγία. Η οδηγία εμφανίζεται στην οθόνη του smartphone και τέλος, στέλνεται στο HC-05 και αυτό με τη σειρά του το στέλνει στο Arduino.





Εικόνα 44. *Viewer του Designer*

Εικόνα 45. *Components του Designer*



Εικόνα 46. *Viewer του Blocks Editor*

Αφού δημιουργήθηκε η εφαρμογή, δημιουργούνται και οι συναρτήσεις. Για τη πρώτη συνάρτηση:

- **void processCommand(String command) (**Εικόνα 47) (Εικόνα 48): συνάρτηση με όνομα 'processCommand' όπου το command ορίζεται ως string, δηλαδή ως τύπο δεδομένων στον οποίο αποθηκεύεται κείμενο.
- **int cmd = parseCommand(command)**: καλείται η συνάρτηση parseCommand.
- **switch (cmd)**: χρησιμοποιείται μια switch για να γίνει ευκολότερος και πιο γρήγορος χειρισμός των command cases.
- Αν το command είναι CMD_START, αρχικά εξετάζεται με το **if (!robotStarted)** αν το όχημα έχει ξεκινήσει να κινείται και αν όχι, μέσω της **robotStarted = true;**, υποδηλώνεται πως το όχημα ξεκίνησε να κινείται. Με το **currentState = MOVE_FORWARD;**, το state του οχήματος δηλώνεται ως MOVE_FORWARD (command της **switch (currentState)** που θα διατυπωθεί αργότερα) και το όχημα ξεκινά να κινείται. Τέλος, μέσω της **stateStartMillis = millis()** και συγκεκριμένα της millis(), ξεκινά να μετριέται σε ms πόση ώρα θα είναι στο state MOVE_FORWARD το όχημα.
- Για τα command CMD_STOP, CMD_ULTRASONIC_TEST και CMD_IR_TEST ισχύουν οι ίδιες αρχές, εκτός από τα states τα οποία είναι αντίστοιχα stopMotors(), currentState = ULTRASONIC_TEST και currentState = IR_TEST.
- Το case **default** υπάρχει σε περίπτωση που δωθεί λάθος φωνητική εντολή απο το χρήστη.

```
void processCommand(String command) {
  int cmd = parseCommand(command); // Parse the received command
  switch (cmd) {
    case CMD_START:
      Serial.println("Received start command.");
      if (!robotStarted) {
        robotStarted = true;
        Serial.println("Robot starting...");
        currentState = MOVE_FORWARD;
        stateStartMillis = millis();          // Record the current time
      }
      break;
    case CMD_STOP:
      Serial.println("Received stop command.");
      if (robotStarted) {
        robotStarted = false;
        Serial.println("Robot stopping...");
        stopMotors();
      }
      break;
```

Εικόνα 47. *Συνάρτηση* ***void processCommand(String command)***

```
    case CMD_ULTRASONIC_TEST:
      Serial.println("Received Ultrasonic Test command.");
      if (!robotStarted) {
        robotStarted = true;
        currentState = ULTRASONIC_TEST;
      }
      break;
    case CMD_IR_TEST:
      Serial.println("Received IR Test command.");
      if (!robotStarted) {
        robotStarted = true;
        currentState = IR_TEST;
      break;
    default:
      Serial.print("Invalid command: ");
      break;
    }
  }
}
```

Εικόνα 48. *Συνέχεια της συνάρτησης* ***void processCommand(String command)***

Για τη δεύτερη συνάρτηση:

- **int parseCommand(String command) (**Εικόνα 49): συνάρτηση με όνομα parseCommand η οποία μετατρέπει το command σε ακέραιο αριθμό και αποθηκεύεται στο cmd. Οι ακέραιοι αντιπροσωπεύουν τις εντολές από το enum Commands.
- Αν δωθεί η φωνητική εντολή **''final test''** από το χρήστη, θα ενεργοποιηθεί το CMD_START.
- Αν δωθεί η φωνητική εντολή **''stop''** από το χρήστη, θα ενεργοποιηθεί το CMD_STOP.
- Αν δωθεί η φωνητική εντολή **''ultrasonic test''** από το χρήστη, θα ενεργοποιηθεί το CMD_ULTRASONIC_TEST.
- Αν δωθεί η φωνητική εντολή **''ir test''** από το χρήστη, θα ενεργοποιηθεί το CMD_IR_TEST.
- Αν δωθεί οποιαδήποτε άλλη φωνητική εντολή, θα ενεργοποιηθεί το CMD_INVALID (default).

```
int parseCommand(String command) {
  if (command == "final test") {
    return CMD_START;
  } else if (command == "stop") {
    return CMD_STOP;
  }else if (command == "ultrasonic test") {
      return CMD_ULTRASONIC_TEST;
  } else if (command == "ir test") {
     return CMD_IR_TEST;
  } else {
    return CMD_INVALID;
  }
}
```

Εικόνα 49*. Συνάρτηση* ***int parseCommand(String command)***

4.3.6 Συνάρτηση updateState και void loop

Η συνάρτηση **updateState()** αποτελεί το πιο σημαντικό κομμάτι του προγράμματος, ο σκοπός της διπλωματικής, καθώς διαχειρίζεται τη συμπεριφορά του οχήματος ενημερώνοντας την κατάστασή του με βάση τις ενδείξεις των αισθητήρων και τον χρόνο που έχει παρέλθει. Η συνάρτηση αναπτύσσεται ως εξής:

- Αρχικά χρησιμοποιείται η **unsigned long currentMillis = millis();** (Εικόνα 49*)* για να καταγράψει τον τρέχοντα χρόνο από την έναρξη της εκτέλεσης της συνάρτησης καθώς και η **int distance = readUltrasonicDistance();** έτσι ώστε να ληφθεί η απόσταση του πιο κοντινού εμποδίου από τον αισθητήρα υπερήχων.
- Χρησιμοποιείται μια **switch** για τις διαφορετικές καταστάσεις του οχήματος.
- Η πρώτη κατάσταση (case) είναι η **case ULTRASONIC_TEST** (Εικόνα 50*)*, η οποία καλείται όταν ειπωθεί από το χρήστη μέσω Bluetooth η έκφραση ‘’ultrasonic test’’. Μέσω της **if (currentMillis - stateStartMillis >= 100)**, γίνεται έλεγχος από τον αισθητήρα υπερήχων για εμπόδια ενώ κινείται ευθεία και αν εντοπίσει εμπόδιο στα 40cm (**OBST_DISTANCE**), τότε θα σταματήσει ακαριαία.

```
void updateState() {
  unsigned long currentMillis = millis();  // Get the current time
  int distance = readUltrasonicDistance(); // Read the distance from the ultrasonic sensor

  switch (currentState) {
    case ULTRASONIC_TEST:
      if (currentMillis - stateStartMillis >= 100) {  // Check every 100ms
        moveForward(currentSpeed);
        if (distance < OBST_DISTANCE) {
          Serial.print("Distance(cm): ");
          Serial.println(distance);
          stopMotors();
          robotStarted = false;
        }
      }
      break;
```

Εικόνα 50. *Συνάρτησεις* ***updateState()*** *και* ***readUltrasonicDistance()****, μεταβλητή* ***millis()*** *και* ***case ULTRASONIC_TEST***

- Η δεύτερη κατάσταση είναι η **case IR_TEST** (Εικόνα 51), η οποία καλείται όταν ειπωθεί από το χρήστη μέσω Bluetooth η έκφραση ‘’ir test’’. Αρχικά το όχημα κινείται ευθεία και μέσω της συνάρτησης **readIRSensor**, όταν και οι δύο αισθητήρες υπέρυθρων εντοπίσουν εμπόδια, το όχημα θα σταματήσει ακαριαία.

```
case IR_TEST:
  moveForward(currentSpeed);
  int rightIRValue = readIRSensor(RIGHT_IR_PIN);
  int leftIRValue = readIRSensor(LEFT_IR_PIN);
  if (leftIRValue < IR_THRESHOLD && rightIRValue < IR_THRESHOLD) {
    stopMotors();
    robotStarted = false;
  }
  break;
```

Εικόνα 51. case IR_TEST

- Η τρίτη κατάσταση είναι η **case MOVE_FORWARD** (Εικόνα 52), η οποία καλείται όταν ειπωθεί από το χρήστη μέσω Bluetooth η έκφραση ‘’final test’’. Το όχημα θα ξεκινήσει να κινείται ευθεία με τη ταχύτητα **currentSpeed** και ύστερα από 500ms μέσω της **delay(500)** θα μειωθεί στη ταχύτητα **newspeed** και θα γίνει εναλλαγή της κατάστασης σε **MOVE_FORWARD_WITH_NEWSPEED**.

```
case MOVE_FORWARD:
    moveForward(currentSpeed);
    delay(500);
    slowDownToSpeed(newspeed);
    currentState = MOVE_FORWARD_WITH_NEWSPEED;
    break;
```

Εικόνα 52. case MOVE_FORWARD

- Η κατάσταση **MOVE_FORWARD_WITH_NEWSPEED** (Εικόνα 53), όπως και η **case ULTRASONIC_TEST**, ενεργοποιεί το όχημα να σκανάρει κάθε 100ms για πιθανό εμπόδιο στα 40cm και όταν εντοπιστεί, ελλατώνεται η ταχύτητα στη minSpeed και γίνεται εναλλαγή της κατάστασης σε **AVOID_OBSTACLE**.

```
case MOVE_FORWARD_WITH_NEWSPEED:
  if (currentMillis - stateStartMillis >= 100) {  // Check every 100ms
    stateStartMillis = currentMillis; // Reset the state start time
    Serial.print("Distance: ");
    Serial.println(distance);
    if (distance < OBST_DISTANCE) {
      slowDownToSpeed(minSpeed);
      currentState = AVOID_OBSTACLE;
      stateStartMillis = currentMillis; // Reset the state start time
    }
  }
  break;
```

Εικόνα 53. case MOVE_FORWARD_WITH_NEWSPEED

- Η κατάσταση **AVOID_OBSTACLE** (Εικόνα 54) ελέγχει για πιθανά εμπόδια αριστερά ή δεξιά του οχήματος μετά από 100ms του εντοπισμού εμποδίου από τον αισθητήρα υπέρηχων. Ο έλεγχος αυτός πραγματοποιείται για να κάνει ελιγμό από την ανάποδη πλευρά και να μην συγκρουστεί με εμπόδιο

κατά τη διάρκεια του πρώτου ελιγμού. Για τον έλεγχο αυτό χρησιμοποιείται η συνάρτηση **readIRSensor**. Αν δεν υπάρχει εμπόδιο, τότε το όχημα θα πραγματοποιήσει τυχαίο ελιγμό. Αν το όχημα κάνει αριστερό ελιγμό, τότε γίνεται εναλλαγή της κατάστασης σε **MANEUVER_LEFT** και αν το όχημα κανει δεξιό ελιγμό τότε γίνεται εναλλαγή σε **MANEUVER_RIGHT**.

```
case AVOID_OBSTACLE:
if (currentMillis - stateStartMillis >= 100) {
 int rightIRValue = readIRSensor(RIGHT_IR_PIN);
 int leftIRValue = readIRSensor(LEFT_IR_PIN);
  if (rightIRValue < IR_THRESHOLD) {
    currentState = MANEUVER_LEFT;
  } else if (leftIRValue < IR_THRESHOLD) {
    currentState = MANEUVER_RIGHT;
  } else {
      if (random(2) == 0) {              // Randomly choose left or right maneuver
      currentState = MANEUVER_RIGHT;
    } else {
      currentState = MANEUVER_LEFT;
    }
  }
  stateStartMillis = currentMillis;   // Reset the state start time
}
  break;
```

Εικόνα 54. case AVOID_OBSTACLE

- Στη κατάσταση **MANEUVER_LEFT** (Εικόνα 55), χρησιμοποιείται η συνάρτηση **maneuverLeft();** για να γίνει ο αριστερός ελιγμός και ο σερβοκινητήρας στρίβει προς τη κατεύθυνση του εμποδίου, ως οπτική ένδειξη για το παρατηρητή. Γίνεται εναλλαγή της κατάστασης σε **CHECK_LEFT**. Αντίστοιχα στη κατάσταση **MANEUVER_RIGHT** (Εικόνα 55), χρησιμοποιείται η συνάρτηση **maneuverRight();** για να γίνει ο δεξιός ελιγμός και ο σερβοκινητήρας στρίβει προς τη κατεύθυνση του εμποδίου. Γίνεται εναλλαγή της κατάστασης σε **CHECK_RIGHT**.

```
case MANEUVER_LEFT:
  maneuverLeft();
  moveServo(70);
  currentState = CHECK_LEFT;
  stateStartMillis = currentMillis;    // Reset the state start time
  break;

case MANEUVER_RIGHT:
  maneuverRight();
  moveServo(110);
  currentState = CHECK_RIGHT;
  stateStartMillis = currentMillis;     // Reset the state start time
  break;
```

Εικόνα 55. case MANEUVER_LEFT και case MANEUVER_RIGHT

- Στη κατάσταση **CHECK_LEFT** (Εικόνα 56), μέσω της **if (currentMillis - stateStartMillis >= 1200)**, ο αριστερός ελιγμός σταματά μετά από 1200ms. Μέσω της **readIRSensor** πραγματοποιείται έλεγχος από το δεξιό IR αισθητήρα και αν εντοπιστεί εμπόδιο, το όχημα θα κινηθεί με μειωμένη

ταχύτητα **minSpeed** και θα γίνει εναλλαγή της κατάστασης σε **MOVE_FORWARD_WHILE_CHECKING_LEFT**. Αν δεν εντοπιστεί εμπόδιο, θα γίνει εναλλαγή της κατάστασης σε **CORRECT_LEFT**. Παρομοίως για τη κατάσταση **CHECK_RIGHT** (Εικόνα 56), ο δεξιός ελιγμός σταματά μετά από 1200ms. Μέσω της **readIRSensor** πραγματοποιείται έλεγχος από τον αριστερό IR αισθητήρα και αν εντοπιστεί εμπόδιο, το όχημα θα κινηθεί με μειωμένη ταχύτητα **minSpeed** και θα γίνει εναλλαγή της κατάστασης σε **MOVE_FORWARD_WHILE_CHECKING_RIGHT**. Αν δεν εντοπιστεί εμπόδιο, θα γίνει εναλλαγή της κατάστασης σε **CORRECT_ RIGHT**.

```
case CHECK_LEFT:
  if (currentMillis - stateStartMillis >= 1200) { // Wait 1200ms before checking
    int rightIRValue = readIRSensor(RIGHT_IR_PIN);
    if (rightIRValue < IR_THRESHOLD) {
      moveForward(minSpeed);
      currentState = MOVE_FORWARD_WHILE_CHECKING_LEFT;
    } else {
      currentState = CORRECT_LEFT;
    }
    stateStartMillis = currentMillis;   // Reset the state start time
  }
  break;

case CHECK_RIGHT:
  if (currentMillis - stateStartMillis >= 1300) { // Wait 1300ms before checking
    int leftIRValue = readIRSensor(LEFT_IR_PIN);
    if (leftIRValue < IR_THRESHOLD) {
      moveForward(minSpeed);
      currentState = MOVE_FORWARD_WHILE_CHECKING_RIGHT;
    } else {
      currentState = CORRECT_RIGHT;
    }
    stateStartMillis = currentMillis;   // Reset the state start time
  }
  break;
```

Εικόνα 56. case CHECK_LEFT και case CHECK_RIGHT

- Στη κατάσταση **MOVE_FORWARD_WHILE_CHECKING_LEFT** (Εικόνα 57), μέσω της **if (currentMillis - stateStartMillis >= 100)**, γίνεται έλεγχος από το δεξιό IR και τη στιγμή που παύει να υπάρχει εμπόδιο, γίνεται εναλλαγή της κατάστασης σε **CORRECT_LEFT**. Παρομοίως για τη κατάσταση **MOVE_FORWARD_WHILE_CHECKING_RIGHT** (Εικόνα 57), γίνεται έλεγχος από το αριστερό IR και τη στιγμή που παύει να υπάρχει εμπόδιο, γίνεται εναλλαγή της κατάστασης σε **CORRECT_RIGHT**.

```
case MOVE_FORWARD_WHILE_CHECKING_LEFT:
  if (currentMillis - stateStartMillis >= 100) { // Check every 100ms
    int rightIRValue = readIRSensor(RIGHT_IR_PIN);
    if (rightIRValue >= IR_THRESHOLD) {
      currentState = CORRECT_LEFT;
      stateStartMillis = currentMillis; // Reset the state start time
    }
  }
  break;

case MOVE_FORWARD_WHILE_CHECKING_RIGHT:
  if (currentMillis - stateStartMillis >= 100) { // Check every 100ms
    int leftIRValue = readIRSensor(LEFT_IR_PIN);
    if (leftIRValue >= IR_THRESHOLD) {
      currentState = CORRECT_RIGHT;
      stateStartMillis = currentMillis; // Reset the state start time
    }
  }
  break;
```

Εικόνα 57. case MOVE_FORWARD_WHILE_CHECKING_LEFT και case MOVE_FORWARD_WHILE_CHECKING_RIGHT

- Στη κατάσταση **CORRECT_LEFT** (Εικόνα 58), το όχημα πραγματοποιεί δεξιό ελιγμό μέσω της **maneuverRight()**, ο σερβοκινητήρας ευθυγραμμίζεται και γίνεται εναλλαγή της κατάστασης σε **ALIGN_LEFT**. Παρομοίως για τη κατάσταση **CORRECT_RIGHT** (Εικόνα 58), το όχημα πραγματοποιεί αριστερό ελιγμό μέσω της **maneuverLeft()**, ο σερβοκινητήρας ευθυγραμμίζεται και γίνεται εναλλαγή της κατάστασης σε **ALIGN_RIGHT**.

```
case CORRECT_LEFT:
  maneuverRight();                  // Perform right maneuver to correct
  moveServo(90);
  currentState = ALIGN_LEFT;
  stateStartMillis = currentMillis;    // Reset the state start time
  break;

case CORRECT_RIGHT:
  maneuverLeft();                   // Perform left maneuver to correct
  moveServo(90);
  currentState = ALIGN_RIGHT;
  stateStartMillis = currentMillis;    // Reset the state start time
  break;
```

Εικόνα 58. case CORRECT_LEFT και case CORRECT_RIGHT

- Στη κατάσταση **ALIGN_LEFT** (Εικόνα 59), μέσω της **if (currentMillis - stateStartMillis >= 1400)**, το όχημα ολοκληρώνει το δεξιό ελιγμό μετά από 1400ms και ευθυγραμμίζει τη πορεία του, έχοντας ξεπεράσει το εμπόδιο που εντοπίστηκε. Στη κατάσταση **ALIGN_RIGHT** (Εικόνα 59), μέσω της **if (currentMillis - stateStartMillis >= 1300)**, το όχημα ολοκληρώνει τον αριστερό ελιγμό μετά από 1300ms και ευθυγραμμίζει τη πορεία του, έχοντας ξεπεράσει το εμπόδιο που εντοπίστηκε. Και στις δύο καταστάσεις γίνεται εναλλαγή στην **MOVE_FORWARD**.

```
case ALIGN_LEFT:
  if (currentMillis - stateStartMillis >= 1400) { // Wait 1400ms for alignment
    speedUpToSpeed(newspeed);
    currentState = MOVE_FORWARD;
    stateStartMillis = currentMillis;            // Reset the state start time
  }
  break;

case ALIGN_RIGHT:
  if (currentMillis - stateStartMillis >= 1300) { // Wait 1300ms for alignment
    speedUpToSpeed(newspeed);
    currentState = MOVE_FORWARD;
    stateStartMillis = currentMillis;            // Reset the state start time
  }
  break;
```

Εικόνα 59. case ALIGN_LEFT και case ALIGN_RIGHT

Η τελευταία συνάρτηση του προγράμματος είναι η **void loop()**, (Εικόνα 60) η κεντρική συνάρτηση η οποία εκτελεί διαδοχικούς βρόχους. Η συνάρτηση στο πρόγραμμα είναι η εξής:

- Αρχικά χρησιμοποιείται η **mySerial.available** για να ελεγχθεί αν υπάρχουν διαθέσιμα δεδομένα από το Bluetooth για να ‘’διαβαστούν’’.
- Μέσω της **String command = mySerial.readStringUntil('\n');**, ‘’διαβάζεται’’ η σειρά χαρακτήρων, δηλαδή κείμενο, το οποίο εστάλη από το Bluetooth και αποθηκεύεται στο **command**.
- Το **command.trim();** αφαιρεί τυχόν κενά στη σειρά χαρακτήρων που εστάλη για να μη δημιουργηθεί πρόβλημα κατά τη διάρκεια της επεξεργασίας των δεδομένων.
- Εκτελείται η συνάρτηση **processCommand(command)**.
- Τέλος, μέσω της **if (robotStarted)**, γίνεται έλεγχος αν το όχημα έχει ξεκινήσει να κινείται και αν ναι, εκτελείται η συνάρτηση **updateState()**.

```
void loop() {
  if (mySerial.available()) {     // Checks if data is available from Bluetooth
    String command = mySerial.readStringUntil('\n');  // Reads the command from Bluetooth
    command.trim();               // Trims the command to remove leading/trailing whitespaces
    processCommand(command);      // Passes the trimmed command to processCommand()
  }
  if (robotStarted) {             // Checks if the robot is started
    updateState();
  }
}
```

Εικόνα 60. Void loop()

5. Αποτελέσματα

Σε αυτό το κεφάλαιο γίνονται οι απαραίτητοι έλεγχοι για τη λειτουργία του οχήματος, αρχικά για το Bluetooth, στη συνέχεια για τους αισθητήρες IR και HC-SR04 και τέλος για τη τελική λειτουργία του.

5.1 Έλεγχος συνδεσιμότητας Bluetooth

Αρχικά, αφού έχει εγκατασταθεί το app του MIT App Inventor στο smartphone, ανοίγεται και επιλέγεται το **scan QR code**. Στη συνέχεια ανοίγεται το site της σελίδας και αφού επιλεχθεί η εφαρμογή που δημιουργήθηκε (**Voice_recognition**), επιλέγεται με τη σειρά **connect**, **AI Companion** και σκανάρεται το QR code. Ύστερα από λίγα δευτερόλεπτα η εφαρμογή θα πρέπει να έχει ανοίξει. Αφού δωθεί τροφοδοσία στο Arduino, το HC-05 αναβοσβήνει περίπου πέντε φορές το δευτερόλεπτο όπου σημαίνει πως είναι έτοιμο να συνδεθεί με συσκευή. Από τις ρυθμίσεις του smartphone αρχικά πραγματοποιείται pairing με το HC-05 και από την εφαρμογή επιλέγεται το Bluetooth. Αφού βρεθεί το HC-05, επιλέγεται και ύστερα από μερικά δευτερόλεπτα, το HC-05 αναβοσβήνει πιο αργά, περίπου δυο φορές σε 5 δευτερόλεπτα, πληροφορώντας το χρήστη πως η σύνδεση είναι επιτυχής.

5.2 Έλεγχος λειτουργίας IR αισθητήρων

Για τον έλεγχο των IR αισθητήρων, ο χρήστης επιλέγει στην εφαρμογή ‘’**Start Voice Recognition**’’ και επικοινωνεί τη φράση ‘’**ir test**’’. Το όχημα ξεκινά να κινείται ευθεία και μόλις και οι δύο IR αισθητήρες εντοπίσουν εμπόδια, το όχημα σταματά ακαριαία. Ο έλεγχος είναι επιτυχής. Ακολουθεί βίντεο του ελέγχου:

SCAN ME

5.3 Έλεγχος λειτουργίας αισθητήρα υπέρηχων

Για τον έλεγχο του αισθητήρα υπέρηχων, ο χρήστης επιλέγει στην εφαρμογή ‘’**Start Voice Recognition**’’ και επικοινωνεί τη φράση ‘’**ultrasonic test**’’. Το όχημα ξεκινά να κινείται ευθεία και μόλις ο αισθητήρας εντοπίσει εμπόδιο στα 20cm, το όχημα σταματά ακαριαία. Ο έλεγχος είναι επιτυχής. Ακολουθεί βίντεο του ελέγχου:

SCAN ME

5.4 Έλεγχοι τελικής λειτουργίας

Για τον πρώτο έλεγχο της τελικής λειτουργίας του οχήματος, ο χρήστης επιλέγει στην εφαρμογή ‘’**Start Voice Recognition**’’ και επικοινωνεί τη φράση ‘’**final test**’’. Το όχημα έχει τοποθετηθεί δίπλα σε τοίχο και μπροστά του υπάρχει ένα μικρό εμπόδιο με διαστάσεις 9x5x15.5cm. Το όχημα ξεκινά να κινείται και μόλις εντοπίσει το εμπόδιο μπροστά του, πραγματοποιεί ελιγμό αντίθετα του τοίχου και στη συνέχεια πραγματοποιεί αντίθετο ελιγμό, ολοκληρώνοντας την αποφυγή του εμποδίου. Για τον δεύτερο έλεγχο, η διαδικασία είναι η ίδια, αλλά με μεγαλύτερο εμπόδιο διαστάσεων 31x14x24cm. Και οι δύο έλεγχοι είναι επιτυχείς. Ακολουθούν βίντεο των ελέγχων:

6 Συμπεράσματα και μελλοντικές επεκτάσεις

Σε αυτό το κεφάλαιο γίνεται σχολιασμός των αποτελεσμάτων της λειτουργίας του οχήματος και γενικότερα τα συμπεράσματα που βγήκαν κατά τη διάρκεια υλοποίησης του καθώς και προτείνονται μελλοντικές προεκτάσεις για τη βελτίωση του.

6.1 Συμπεράσματα

Ο στόχος της παρούσας διπλωματικής ήταν να δημιουργηθεί ένα ρομποτικό όχημα το οποίο να προσομοιάζει, εν μέρει, ένα AMR και συγκεκριμένα την ιδιότητα που έχει να μπορεί να αποφεύγει εμπόδια ομαλά και αποτελεσματικά σε στενούς χώρους, όπως σε ένα ιατρικό περιβάλλον. Το project αυτό αποδείχτηκε μια ενδιαφέρουσα πρόκληση καθώς δεν προϋπήρχε εμπειρία πάνω σε παρόμοιες κατασκευές και χρειάστηκε ενδελεχής έρευνα και μελέτη πάνω σε βιβλία, άρθρα, παρόμοιες εργασίες και βίντεο. Τόσο σε θεωρητικό όσο σε πρακτικό επίπεδο, οι γνώσεις και οι δεξιότητες που αποκτήθηκαν ήταν πολύτιμες και αποτελούν γερές βάσεις για μελλοντικά project.

Κατά τη διάρκεια της συλλογής των εξαρτημάτων και τη συναρμολόγηση του οχήματος παρουσιάστηκαν κάποια μικροπροβλήματα, συγκεκριμένα χρειάστηκαν αντικαταστάσεις εξαρτημάτων (IR αισθητήρες και HC-05) και αλλαγές στους τρόπους τροφοδοσίας του Arduino αλλά και των κινητήρων καθώς υπολειτουργούσαν. Αφού πραγματοποιήθηκαν αυτές οι αλλαγές, το όχημα λειτουργούσε αρμονικά. Κατά τη διάρκεια του προγραμματισμού παρατηρήθηκαν τα εξής:

- Το HC-SR04 ανά διαστήματα λειτούργουσε ακριβώς όπως είχε σχεδιαστεί, δηλαδή εντοπισμός εμποδίου εντός των 40cm, αλλά ανά διαστήματα υπολειτούργουσε. Συγκεκριμένα, παρατηρήθηκε πως εντόπιζε εμπόδιο συνεχόμενα, ακόμη και όταν δεν υπήρχε αντικείμενο εντός 40cm και με τη

χρήση του Serial Monitor και μεθόδους debugging αποδείχτηκε πως ο αισθητήρας εντόπιζε συνεχόμενα εμπόδιο στα 0cm. Με την εφαρμογή ενός test code για το HC-SR04, το πρόβλημα διορθώθηκε. Ύστερα από αρκετούς ελέγχους, παρατηρήθηκε πως η εξάντληση των μπαταριών προκαλούσε την υπολειτουργία του αισθητήρα.

- Ύστερα από τη πετυχημένη σύνδεση του smartphone με το HC-05, παρατηρήθηκε πως ενώ η επικοινωνία των ‘’**ir test’’**, ‘’**ultrasonic test**’’, ‘’**final test**’’ και ‘’**stop**’’ ήταν σωστή, συχνά το όχημα δε ανταποκρινόταν και εμφάνιζε ‘’Invalid command’’ στο Serial Monitor ή αποσυνδεόταν. Δε μπόρεσε να βρεθεί λύση για το πρόβλημα, καθιστώντας το απρόβλεπτο κατά τη λειτουργία του.
- Η ταχύτητα του οχήματος ήταν άλλο ένα πρόβλημα που συναντήθηκε. Μέσω της **analogWrite()** η οποία κάνει χρήση του κύματος PWM, οι κινητήρες μπορούν να προγραμματιστούν να κινούνται με βάση τις τιμές 0-255, όπου στη περίπτωση των κινητήρων 0 σημαίνει ακίνητο όχημα και 255 μέγιστη ταχύτητα. Παρατηρήθηκε πως η ελάχιστη τιμή που μπορούσε να λάβει το όχημα για να περιστρέφονται οι κινητήρες ήταν 58 (μεταβλητή **currentSpeed**), μια ταχύτητα σχετικά μεγάλη για τα δεδομένα της διπλωματικής. Μια μικρότερη ταχύτητα ήταν επιθυμητή (όπως η newspeed και η minSpeed) αλλά οι κινητήρες δεν περιστρέφονταν σε αυτές τις ταχύτητες, κάνοντας μηχανικό θόρυβο κατά τη διάρκεια της δοκιμής. Βρέθηκε μια εναλλακτική λύση με τη χρήση των συναρτήσεων **slowDownToSpeed** και **speedUpToSpeed** όπου ενώ το όχημα ξεκινούσε να κινείται με τη **currentSpeed**, στη συνέχεια είτε σταδιακά ελάττωνε τη ταχύτητα του είτε επιτάχυνε, όπου λειτούργησε ιδανικά.
- Τελευταίο πρόβλημα και πιο σημαντικό που παρατηρήθηκε ήταν η αδυναμία του οχήματος να μπορέσει να κινηθεί εντελώς ευθεία. Ύστερα από τη διάγνωση του προβλήματος, βρέθηκε πως οι τάσεις στα outputs του L298N ήταν ελαφριά διαφορετικές, με αποτέλεσμα ο ένας κινητήρας να έχει μεγαλύτερη περιστροφική ταχύτητα από τον άλλο και το όχημα όταν ξεκινούσε να κινείται, έπαιρνε μια κλίση προς τα αριστερά. Η διάγνωση του προβλήματος υπήρξε ενδελεχής, όπως αλλαγή τυχόν ελαττωματικών καλωδίων, αλλαγή της τροφοδοσίας, αλλαγή του L298N αλλά και αλλαγή των ίδιων των κινητήρων, δίχως αποτέλεσμα. Έγιναν ρυθμίσεις μέσω του προγράμματος και συγκεκριμένα μέσω του σήματος PWM, όπου είναι δυνατή η ρύθμιση της ταχύτητας του κάθε κινητήρα ξεχωριστά, αλλά επίσης δίχως αποτέλεσμα. Εν τέλει, το συμπέρασμα είναι πως οι DC κινητήρες δεν ενδείκνυται να χρησιμοποιηθούν σε project οχημάτων προγραμματισμένα να κινούνται με χαμηλές ταχύτητες καθώς παρουσιάζουν το πρόβλημα που αναφέρθηκε, ενώ σε πολύ μεγαλύτερες ταχύτητες το πρόβλημα παύει να υπάρχει.

6.2 Μελλοντικές επεκτάσεις

Όσον αφορά την εξέλιξη της παρούσας διπλωματικής, μια σημαντική αλλαγή που θα βελτίωνε σε μεγάλο βαθμό τη λειτουργία του οχήματος θα ήταν να χρησιμοποιηθούν πατενταρισμένοι σερβοκινητήρες, δηλαδή σερβοκινητήρες που μπορούν να ολοκληρώσουν μια στροφή 360°, αντί για τους DC κινητήρες. Η αλλαγή αυτή θα εξάλειφε το πρόβλημα της κλίσης που λαμβάνει το όχημα όταν ξεκινά να κινείται καθώς οι σερβοκινητήρες είναι

σχεδιασμένοι να περιστρέφονται σε χαμηλές ταχύτητες. Ταυτοχρόνως, το όχημα θα γινόταν πιο ακριβές στις κινήσεις του, πιο αθόρυβο και το L298N θα μπορούσε να αφαιρεθεί από το όχημα καθώς οι σερβοκινητήρες μπορούν να λετουργήσουν με απευθείας τροφοδοσία από το Arduino αλλά και να ελεγχθούν μέσω των σημάτων PWM.

Μια άλλη αλλαγή που θα μπορούσε να γίνει θα ήταν αντί για τον αισθητήρα υπέρηχων να χρησμοποιηθεί ένας οπτικός ανιχνευτής, όπως η κάμερα ενός smartphone. Οι δυνατότηρες που θα προσέφερε μια τέτοια αναβάθμιση είναι αμέτρητες, όπως ευρύτερο πεδίο όρασης, οπτική αναγνώριση αντικειμένων και χαρτογράφηση περιβάλλοντος.

Τέλος μια αλλαγή θα μπορούσε να ήταν η αντικατάσταση του HC-05 με ένα WiFi module, ειδικότερα αν χρησιμοποιηθεί ένα smartphone ως οπτικός ανιχνευτής, για σταθερότερη λειτουργία και ταχύτερη μεταφορά δεδομένων μεγαλύτερου όγκου.

Βιβλιογραφία

[1] Subir Kumar Saha (2014), "Introduction to Robotics, 2e", *McGraw Hill Education (India) Private Limited,* **ISBN 978-93-3290-280-0**

[2] Jahn U., Heß, D., Stampa, M., Sutorma, A., Röhrig, C., Schulz, P., & Wolff, C. (2020), "A Taxonomy for Mobile Robots: Types, Applications, Capabilities, Implementations, Requirements, and Challenges", *Robotics*, ***9,*** 4, p. 7. https://doi.org/10.3390/robotics9040109

[3] Rigatos Gerasimos, Busawon Krishna (2018), "Robotic manipulators and vehicles", *Springer* https://link.springer.com/book/10.1007/978-3-319-77851-8

[4] Giuseppe Fragapane, Dmitry Ivanov, Mirco Peron, Fabio Sgarbossa, Jan Ola Strandhagen (2020), "Increasing flexibility and productivity in Industry 4.0 production networks with autonomous mobile robots and smart intralogistics", *Spinger Link,* https://doi.org/10.1007/s10479-020-03526-7 S.I.

[5] Ata Jahangir Moshayedi, Jinsong Li, Liefa Liao (2019), "AGV (automated guided vehicle) robot: Mission and obstacles in design and performance", *Journal of Simulation & Analysis of Novel Technologies in Mechanical Engineering,* **12,** 4, pp. 6-7.
https://jsme.khsh.iau.ir/article_669256_3a335ebc4024b8096eb12808cae4fd0e.pdf

[6] John-David Warren, Josh Adams, Harald Molle (2011), "Arduino Robotics", *Paul Manning,* **ISBN 978-1-4302-3184-4**

[7] N. Anju Latha, B. Rama Murthy, K. Bharat Kumar (2016), "Distance Sensing with Ultrasonic Sensor and Arduino", *International Journal of Advance Research, Ideas and Innovations in Technology,* **2,** 5, pp. 1-2.

[8] Margaux Baum, Jeri Freedman (2017), "The History of Robots and Robotics (Hands-on Robotics)", *Rosen Central,* **ISBN-13: 978-1499438925**

[9] González D, Romero L, Espinosa MdM, Domínguez M (2017): "An optimization design proposal of automated guided vehicles for mixed type transportation in hospital environments" *PLoS ONE,* **12, (5)**, pp 1-2.

[10] Marko Pedan, Milan Gregor, Dariusz Plinta (2017): "Implementation of Automated Guided Vehicle system in healthcare facility" *Procedia Engineering, **192,*** pp 665-666.

[11] Siegwart, R., Nourbakhsh, I. R., & Scaramuzza, D. (2011): "Introduction to Autonomous Mobile Robots". *MIT Press,* **ISBN: 9780262015356**

[12] Singh, G., Bajaj, R., & Rani, M. (2017): '' Obstacle Detection and Distance Measurement for Autonomous Vehicles Using Sensor Technology'', *International Journal of Innovative Research in Science, Engineering and Technology***, 6(4),** 6179-6183.

[13] Siciliano, B., Sciavicco, L., Villani, L., & Oriolo, G. (2010): ''Robotics: Modelling, Planning, and Control'', *Springer,* **ISBN 978-1-84628-641-4**

[14] Dario, P., & Guglielmelli, E. (2013), *Robotic Assistive Technologies for the Disabled and Elderly*. *Annual Review of Biomedical Engineering*, **15**, 149–173.

[15] Banzi M. and Shiloh M. (2022): "Getting started with Arduino, 4th Edition", *Make Community, LLC,* **ISBN 978-1-680-45693-6**

[16] Hughes J.M. (2016): "Arduino: A Technical Reference", *O'Reilly Media,* **ISBN 978-1-491-92176-0**

[17] Correll N, Hayes B, Heckman C, and Roncone A. (2022): "Introduction to Autonomous Robots: Mechanisms, Sensors, Actuators, and Algorithms", *MIT Press*, **ISBN 978-0-262-37295-4**
[18] Scherz P, Monk S. (2016): "Practical Electronics for Inventors, 4th Edition", *McGraw-Hill Education*, **ISBN 978-1-25-958754-2**
[19] McComb, Gordon. (2013): "Arduino Robot Bonanza", McGraw-Hill Education, **ISBN 978-0-07-178278-4**
[20] Warren, John-David C. (2011): "Arduino Robotics", Apress, **ISBN 978-1-4302-3184-4**
[21] WIKIPEDIA, ''WIKIPEDIA'', [Ιστοσελίδα, πρόσβαση 2023 12 12].

Βιβλιογραφία εικόνων

[I] https://en.wikipedia.org/wiki/Leonardo%27s_robot
[II] https://collection.sciencemuseumgroup.org.uk/objects/co53901/william-thomsons-tide-predicting-machine-1872
[III] https://www.ri.cmu.edu/robot/dante-ii/
[IV] https://en.wikipedia.org/wiki/Arduino
https://www.why.gr/%CE%BA%CE%B1%CF%84%CE%B1%CF%83%CF%84%CE%B7%CE%BC%CE%B1/open-hardware/arduino/arduino-main-boards/arduino-mega-2560-rev3/
https://www.cableworks.gr/ilektronika/arduino/boards/compatible-boards/arduino-nano-v3.0-w-atmega328p-oem/
[V] https://newbiely.com/tutorials/arduino-nano/arduino-nano-motor
https://grobotronics.com/dual-motor-driver-module-l298n.html
[VI] https://topelectronics.gr/electronics/sensors/ultrasonic/hc-sr04-ultrasonic-module-distance-for-arduino/
https://www.devobox.com/el/proximity/679-obstacle-avoidance-ir-sensor-module-gr.html
[VII] https://www.hackster.io/embeddedlab786/line-follower-and-obstacle-avoiding-robot-baa2bb
[VIII] https://grobotronics.com/bluetooth-module-for-arduino-hc05.html
[IX] https://grobotronics.com/arduino-uno-rev3.html?sl=en
[X] https://www.elecbee.com/en-31457-5Pcs-L298N-Dual-H-Bridge-Stepper-Motor-Driver-Board-for-Arduino-products-that-work-with-official-Arduino-boards
[XI] https://grobotronics.com/ultrasonic-sensor-sr04.html
[XII] https://www.devobox.com/el/proximity/679-obstacle-avoidance-ir-sensor-module-gr.html?skr_prm=WyI2ZWJkOTRlZC00MWY2LTRkYjktYTcyMi1kZmQ5MTAxNTk3YzEiLDE2OTgzNTA2NjAzODAseyJhcHBfdHlwZSI6IndlYiIsImNwIjoiYiIsInRhZ3MiOiIifV0
[XIII] https://grobotronics.com/bluetooth-module-for-arduino-hc05.html
[XIV] https://robotbits.co.uk/product/dagu-4wd-chassis-red/

Παράρτημα

Κώδικας ρομποτικού οχήματος

```
#include <NewPing.h>          //  NewPing library for maximising ultrasonic sensor capabilities
#include <Servo.h>            //  Servo library for controlling the servo motor
#include <SoftwareSerial.h>   //  SoftwareSerial library for Bluetooth communication

// Define pin numbers
#define ENA 6
#define IN1 10
#define IN2 9
#define IN3 8
#define IN4 7
#define ENB 5
#define SERVO_PIN 11
#define TRIG_PIN 4
#define ECHO_PIN 3
#define LEFT_IR_PIN A0
#define RIGHT_IR_PIN A1

// Define constants
#define MAX_DISTANCE 400      // Maximum distance for ultrasonic sensor in cm
#define OBST_DISTANCE 40      // Distance threshold to detect obstacle in cm
#define IR_THRESHOLD 500      // Threshold value for IR sensors

// Objects creation
Servo myServo;                // Create a Servo object
NewPing sonar(TRIG_PIN, ECHO_PIN, MAX_DISTANCE);  // Create a NewPing object
SoftwareSerial mySerial(12, 13); // RX, TX - Create a SoftwareSerial object for Bluetooth

// Define commands enum
enum Commands {
  CMD_START,
  CMD_STOP,
  CMD_ULTRASONIC_TEST,
  CMD_IR_TEST,
  CMD_INVALID
};

// Define states for non-blocking delay management
enum State {
  MOVE_FORWARD,
  AVOID_OBSTACLE,
  MANEUVER_LEFT,
  MANEUVER_RIGHT,
  CHECK_LEFT,
  CHECK_RIGHT,
  MOVE_FORWARD_WHILE_CHECKING_LEFT,
```

```
  MOVE_FORWARD_WHILE_CHECKING_RIGHT,
  CORRECT_LEFT,
  CORRECT_RIGHT,
  ALIGN_LEFT,
  ALIGN_RIGHT,
  ULTRASONIC_TEST,
  IR_TEST
};

// Initialize variables
unsigned long previousMillis = 0;  // Store the last time an action was performed
bool robotStarted = false;         // Track whether the robot is started
State currentState = MOVE_FORWARD;  // Initialize the current state to
MOVE_FORWARD
unsigned long stateStartMillis = 0; // Store the time when the state started

// Speed variables
int currentSpeed = 58;
int newspeed = 45;
const int minSpeed = 35;

void setup() {
  pinMode(ENA, OUTPUT);          // Set ENA as output
  pinMode(IN1, OUTPUT);          // Set IN1 as output
  pinMode(IN2, OUTPUT);          // Set IN2 as output
  pinMode(IN3, OUTPUT);          // Set IN3 as output
  pinMode(IN4, OUTPUT);          // Set IN4 as output
  pinMode(ENB, OUTPUT);          // Set ENB as output
  pinMode(SERVO_PIN, OUTPUT);     // Set SERVO_PIN as output
  myServo.attach(SERVO_PIN);      // Attach the servo to SERVO_PIN
  Serial.begin(9600);             // Start the Serial communication at 9600 baud rate
  mySerial.begin(9600);          // Start the SoftwareSerial communication at 9600 baud
rate
  Serial.println("Bluetooth is ready for connection. Connect to HC-05 with 1234 as
pairing key!");
}

void loop() {
  if (mySerial.available()) {     // Check if data is available from Bluetooth
    String command = mySerial.readStringUntil('\n');  // Read the command from
Bluetooth
    command.trim();               // Trim the command to remove leading/trailing
whitespaces
    processCommand(command);      // Pass the trimmed command to
processCommand()
  }
  if (robotStarted) {             // Check if the robot is started
    updateState();                // Update the robot state
  }
}

void processCommand(String command) {
  int cmd = parseCommand(command); // Parse the received command
  switch (cmd) {
```

```
    case CMD_START:
      Serial.println("Received start command.");
      if (!robotStarted) {
        robotStarted = true;
        Serial.println("Robot starting...");
        currentState = MOVE_FORWARD;
        stateStartMillis = millis();          // Record the current time
      }
      break;
    case CMD_STOP:
      Serial.println("Received stop command.");
      if (robotStarted) {
        robotStarted = false;
        Serial.println("Robot stopping...");
        stopMotors();
      }
      break;
    case CMD_ULTRASONIC_TEST:
      Serial.println("Received Ultrasonic Test command.");
      if (!robotStarted) {
        robotStarted = true;
        currentState = ULTRASONIC_TEST;
      }
      break;
    case CMD_IR_TEST:
      Serial.println("Received IR Test command.");
      if (!robotStarted) {
        robotStarted = true;
        currentState = IR_TEST;
      break;
    default:
      Serial.print("Invalid command: ");
      break;
    }
  }
}

int parseCommand(String command) {
  if (command == "final test") {
    return CMD_START;
  } else if (command == "stop") {
    return CMD_STOP;
  }else if (command == "ultrasonic test") {
      return CMD_ULTRASONIC_TEST;
  } else if (command == "ir test") {
     return CMD_IR_TEST;
  } else {
    return CMD_INVALID;
  }
}

int readUltrasonicDistance() {
  unsigned int uS = sonar.ping();       // Send a ping and get the ping time in microseconds
  int distance = uS / US_ROUNDTRIP_CM;  // Convert the ping time to distance in cm
```

```
  return distance;                    // Return the distance
}

int readIRSensor(int pin) {
  return analogRead(pin);             // Read the analog value from the specified IR sensor
pin
}

void updateState() {
  unsigned long currentMillis = millis();  // Get the current time
  int distance = readUltrasonicDistance(); // Read the distance from the ultrasonic sensor

  switch (currentState) {
    case MOVE_FORWARD:
      moveForward(currentSpeed);
      delay(1500);
      Serial.print("SUCCESS: ");
      slowDownToSpeed(newspeed);
      if (currentMillis - stateStartMillis >= 100) {  // Check every 100ms
        stateStartMillis = currentMillis; // Reset the state start time
        Serial.print("Distance: ");
        Serial.println(distance);
        if (distance < OBST_DISTANCE) {
          slowDownToSpeed(minSpeed);
          currentState = AVOID_OBSTACLE;
          stateStartMillis = currentMillis; // Reset the state start time
        }
      }
      break;

    case AVOID_OBSTACLE:
    if (currentMillis - stateStartMillis >= 100) {
     int rightIRValue = readIRSensor(RIGHT_IR_PIN);
     int leftIRValue = readIRSensor(LEFT_IR_PIN);
      if (rightIRValue < IR_THRESHOLD) {
        currentState = MANEUVER_LEFT;
      } else if (leftIRValue < IR_THRESHOLD) {
        currentState = MANEUVER_RIGHT;
      } else {
          if (random(2) == 0) {               // Randomly choose left or right maneuver
          currentState = MANEUVER_RIGHT;
        } else {
          currentState = MANEUVER_LEFT;
        }
      }
      stateStartMillis = currentMillis;    // Reset the state start time
    }
      break;

    case MANEUVER_LEFT:
      maneuverLeft();
      moveServo(70);
      currentState = CHECK_LEFT;
      stateStartMillis = currentMillis;     // Reset the state start time
```

```
  break;

case MANEUVER_RIGHT:
  maneuverRight();
  moveServo(110);
  currentState = CHECK_RIGHT;
  stateStartMillis = currentMillis;     // Reset the state start time
  break;

case CHECK_LEFT:
  if (currentMillis - stateStartMillis >= 1200) { // Wait 1200ms before checking
    int rightIRValue = readIRSensor(RIGHT_IR_PIN);
    if (rightIRValue < IR_THRESHOLD) {
      moveForward(minSpeed);
      currentState = MOVE_FORWARD_WHILE_CHECKING_LEFT;
    } else {
      currentState = CORRECT_LEFT;
    }
    stateStartMillis = currentMillis;   // Reset the state start time
  }
  break;

case CHECK_RIGHT:
  if (currentMillis - stateStartMillis >= 1300) { // Wait 1300ms before checking
    int leftIRValue = readIRSensor(LEFT_IR_PIN);
    if (leftIRValue < IR_THRESHOLD) {
      moveForward(minSpeed);
      currentState = MOVE_FORWARD_WHILE_CHECKING_RIGHT;
    } else {
      currentState = CORRECT_RIGHT;
    }
    stateStartMillis = currentMillis;   // Reset the state start time
  }
  break;

case MOVE_FORWARD_WHILE_CHECKING_LEFT:
  if (currentMillis - stateStartMillis >= 300) { // Check every 300ms
    int rightIRValue = readIRSensor(RIGHT_IR_PIN);
    if (rightIRValue >= IR_THRESHOLD) {
      currentState = CORRECT_LEFT;
      stateStartMillis = currentMillis; // Reset the state start time
    }
  }
  break;

case MOVE_FORWARD_WHILE_CHECKING_RIGHT:
  if (currentMillis - stateStartMillis >= 300) { // Check every 300ms
    int leftIRValue = readIRSensor(LEFT_IR_PIN);
    if (leftIRValue >= IR_THRESHOLD) {
      currentState = CORRECT_RIGHT;
      stateStartMillis = currentMillis; // Reset the state start time
    }
  }
  break;
```

```
    case CORRECT_LEFT:
      maneuverRight();                  // Perform right maneuver to correct
      moveServo(90);
      currentState = ALIGN_LEFT;
      stateStartMillis = currentMillis;    // Reset the state start time
      break;

    case CORRECT_RIGHT:
      maneuverLeft();                   // Perform left maneuver to correct
      moveServo(90);
      currentState = ALIGN_RIGHT;
      stateStartMillis = currentMillis;    // Reset the state start time
      break;

    case ALIGN_LEFT:
      if (currentMillis - stateStartMillis >= 1400) { // Wait 1200ms for alignment
        speedUpToSpeed(newspeed);
        currentState = MOVE_FORWARD;
        stateStartMillis = currentMillis;            // Reset the state start time
      }
      break;

    case ALIGN_RIGHT:
      if (currentMillis - stateStartMillis >= 1300) { // Wait 1300ms for alignment
        speedUpToSpeed(newspeed);
        currentState = MOVE_FORWARD;
        stateStartMillis = currentMillis;            // Reset the state start time
      }
      break;

    case ULTRASONIC_TEST:
      if (currentMillis - stateStartMillis >= 100) {
        moveForward(currentSpeed);
        if (distance < OBST_DISTANCE) {
          Serial.print("Distance(cm): ");
          Serial.println(distance);
          stopMotors();
          robotStarted = false;
        }
      }
      break;

    case IR_TEST:
      moveForward(currentSpeed);
      int rightIRValue = readIRSensor(RIGHT_IR_PIN);
      int leftIRValue = readIRSensor(LEFT_IR_PIN);
      if (leftIRValue < IR_THRESHOLD && rightIRValue < IR_THRESHOLD) {
        stopMotors();
        robotStarted = false;
      }
      break;
  }
}
```

```
void moveForward(int speed) {
  digitalWrite(IN1, HIGH);
  digitalWrite(IN2, LOW);
  digitalWrite(IN3, HIGH);
  digitalWrite(IN4, LOW);
  analogWrite(ENA, speed);
  analogWrite(ENB, speed);
}

void maneuverLeft() {
  digitalWrite(IN1, HIGH);
  digitalWrite(IN2, LOW);
  digitalWrite(IN3, HIGH);
  digitalWrite(IN4, LOW);
  analogWrite(ENA, 55);
  analogWrite(ENB, 35);
}

void maneuverRight() {
  digitalWrite(IN1, HIGH);
  digitalWrite(IN2, LOW);
  digitalWrite(IN3, HIGH);
  digitalWrite(IN4, LOW);
  analogWrite(ENA, 35);
  analogWrite(ENB, 55);
}

void stopMotors() {
  digitalWrite(IN1, LOW);
  digitalWrite(IN2, LOW);
  digitalWrite(IN3, LOW);
  digitalWrite(IN4, LOW);
  analogWrite(ENA, 0);
  analogWrite(ENB, 0);
}

void slowDownToSpeed(int targetSpeed) {
  while (currentSpeed > targetSpeed) {
    currentSpeed -= 1;
    moveForward(currentSpeed);
    delay(10);  // Delay adjustment for smoother or quicker transitions
  }
}

void speedUpToSpeed(int targetSpeed) {
  while (currentSpeed < targetSpeed) {
    currentSpeed += 1;
    moveForward(currentSpeed);
    delay(10);  // Delay adjustment for smoother or quicker transitions
  }
}

void moveServo(int angle) {
```

```
  myServo.write(angle);  // Move the servo to the specified angle
  delay(100);   // Wait for the servo to reach the desired position
}
```