ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΔΙΑΤΜΗΜΑΤΙΚΟ ΠΡΟΓΡΑΜΜΑ ΜΕΤΑΠΤΥΧΙΑΚΩΝ ΣΠΟΥΔΩΝ
«ΣΥΣΤΗΜΑΤΑ ΑΥΤΟΜΑΤΙΣΜΟΥ»

Μεταπτυχιακή Εργασία

ΤΙΤΛΟΣ

Ρομποτικό όχημα διαφορικής οδήγησης για αποφυγή εμποδίων σε τυχαία περιήγηση στον χώρο

Βασίλειος Τζίμας

Α.Μ.: 02119221

Επιβλέπων Καθηγητής: Δρ. Κωνσταντίνος Τζαφέστας

ΑΘΗΝΑ, Οκτώβριος 2022

ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΔΙΑΤΜΗΜΑΤΙΚΟ ΠΡΟΓΡΑΜΜΑ ΜΕΤΑΠΤΥΧΙΑΚΩΝ ΣΠΟΥΔΩΝ
«ΣΥΣΤΗΜΑΤΑ ΑΥΤΟΜΑΤΙΣΜΟΥ»

Μεταπτυχιακή Εργασία

ΤΙΤΛΟΣ

Ρομποτικό όχημα διαφορικής οδήγησης για αποφυγή εμποδίων σε τυχαία περιήγηση στον χώρο

Βασίλειος Τζίμας

Α.Μ.: 02119221

Επιβλέπων Καθηγητής: Δρ. Κωνσταντίνος Τζαφέστας

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την 31η Οκτωβρίου 2022

.....	.....	.....
Δρ. Κωνσταντίνος Τζαφέστας	*Δρ. Κωνσταντίνος Κυριακόπουλος*	*Δρ. Χαράλαμπος Ψυλλάκης*

ΑΘΗΝΑ, Οκτώβριος 2022

Περίληψη

Στο πλαίσιο της μεταπτυχιακής αυτής εργασίας μελετήθηκε το θεωρητικό υπόβαθρο ενός αυτοκινούμενου ρομπότ διαφορικής οδήγησης περιλαμβανόμενου της ευθείας και αντιστροφής κινηματικής καθώς και η οδομετρία. Επιπλέον μελετήθηκε και αναπτύχθηκε αλγόριθμος αποφυγής εμποδίων σε τέτοιου τύπου αυτοκινούμενου ρομπότ με τυχαία περιήγηση στον χώρο παρουσία εμποδίων.

Για τον σκοπό αυτό χρησιμοποιήθηκε μια υφιστάμενη πειραματική διάταξη όπου πραγματοποιήθηκαν οι κατάλληλες παρεμβάσεις για να γίνει πιο «έξυπνη», όπως με την αντικατάσταση του main-board με έναν επεξεργαστή τελευταίας γενιάς, για τον εντοπισμό εμποδίων χρησιμοποιήθηκαν πέντε υπέρυθροι αισθητήρες. Ο αλγόριθμος αναπτύχθηκε στην αντικειμενοστραφή γλώσσα προγραμματισμού python σε συνεργασία με το ρομποτικό σύστημα διαχείρισης (ROS).

Επίσης πραγματοποιήθηκε και προσομοίωση στο περιβάλλον προσομοίωσης του Gazebo για να αποφευχθεί ο επαναλαμβανόμενος έλεγχος στην πειραματική διάταξη για να ελαχιστοποιήσουμε το κίνδυνο ατυχήματος με αποτέλεσμα την καταστροφή της.

Abstract

In this master's thesis context, the theoretical background of a self-propelled differential drive robot was studied, including forward and inverse kinematics and odometry. In addition, an obstacle avoidance algorithm was researched and developed for this self-driven robot with random walking in the open space in the presence of obstacles.

For this purpose, an existing experimental setup was used where appropriate interventions were made to make it more "intelligent". For example, five infrared sensors were used to detect obstacles and replace the main board with a cutting-edge processor. The algorithm was developed in the object-oriented programming language python in collaboration with the robot operating system (ROS).

A simulation was also carried out in the Gazebo simulation environment to avoid repeated testing of the experimental setup to minimize the risk of an accident resulting in its destruction.

Ευχαριστίες

Ξεκινώντας θα ήθελα να αναφερθώ στο μεταπτυχιακό που φτάνει στο τέλος του. Ήταν πολύ σημαντικό για εμένα να κάνω αυτό το μεταπτυχιακό, γιατί θα με βοηθήσει να αναρριχηθώ σε υψηλότερους στόχους. Να κάνω κάτι που αγαπάω πολύ και να ξεπεράσω τα όρια του εαυτού μου. Τα μαθητικά μου χρόνια ήταν αρκετά δύσκολα, καθώς έχω δυσλεξία. Αυτό με δυσκόλευε αρκετά στα μαθήματα, αλλά πάντα ήθελα να βάζω στόχους και να τους πετυχαίνω. Κατάφερα να βρω τον εαυτό μου την περίοδο των φοιτητικών μου χρόνων, όταν ακόμα ήμουν στην Κοζάνη.

Το μεταπτυχιακό αυτό δεν ήταν μια τυχαία επιλογή. Ήταν κάτι που ήθελα να πραγματοποιήσω και η πρώτη προσπάθεια έγινε μόλις αποφοίτησα από την Κοζάνη ως ηλεκτρολόγος μηχανικός αλλά δεν έγινα δεκτός, καθώς δεν πληρούσα τα κριτήρια εισαγωγής. Όμως τα κατάφερα, μετά από πολλές δυσκολίες να πάρω το πτυχίο Β2 των Αγγλικών και να γίνω δεκτός στο μεταπτυχιακό που ήθελα να κάνω.

Αρχικά, θα ήθελα να ευχαριστήσω του καθηγητές του μεταπτυχιακού, καθώς και τη γραμματεία του ΔΠΜΣ. Ειδικότερα τον κ. Κωνσταντίνο Τζαφέστα επιβλέπων της διπλωματικής μου καθώς και τους βοηθούς του κ. Αθανάσιο Δομέτιο και κ. Γεώργιο Θανέλλα.

Σε αυτήν μου την προσπάθεια ήταν δίπλα μου άνθρωποι που με βοήθησαν, συγγενείς και μη. Θα αναφερθώ στην πολύτιμη στήριξη των γονιών μου. Του πατέρα μου Κωνσταντίνο και της μητέρας μου Σταματίας που με βοήθησαν οικονομικά και ψυχολογικά, ώστε να ολοκληρώσω με επιτυχία αυτή μου την προσπάθεια. Επίσης θα ήθελα να ευχαριστήσω τη σύντροφο μου Θεοδώρα για την στήριξη της και την προσπάθεια της να με καταλάβει στο πολύωρο διάβασμα μου. Κλείνοντας, δεν θα μπορούσα να μην αναφέρω την αδερφή μου Ευαγγελία και τα ανίψια μου Βίκτωρα και Κωνσταντίνο, που αν και μακριά, χαίρονται για μένα.

Είμαι πολύ χαρούμενος που κατάφερα να τελειοποιήσω το στόχο μου και θα ήθελα να τονίσω πως όλα μπορούμε να τα καταφέρουμε, αν το θελήσουμε.

Περιεχόμενα

Κατάλογος Εικόνων

Κατάλογος Πινάκων

1 Εισαγωγή

1.1 Σκοπός Εργασίας

Ο βασικός σκοπός της διπλωματικής εργασίας είναι η εκμάθηση των κινηματικών εξισώσεων των ρομπότ διαφορικής οδήγησης, δηλαδή, της ευθείας και αντίστροφης κινηματικής εξίσωσης, και για τον έλεγχο της θέσης του ρομπότ μέσω της οδομετρίας.

Για την πρακτική κατανόηση της θεωρίας πραγματοποιήθηκε τροποποίηση υπάρχουσας πειραματικής διάταξης όπου αναβαθμίστηκε ώστε να μπορεί να δεχτεί περισσότερες παραμετροποιήσεις όπως υποστήριξη λειτουργικού συστήματος και αισθητήρων κίνησης, επιτάχυνσης και εντοπισμού εμποδίων.

Ακόμα ένας από τους σκοπούς ήταν ότι στο πρακτικό μέρος υπήρξε η ενασχόληση με το hardware κομμάτι του ρομπότ, όπως η εκμάθηση με την θεωρία ηλεκτρονικών, με εργαλεία κατασκευής ενός ρομπότ, όπως η κατασκευή πλακέτας.

Επιπλέον, από το κομμάτι του software έγινε εξοικείωση και εκβάθυνση, στο λειτουργικό σύστημα των Ubuntu κυρίως των εντολών στο terminal, της γλώσσας προγραμματισμού Python, και τέλος η σημαντική εντριβή με το ROS και το Gazebo.

Τέλος το ρομπότ θα χρησιμοποιηθεί για εκπαιδευτικούς σκοπούς των φοιτητών από το εργαστήριο της ρομποτικής του τμήματος Ηλεκτρολόγων Μηχανικών.

1.2 Βιβλιογραφική ανασκόπηση

Οι εφαρμογές των ρομπότ διαφορικής οδήγησης ήταν οι πιο συνηθισμένες από τα παλιά χρόνια. Μερικές από αυτές αναφέρονται στην συνέχεια.

Cye robot

Το Cye robot είναι ένα ρομπότ διαφορικής οδήγησης με δύο τροχούς, το οποίο αρχικά αναπτύχθηκε να κάνει πλοήγηση με την μέθοδο dead-reckoning, το οποίο είχε μόνο encoders στους τροχούς ως αισθητήρες [9].

Khepera robot

Το Khepera είναι και αυτό ένα ρομπότ διαφορικής οδήγησης που αναπτύχθηκε για ερευνητικούς σκοπούς, workshop και σεμινάρια από το πολυτεχνείο της Λοζάνης στην Ελβετία το 1990 [10].

Robot Alice

Το robot Alice είναι ακόμα ένα ρομπότ διαφορικής οδήγησης, και αυτό αναπτύχθηκε στο πολυτεχνείο της Λοζάνης στην Ελβετία μεταξύ του 1998 με 2004 [11].

1.3 Δομή εργασίας

Η διπλωματική εργασία αποτελείται από έξι ενότητες, στο πρώτο κεφάλαιο έχουμε την εισαγωγή της, στο δεύτερο κεφάλαιο αναλύουμε το θεωρητικό υπόβαθρό της και την λειτουργία του ROS. Στο τρίτο κεφάλαιο συναντάμε το κομμάτι του hardware, στο τέταρτο κεφάλαιο συναντάμε το software όπου εκεί γίνεται ανάλυση των πακέτων του ROS και κάποιες λεπτομέρειες για το λειτουργικό σύστημα των Ubuntu. Στο πέμπτο κεφάλαιο βλέπουμε την λειτουργία του Gazebo και τέλος στο έκτο κεφάλαιο αναλύουμε τις πειραματικές διαδικασίες που ακολουθήσαμε.

2 Θεωρητικό υπόβαθρο

2.1 Κινηματικά μοντέλα ρομπότ

Η κινηματική είναι η μελέτη της κίνησης ενός αντικειμένου χωρίς να εξετάζουμε τα αίτια του. Με την έννοια κίνηση αναφερόμαστε στην ταχύτητα και τις επιταχύνσεις. Και με τον όρο αίτια αναφερόμαστε σε αυτό που προκαλεί αυτή την κίνηση, όπως μια δύναμη ή μια ροπή.

Στα μαθηματικά, κατανοούμε ένα μοντέλο ως μια απλοποιημένη, μαθηματική αναπαράσταση ενός πραγματικού/φυσικού συστήματος. Το μοντέλο είναι μια απλοποίηση, με αυτό εννοούμε ότι, όταν δημιουργούμε ένα μοντέλο, πρέπει να απλοποιήσουμε ή να υποθέσουμε ορισμένα πράγματα και πρέπει να είναι έτσι γιατί είναι αδύνατο να αποτυπωθεί όλη η πολυπλοκότητα ενός πραγματικού/φυσικού συστήματος. Επίσης, ένα μοντέλο είναι μια μαθηματική αναπαράσταση, δηλαδή είναι ένα μοντέλο από ένα σύνολο μαθηματικών εξισώσεων, όπου αυτές οι εξισώσεις προορίζονται να περιγράψουν τη συμπεριφορά ενός πραγματικού/φυσικού συστήματος.

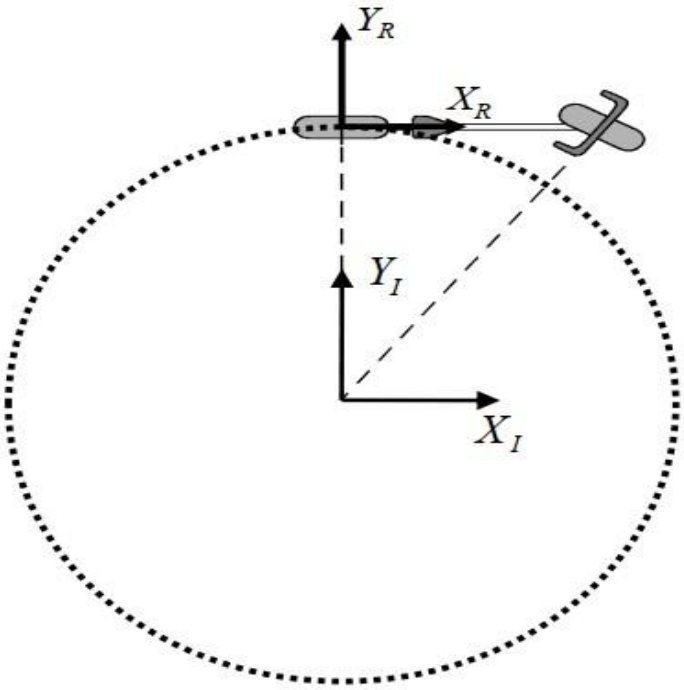
Έχοντας όλα αυτά κατά νου, μπορούμε στη συνέχεια να συμπεράνουμε τα εξής: το κινηματικό μοντέλο ενός κινητού ρομπότ θα είναι ένα σύνολο εξισώσεων που θα περιγράφουν τη συμπεριφορά της κίνησης ενός κινητού ρομπότ.

Υπάρχουν δυο διαφορετικές κατηγορίες, τα ολονομικά και τα μη-ολονομικά συστήματα. Στην ρομποτική ο όρος αναφέρεται συγκεκριμένα στους κινηματικούς περιορισμούς του πλαισίου του ρομπότ [3].

2.1.1 Ολονομικά συστήματα ρομπότ

Στα ολονομικά συστήματα οι διαφορικοί περιορισμοί είναι ολοκληρώσιμοι. Αυτά μπορούν να εκφραστούν ως περιορισμοί στην στάση του ρομπότ, δηλαδή, το ρομπότ είναι σε θέση να κινείται αμέσως προς οποιαδήποτε κατεύθυνση στο χώρο των βαθμών ελευθερίας του.

Ένα παράδειγμα ενός ολονομικού ρομπότ είναι ένα ποδήλατο με σταθερό σύστημα οδήγησης [Εικόνα 1].



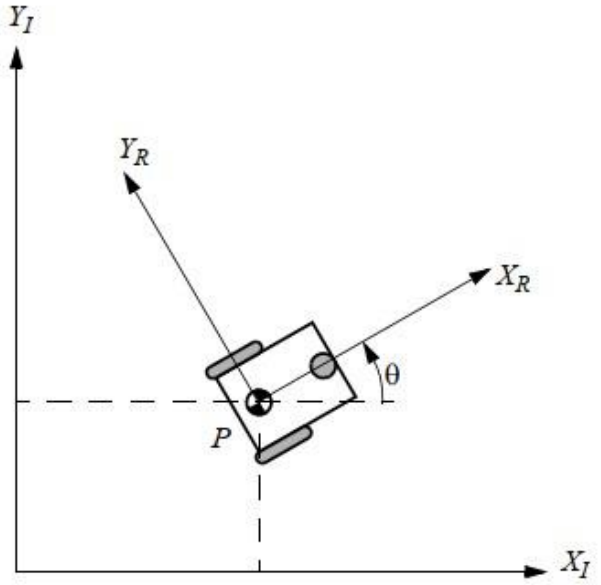
Εικόνα 1: Ολονομικό σύστημα (ποδήλατο)
[Παρουσίαση από το Autonomous Systems Lab του ETH Ζυρίχης]

Αυτό αποτελείται από δύο μη κατευθυνόμενους τροχούς, όπου το περιβάλλον εργασίας του ρομπότ περιορίζεται σε έναν βαθμό ελευθερίας, δηλαδή, σε ένα κύκλο. Οι διαφορικοί περιορισμοί μπορούν να εκφραστούν ως περιορισμοί της θέσης του.

2.1.2 Μη-ολονομικά συστήματα ρομπότ

Στα μη-ολονομικά συστήματα οι διαφορικοί περιορισμοί δεν είναι ολοκληρώσιμοι, δεν υπάρχει τρόπος να εκφράσουμε αυτά ως περιορισμούς της στάσης του ρομπότ. Το ρομπότ δεν είναι ικανό να κινηθεί στιγμιαία προς κάθε κατεύθυνση στον χώρο των βαθμών ελευθερίας του.

Το παράδειγμα ενός μη-ολονομικού ρομπότ είναι το ρομπότ διαφορικής οδήγησης που αναλύουμε στην διπλωματική εργασία.



Εικόνα 2: Μη-ολονομικό σύστημα (ρομπότ διαφορικής οδήγησης)
[Βιβλίο: Introduction to Autonomous Mobile Robots Second Edition, The MIT Press]

Το ρομπότ διαφορικής οδήγησης έχει δύο μη κατευθυνόμενους τροχούς στο ίδιο άξονα ευθυγραμμισμένους. Το περιβάλλον εργασίας του ρομπότ περιλαμβάνει όλες τις στάσεις του στο επίπεδο. Δεν υπάρχει μετατόπιση κατά μήκος του άξονα y, δηλαδή, σύμφωνα με την δομή του ρομπότ δεν είναι δυνατή η στιγμιαία κίνηση κατά τον y άξονα.

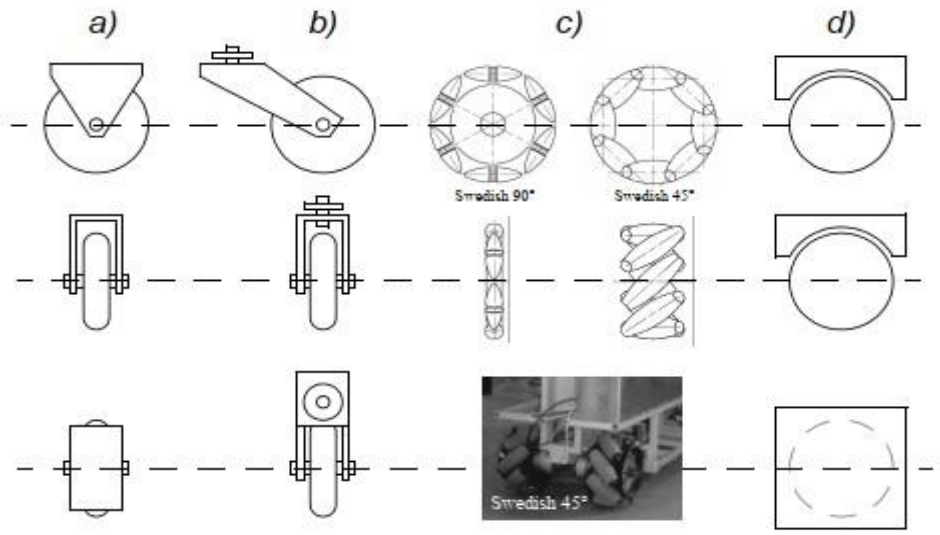
2.2 Τύποι τροχών των αυτοκινούμενων ρομπότ

Υπάρχουν τέσσερεις βασικοί τύποι τροχών, κάθε ένας διαφέρει μεταξύ τους ως προς τις κινηματικές ενός αυτοκινούμενου ρομπότ αυτοί διακρίνονται στην εικόνα 3 [1], [2].

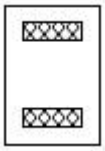
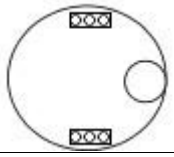
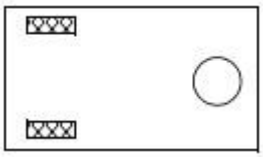
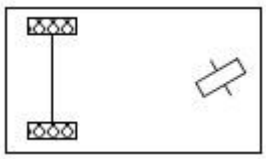
Ο πρώτος είναι ο τυπικός τροχός (σταθερού ή περιστρεφόμενου άξονα) (a), ο δεύτερος είναι ο προσανατολιζόμενος τροχός (castor wheel) (b). Και οι δύο έχουν δύο βαθμούς ελευθερίας περιστροφή γύρω από τον άξονα του τροχού και ελεύθερη περιστροφή από το σημείο ένωσης, η διαφορά μεταξύ τους είναι ότι στον πρώτο τροχό έχουμε τον κατακόρυφο άξονα να ακουμπάει στο έδαφος κατευθείαν, επομένως δεν απαιτεί περισσότερη δύναμη στήριξης, ενώ στον δεύτερο οι άξονες δεν τέμνονται, δηλαδή υπάρχει ένα offset μεταξύ των αξόνων, που συνεπώς απαιτεί περισσότερη δύναμη.

Ο τρίτος τροχός είναι ο Swedish ή omnidirectional (κίνηση προς κάθε κατεύθυνση) (c), και ο τέταρτος τροχός είναι ο spherical (d). Και οι δύο τελευταίοι έχουν από τρεις βαθμούς ελευθερίας, ο τρίτος συμπεριφέρεται ως κανονικός τροχός, αλλά έχει μικρή αντίσταση προς τις

άλλες κατευθύνσεις, ο συγκεκριμένος υπάρχει σε δύο τύπους ο Swedish 45 και ο Swedish 90. Και ο τέταρτος τροχός είναι ο spherical, καθαυτού omnidirectional.



Εικόνα 3: Τέσσερεις τύποι τροχών:
(a) Τυπικός τροχός, (b) Castor Wheel, (c) Swedish Wheel και (d) Spherical wheel
[Βιβλίο: Introduction to Autonomous Mobile Robots Second Edition, The MIT Press]

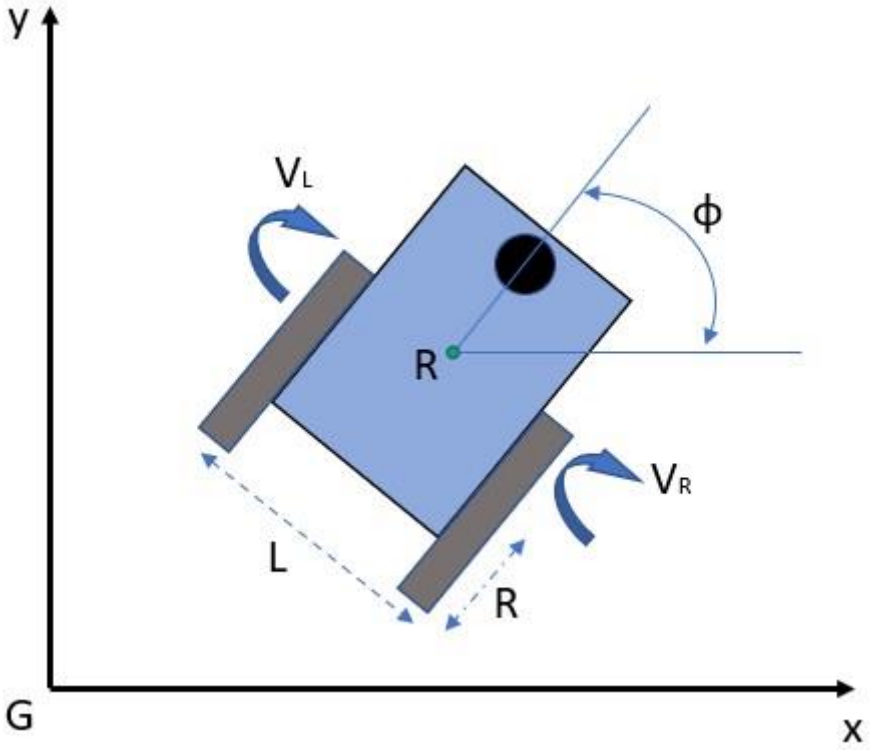
Πλήθος τροχών	Διάταξη τροχών	Περιγραφή
2		Ένας κατευθυντήριος τροχός εμπρός και ένας σταθερός πίσω
		Δύο τροχοί διαφορικής οδήγησης με το κέντρο μάζας κάτω από τον άξονα
3		Δυο κεντραρισμένοι τροχοί διαφορικής οδήγησης και με έναν τρίτο στήριξης
		Δύο ανεξάρτητοι τροχοί πίσω και ένας omnidirectional χωρίς μοτέρ εμπρός
		Δυο συνδεδεμένοι τροχοί πίσω και ένας ελεύθερης κατεύθυνσης εμπρός

Πίνακας 1: Διατάξεις τροχών για τροχοφόρα οχήματα
[Βιβλίο: Introduction to Autonomous Mobile Robots Second Edition, The MIT Press]

2.3 Κινηματική του ρομπότ διαφορικής οδήγησης

Για το σύστημα διαφορικής κίνησης, θα πρέπει να γνωρίζουμε δύο παραμέτρους: την απόσταση μεταξύ των τροχών του ρομπότ L και την ακτίνα των τροχών R, οι οποίες μπορούν να μετρηθούν πολύ εύκολα στο πραγματικό μας ρομπότ με την χρήση ενός μέτρου. Η κατάσταση του ρομπότ αναμένεται να είναι η θέση του x, y και ο προσανατολισμός του ϕ, οπότε αυτές αναμένεται να είναι οι έξοδοι του συστήματος μας [4].

Όπως βλέπουμε και στην οπτική αναπαράσταση εικόνα 4, του ρομπότ διαφορικής οδήγησης θα έχουμε δύο εισόδους οι οποίες είναι: ο ρυθμός περιστροφής του δεξιού και αριστερού τροχού V_r, V_l.



Εικόνα 4: Οπτική αναπαράσταση του ρομπότ διαφορικής οδήγησης

Τώρα λοιπόν, για να έχουμε το κινηματικό μοντέλο του συστήματός μας, θα χρειαστούμε ένα σύνολο εξισώσεων που θα συνδέουν ξανά τις εισόδους του συστήματός μας με τις εξόδους.

Για το σύστημα μας αυτές αναμένεται να είναι οι εξισώσεις:

$$\dot{x} = \frac{R}{2}(V_r + V_l)cos\phi$$

$$\dot{y} = \frac{R}{2}(V_r + V_l)sin\phi$$

$$\dot{\phi} = \frac{R}{L}(V_r - V_l)$$

Αυτές οι εξισώσεις αναμένεται να είναι το κινηματικό μοντέλο του ρομπότ διαφορικής οδήγησης.

Οι κινηματικές εξισώσεις χωρίζονται στην ευθεία και στην αντίστροφη. Στην ευθεία κινηματική ψάχνουμε την γραμμική και γωνιακή ταχύτητα του ρομπότ, δοσμένου ότι έχουμε τις ταχύτητες περιστροφής των δύο τροχών. Οπότε με βάση ότι γνωρίζουμε την απόσταση μεταξύ των τροχών και την ακτίνα κάθε τροχού μπορούμε εύκολα να τις βρούμε.

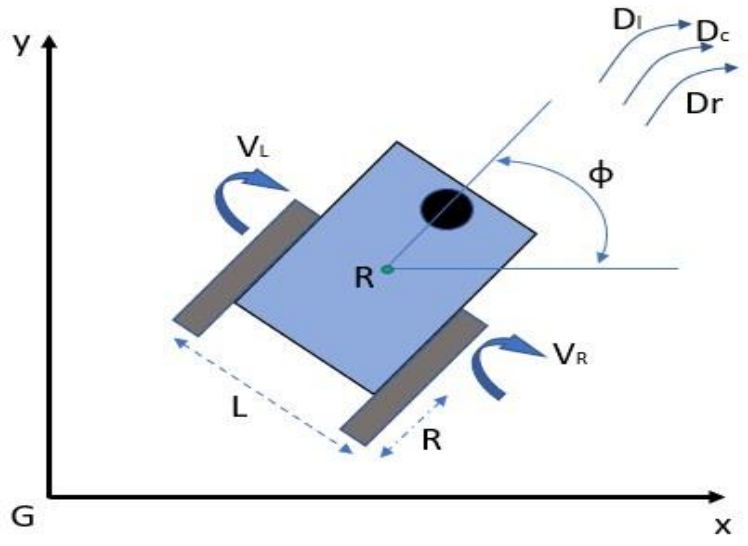
$$v = \frac{R}{2}(V_r + V_l)$$
$$\omega = \frac{R}{L}(V_r - V_l)$$

Ακολούθως για την εύρεση της αντίστροφης κινηματικής δοσμένου ότι έχουμε την επιθυμητή γραμμική και γωνιακή ταχύτητα, θέλουμε να υπολογίσουμε τις ταχύτητες περιστροφής των τροχών.

$$V_r = \frac{2v + \omega L}{2R}$$
$$V_l = \frac{2v - \omega L}{2R}$$

2.4 Έλεγχος κίνησης μέσω της οδομετρίας

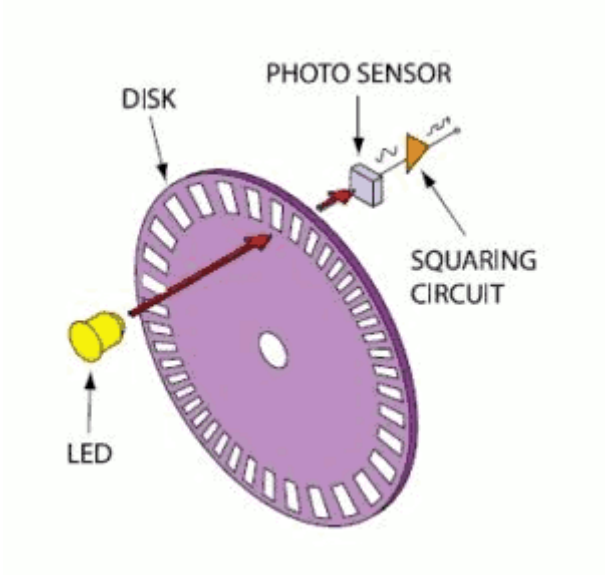
Η οδομετρία χρησιμοποιείται για να λάβουμε την πληροφορία της κατάστασης του ρομπότ δηλαδή το που βρίσκεται. Για να επιτευχθεί αυτό χρησιμοποιούμε αισθητήρες και ο πιο κοινός τρόπος είναι με την χρήση κωδικοποιητών τροχών (wheel encoders). Οι κωδικοποιητές τροχών χρησιμοποιούνται για να μετρούν πόσες φορές έχει περιστραφεί ο κινητήρας. Αυτό μπορεί στη συνέχεια να χρησιμοποιηθεί για να υπολογιστεί η απόσταση που έχει διανύσει το ρομπότ ή πόσο έχει γυρίσει. Ο τρόπος με τον οποίο οι κωδικοποιητές τροχών μετρούν πόσες φορές έχει περιστραφεί ένας κινητήρας είναι με τα τικ (ticks). Βασικά, μετρούν τον αριθμό των ticks που κάνει ένας τροχός ανά κάθε περιστροφή. Συνοψίζοντας, οι κωδικοποιητές τροχών παρέχουν δεδομένα σχετικά με την απόσταση που διανύει κάθε τροχός [5].



Εικόνα 5: Οδομετρία ενός ρομπότ διαφορικής οδήγησης

Για να πραγματοποιηθεί κάτι τέτοιο στο ρομπότ διαφορικής οδήγησης θα χρειαστεί να τοποθετηθεί από ένα wheel encoder σε κάθε τροχό. Ας υποθέσουμε λοιπόν ότι το ρομπότ ακολουθεί μια διαδρομή σε σχήμα τόξου που σημαίνει ότι η γραμμική v και γωνιακή ω ταχύτητα είναι σταθερές. Στην πραγματικότητα, εάν αναλύσουμε το σύστημα σε πολύ μικρά χρονικά διαστήματα, μπορούμε να πούμε ότι αυτή η υπόθεση είναι σωστή.

Στην εικόνα 6, βλέπουμε πως είναι εσωτερικά ένας οπτικός encoder, αποτελείται από τον πομπό και τον δέκτη όπου τους χωρίζει ο δίσκος με τα ticks (δηλαδή μικρά κενά για να περνάει η δέσμη).



Εικόνα 6: Το εσωτερικό ενός encoder
[https://eltra-encoder.eu/news/quadrature-encoder]

Στην εικόνα 5, βλέπουμε την οπτική αναπαράσταση του ρομπότ, όπου Dl θα είναι η απόσταση περιστροφής που διένυσε ο αριστερός τροχός και Dr θα είναι η απόσταση περιστροφής που διένυσε ο δεξιός τροχός. Ωστόσο μας ενδιαφέρει να γνωρίζουμε την κατάσταση του ρομπότ x, y, ϕ. Και αυτή η κατάσταση αναφέρεται στο κέντρο του ρομπότ και όχι στους τροχούς του. Επομένως, η απόσταση που μας ενδιαφέρει θα είναι το Dc όπου θα είναι η απόσταση που έχει γυρίσει το κέντρο του ρομπότ. Το Dc δίνεται από τον ακόλουθο τύπο:

$$D_c = \frac{D_l + D_r}{2}$$

Με αυτήν την εξίσωση μπορούμε ήδη να δημιουργήσουμε ένα σύνολο εξισώσεων για να υπολογίσουμε τη μελλοντική κατάσταση του ρομπότ μας:

$$\acute{x} = x + D_c cos\phi$$
$$\acute{y} = y + D_c sin\phi$$
$$\acute{\phi} = \phi + \frac{D_r - D_l}{L}$$

όπου με τον τόνο (') συμβολίζουμε την εκτίμηση των μεταβλητών δηλαδή την τιμή της μετά από ένα χρονικό διάστημα.

Για να υπολογίσουμε το D_l και D_r, πρέπει να υποθέσουμε τον αριθμό N των ticks ανά περιστροφή (οι περισσότεροι encoder μας δίνουν το αριθμό των ticks εξ αρχής). Για να υπολογίσουμε τα ticks κατά το χρονικό διάστημα που παρακολουθούμε το σύστημά μας δηλαδή τα (Δ_{ticks}), θα είναι η διαφορά μεταξύ των συνολικών ticks στο τέλος του χρονικού διαστήματος (ticks') και των ticks στην αρχή του χρονικού διαστήματος (ticks). Δηλαδή δίνεται από την παρακάτω σχέση:

$$\Delta_{ticks} = ticks' - ticks$$

Με βάση αυτό, μπορούμε στη συνέχεια να υπολογίσουμε την απόσταση που έχει γυρίσει ένας τροχός με την ακόλουθη εξίσωση:

$$D = 2\pi R \frac{\Delta_{ticks}}{N}$$

επομένως μπορούμε να υπολογίσουμε και τα D_l,D_r.

$$D_l = 2\pi R \frac{\Delta_{ticks_l}}{N}$$
$$D_r = 2\pi R \frac{\Delta_{ticks_r}}{N}$$

Με βάση τα D_l και D_r μπορούμε τώρα να υπολογίσουμε την γραμμική και γωνιακή ταχύτητα, όπου δίνονται από τους παρακάτω τύπους:

$$v = \frac{D_c}{\Delta_t}$$
$$\omega = \frac{\frac{D_r - D_l}{L}}{\Delta_t}$$

Τέλος, χρησιμοποιώντας αυτές τις εξισώσεις ταχυτήτων, μπορούμε να πάρουμε το ακόλουθο σύνολο εξισώσεων για να υπολογίσουμε τη μελλοντική κατάσταση του ρομπότ μας:

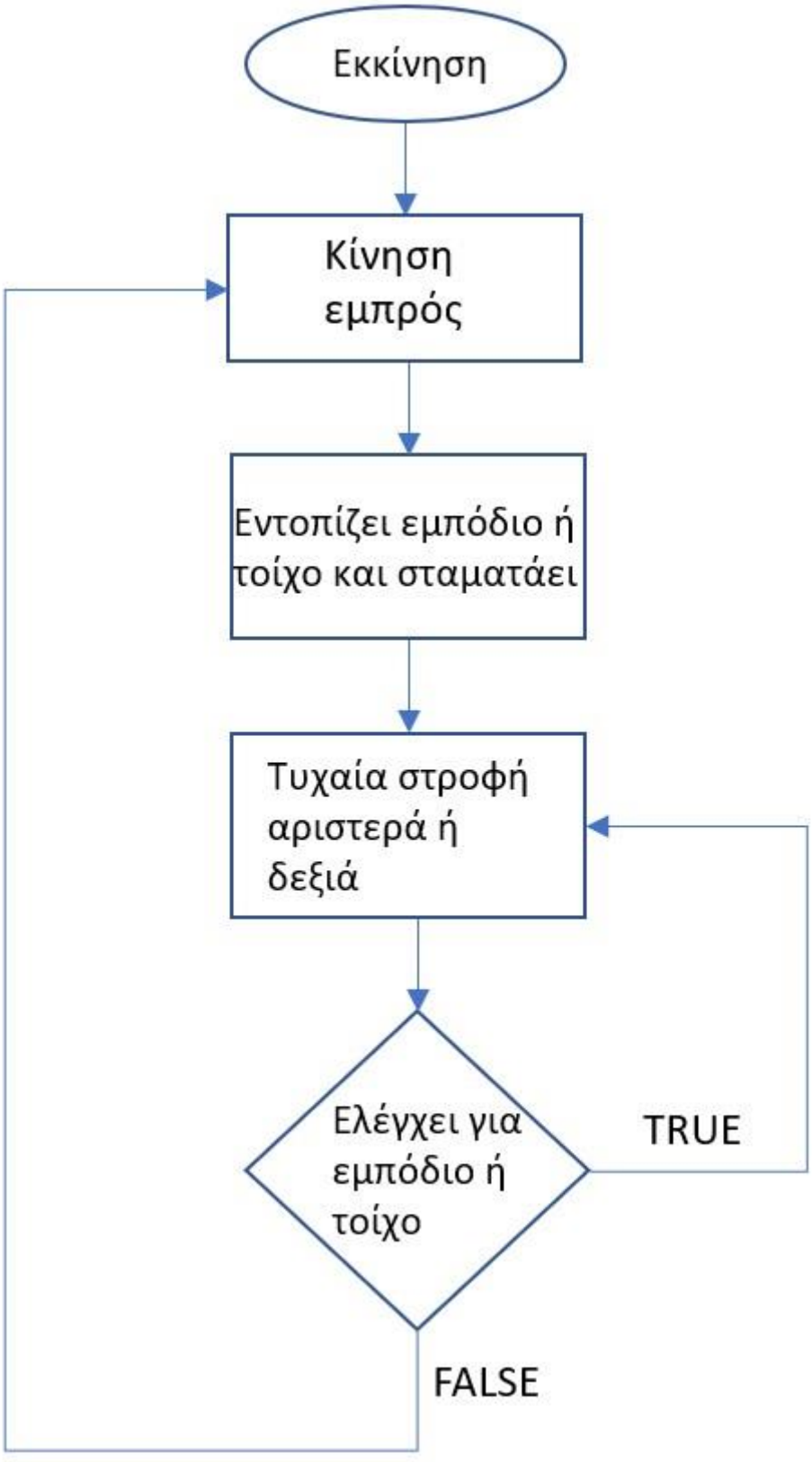
$$\acute{x} = x + vcos\phi$$
$$\acute{y} = y + vsin\phi$$
$$\acute{\phi} = \phi + \omega$$

2.5 Αποφυγή εμποδίων μέσω random walk

Ένας από τους τρόπους αποφυγής εμποδίων είναι με τον αλγόριθμο random walk (τυχαία κίνηση) στο χώρο. Η κύρια πτυχή αυτού του αλγόριθμου είναι ότι δεν απαιτεί χαρτογράφηση του χώρου, δηλαδή δεν χρειάζεται να έχει αποθηκευμένο κάποιο pattern διαδρομής. Βέβαια το μειονέκτημα αυτής της μεθόδου είναι ότι θα περάσει πολλές φορές από τα ίδια σημεία. Ο εντοπισμός εμποδίων πραγματοποιείται από πέντε υπέρυθρους αισθητήρες.

Το ρομπότ ξεκινάει να κινείται ευθεία μέχρι να εντοπίσει κάποιο εμπόδιο ή τοίχο, μόλις εντοπίσει ένα από τα δύο σταματάει και με τυχαία επιλογή στρίβει αριστερά ή δεξιά και μετά

συνεχίζει πάλι να κινείται μέχρι να βρει το επόμενο εμπόδιο ή τοίχο. Στην εικόνα 7 βλέπουμε το διάγραμμα ροής.



Εικόνα 7: Διάγραμμα ροής random walk

2.6 Άλλες μεθοδολογίες αποφυγής εμποδίων

Μια ακόμα μεθοδολογία αποφυγής εμποδίων είναι ο αλγόριθμος snake, αυτός ο αλγόριθμος ονομάζεται έτσι επειδή βασίζεται στην κίνηση ενός φιδιού [6]. Με βάση αυτόν τον αλγόριθμο μπορούν να προκύψουν πολλά σφάλματα λόγω του ότι συνεχώς θα σταματάει και θα εκκινεί ξανά, λόγω αυτής την ιδιομορφίας προτιμάτε σε μικρούς χώρους και χωρίς πολλά εμπόδια ενδιάμεσα στο χώρο.

Η λειτουργία του δεν είναι δύσκολη, δηλαδή εκκινεί μέχρι να βρει εμπόδιο τότε θα στρίψει αριστερά ή δεξιά 90 μοίρες αναλόγως από ποια πλευρά του είναι το εμπόδιο. Θα κινηθεί λίγο και θα στρίψει από την πλευρά που έστριψε προηγουμένως πάλι 90 μοίρες για να ακολουθήσει παράλληλα την πορεία προς την αντίθετη κατεύθυνση μέχρι να βρει εμπόδιο. Αφού βρει το εμπόδιο θα κινηθεί από την αντίθετη φορά που έστριψε τις δύο προηγούμενες φορές πάλι 90 μοίρες, θα κινηθεί πάλι για λίγα δευτερόλεπτα και θα στρίψει 90 μοίρες με την ιδιά φορά που έστριψε προηγουμένως. Αν φτάσει σε σημείο που δεν έχει επιλογή να στρίψει είτε αριστερά ή δεξιά τότε θα κάνει στροφή 180 μοιρών για να ακολουθήσει την ίδια διαδρομή με αντίθετη φορά. Συνεπώς επαναλαμβάνεται η ίδια διαδικασία εκ νέου από την αρχη.

2.7 Θεωρητικό μέρος του ROS και Python

Ο προγραμματισμός στο ROS βασίζεται πάνω σε δύο γλώσσες προγραμματισμού (Python και C++) που συνεργάζονται άψογα με τις εντολές που έχουν αναπτυχθεί για το περιβάλλον του ROS. Στην παρούσα διπλωματική εργασία έχουμε επιλέξει την Python για γλώσσα προγραμματισμού. Περισσότερες λεπτομέρειες για το ROS θα δούμε στο 4ο κεφάλαιο

2.7.1 Εισαγωγή στην Python

Η Python είναι η πιο δημοφιλής γλώσσα προγραμματισμού για τα ρομπότ, η οποία κερδίζει έδαφος κάθε χρόνο στον χώρο της ρομποτικής. Ειδικά στην έρευνα των ρομπότ η Python είναι η νούμερο ένα προτεινόμενη γλώσσα. Πολλές από τις βιβλιοθήκες τεχνητής νοημοσύνης που χρησιμοποιήθηκαν για την ρομποτική γράφτηκαν σε Python, όπως αλγόριθμοι reinforcement learning της OpenAI, η βιβλιοθήκη OpenCV για επεξεργασία εικόνας και πολλές άλλες.

- **Μεταβλητές (variables)**: Μια μεταβλητή μπορεί να είναι σαν ένα δοχείο το οποίο αποθηκεύει κάποια δεδομένα, αυτά μπορεί να είναι ένας αριθμός, ένα txt αρχείο ή ακόμα πιο πολύπλοκοι τύποι δεδομένων. Ενώ το πρόγραμμα μας εκτελείται, οι μεταβλητές μπορούν να είναι προσβάσιμες ή ακόμα να αλλάζουν, αυτό σημαίνει ότι μια νέα τιμή θα ανατεθεί στην μεταβλητή.

- **Τύποι δεδομένων (data types)**: Κάθε τιμή στην python έχει έναν τύπο δεδομένων. Βασικά, ο τύπος δεδομένων από μια μεταβλητή υποδεικνύει τι περιέχει η μεταβλητή. Αυτοί οι τύποι είναι είτε αριθμοί είτε συμβολοσειρά είτε λίστες.

- **Πρόταση υπό συνθήκες (conditional statements)**: Στον προγραμματισμό, οι αποφάσεις γίνονται χρησιμοποιώντας conditional statements κυρίως σε μορφή “if statement”. Στις περισσότερες περιπτώσεις, η απόφαση θα εξαρτηθεί από την τιμή των μεταβλητών ή των αριθμητικών εκφράσεων. Αυτές οι εκφράσεις αξιολογούνται σε true ή false.

- **Βρόχοι (loops)**: Οι δύο κύριοι βρόχοι η while και η for, όπου χρησιμοποιούνται για επαναλαμβανόμενες διαδικασίες.

- **Μέθοδοι (methods)**: Με την πιο γενική έννοια, μια μέθοδος είναι ένα δομικό στοιχείο σε γλώσσες προγραμματισμού, που χρησιμοποιείται για την ομαδοποίηση ενός συνόλου δηλώσεων (statements) ώστε να μπορούν να χρησιμοποιηθούν περισσότερες από μία φορές σε ένα πρόγραμμα. Ο μόνος τρόπος για να επιτευχθεί αυτό χωρίς μεθόδους θα ήταν η επαναχρησιμοποίηση του κώδικα αντιγράφοντας τον και προσαρμόζοντάς τον σε διαφορετικό πλαίσιο. Η χρήση μεθόδων συνήθως βελτιώνει την κατανόηση και την ποιότητα του προγράμματος. Μειώνει επίσης το κόστος ανάπτυξης και συντήρησης του λογισμικού.

- **Κλάσεις (classes)**: Μια κλάση python είναι, βασικά, μια μήτρα κώδικα για τη δημιουργία ενός αντικειμένου.

2.7.2 Εισαγωγή στο ROS

:

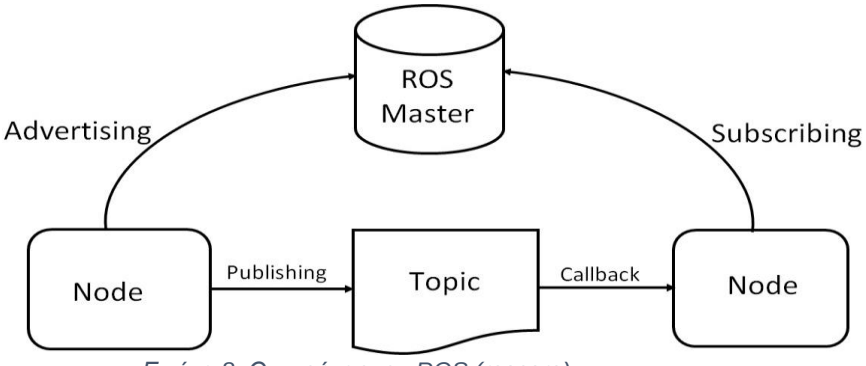
Ξεκινώντας να εργαζόμαστε πάνω στο ROS θα πρέπει να δημιουργήσουμε και το κατάλληλο περιβάλλον εργασίας το λεγόμενο workspace (ws), εκεί είναι ο χώρος όπου θα αναπτύξουμε όλα όσα θα χρειαστούμε.

- **Πακέτα (packages)**: Το ROS χρησιμοποιεί πακέτα για να οργανώσει τα προγράμματά του. Μπορούμε να σκεφτούμε ένα πακέτο ως όλα τα αρχεία που περιέχει ένα συγκεκριμένο πρόγραμμα ROS. όλα τα αρχεία C++, τα αρχεία python, τα αρχεία configuration, τα αρχεία compilation, τα αρχεία launch και τα αρχεία παραμέτρων.

- **Κόμβοι (nodes)**: Ένας κόμβος ROS, σύμφωνα με το ROS wiki, είναι βασικά μια διαδικασία που εκτελεί υπολογισμούς. Είναι ένα εκτελέσιμο πρόγραμμα που τρέχει μέσα στην εφαρμογή μας. Οι κόμβοι συνδυάζονται σε ένα γράφημα και επικοινωνούν μεταξύ τους χρησιμοποιώντας τα ROS topics, service, action και άλλα πολλά. Δύο κόμβοι δεν μπορούν να έχουν το ίδιο όνομα. Εάν θέλουμε να εκτελέσουμε πολλές περιπτώσεις του ίδιου κόμβου, θα πρέπει να προσθέσουμε ένα πρόθεμα ή ένα επίθημα στο όνομα ή να τα δηλώσουμε ως ανώνυμα.

- **Χώρος αποθήκευσης παραμέτρων (parameter server)**: Ο parameter server είναι ένα dictionary, το οποίο χρησιμοποιείται από το ROS για να αποθηκεύει τις παραμέτρους. Αυτές οι παράμετροι μπορούν να χρησιμοποιηθούν από κόμβους κατά το χρόνο εκτέλεσης και χρησιμοποιούνται συνήθως για στατιστικά, όπως οι παράμετροι διαμόρφωσης. Όλοι αυτοί οι παράμετροι αποθηκεύονται σε κάποιο σημείο, το οποίο ονομάζεται parameter server.

- **Ο πυρήνας του ROS (roscore)**: Το **roscore** είναι η κύρια διαδικασία που διαχειρίζεται όλο το σύστημα ROS. Ας δούμε στην εικόνα 8 ένα διάγραμμα απεικόνισης του **roscore**.



Εικόνα 8: Ο πυρήνας του ROS (roscore)
[https://www.designnews.com/gadget-freak/ros-101-intro-robot-operating-system]

- **Δίαυλοι επικοινωνίας (publisher και subscriber topics)**: Τα topics είναι ένας δίαυλος επικοινωνίας στον οποίο ο publisher στέλνει την πληροφορία, ας το σκεφτούμε σαν ένα σωλήνα. Τα nodes χρησιμοποιούν τα topics για να στέλνουν μια πληροφορία σε άλλα nodes μέσω του publisher. Από την στιγμή που στέλνουν αυτή την πληροφορία θα πρέπει να υπάρχει και ένας subscriber, αυτός ο subscriber θα βρίσκεται σε διαφορετικό node και θα λαμβάνει την πληροφορία που έχει σταλθεί στο topic.

- **Μηνύματα (messages)**: Τα μηνύματα αυτά είναι μια απλή δομή δεδομένων, που αποτελούνται από έτοιμα πεδία. Υποστηρίζονται οι συνήθεις τύποι, ακέραιοι (integer), δεκαδικοί (floating point), και boolean και άλλοι πολλοί.

- **Υπηρεσίες (services)**: Μια υπηρεσία ROS είναι στην ουσία ένα σύστημα πελάτη και διακομιστή. Μερικά από τα κύρια χαρακτηριστικά μιας υπηρεσίας ROS είναι τα εξής, είναι σύγχρονο, δηλαδή, ο πελάτης στέλνει ένα αίτημα και περιμένει μέχρι να λάβει απάντηση, μια υπηρεσία ορίζεται από ένα όνομα και ένα ζεύγος μηνυμάτων. Ένα μήνυμα είναι το αίτημα, ένα μήνυμα είναι η απάντηση. Πρέπει να τηρούμε τη μορφή των δεδομένων και από τις δυο πλευρές.

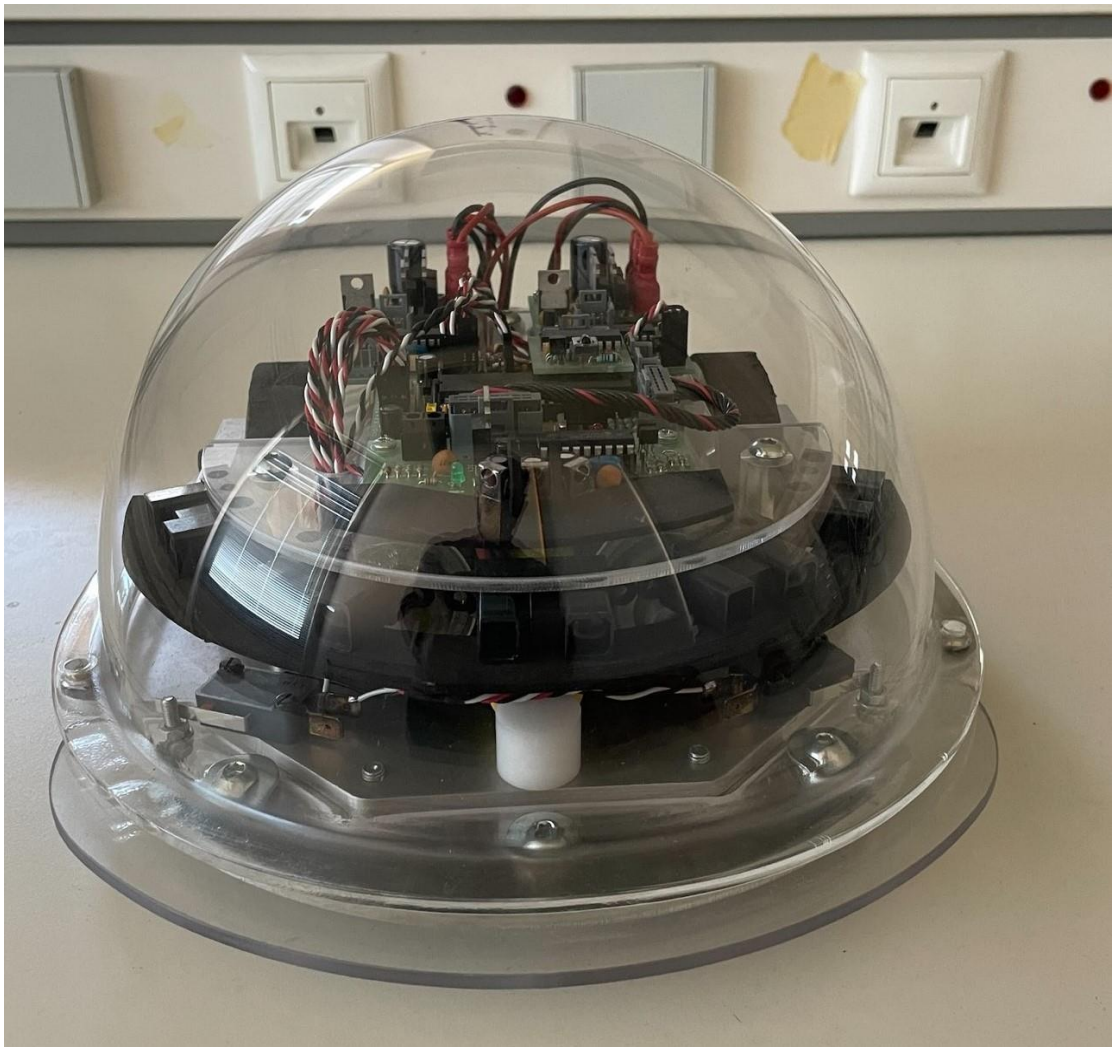
3 Hardware

3.1 Το αρχικό hardware του ρομπότ

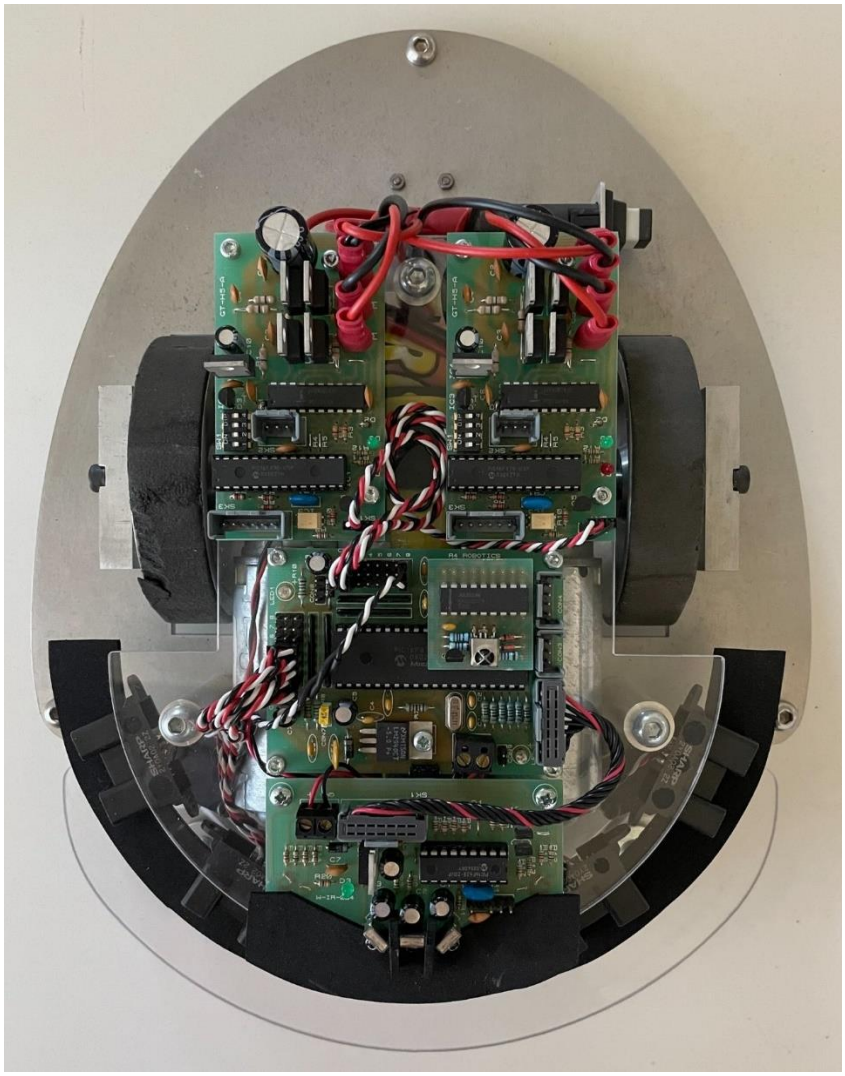
Το ρομπότ ήταν και είναι στην κατοχή του εργαστήριου ρομποτικής της σχολής των Ηλεκτρολόγων Μηχανικών ως ένα παλιό ρομπότ διαφορικής οδήγησης με παλαιότερο hardware και software. Το hardware του ρομπότ αποτελούταν από τα εξής:

- Οι τροχοί μαζί με τα μοτέρ τους
- Δύο πλακέτες που είναι τα motor drives των μοτέρ
- Η κεντρική πλακέτα (μικρό-επεξεργαστής) του ρομπότ
- Μια πλακέτα ελέγχου για τους υπέρυθρους (infrared) αισθητήρες και τα γυροσκόπια
- Πέντε υπέρυθροι αισθητήρες (SHARP 2Y0A02)
- Και μία μπαταρία παλαιού τύπου (Li-Po)

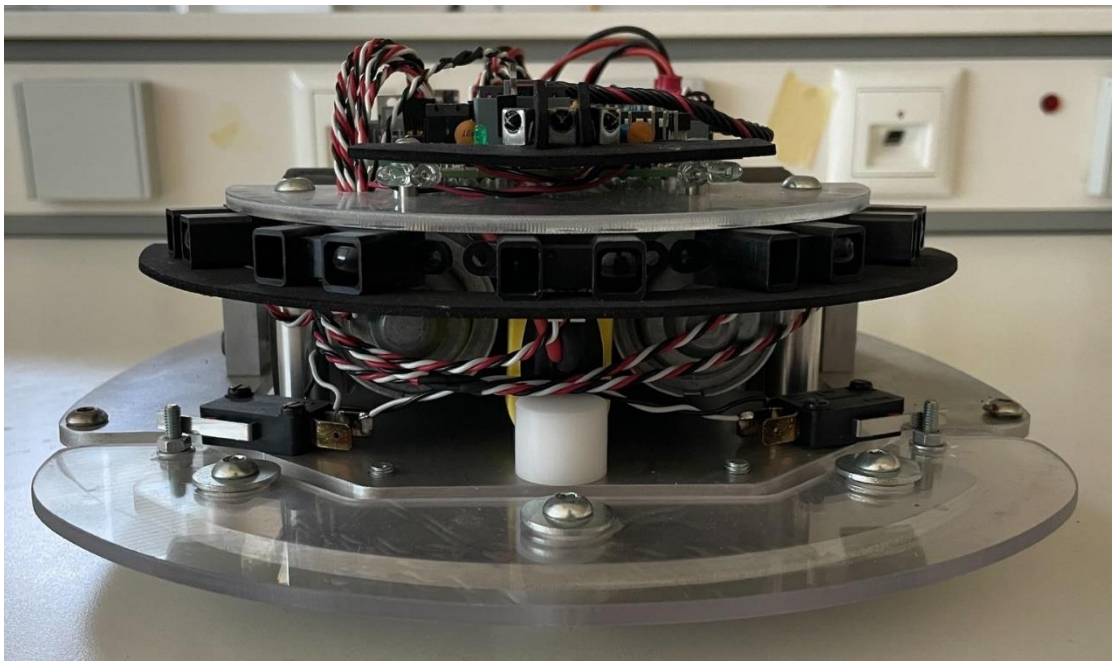
Στις επόμενες τρεις φωτογραφίες βλέπουμε πως ήταν αρχικά ολόκληρο το ρομπότ.



Εικόνα 9: Αρχική μορφή του ρομπότ (part 1)



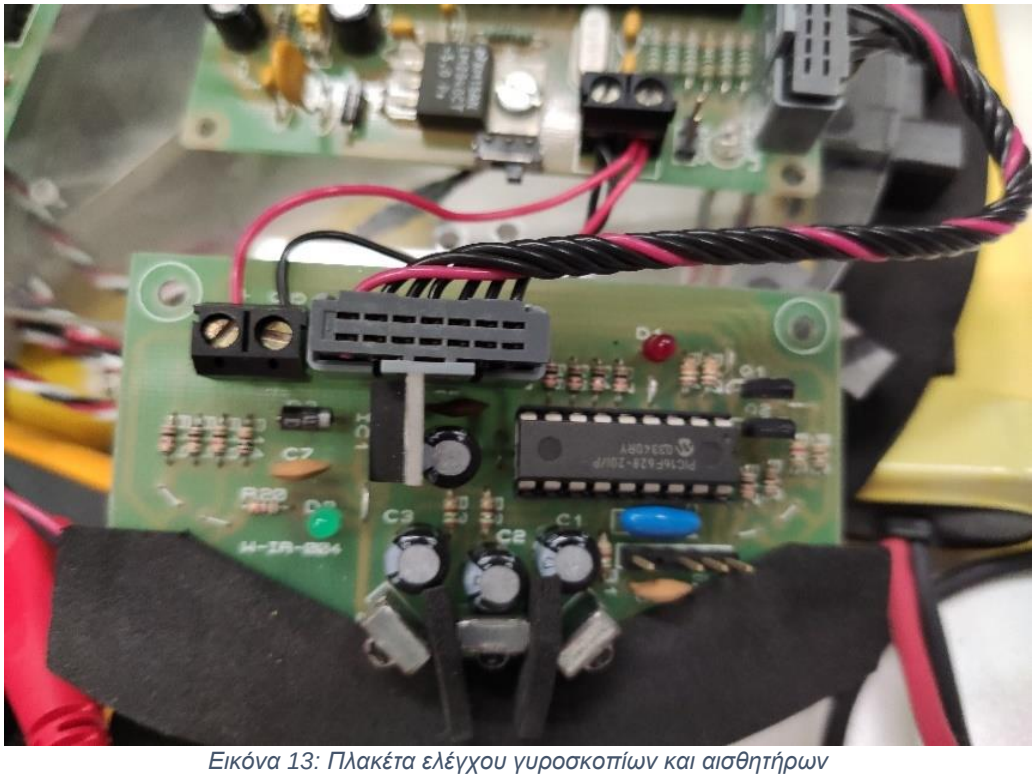
Εικόνα 10: Αρχική μορφή του ρομπότ (part 2)



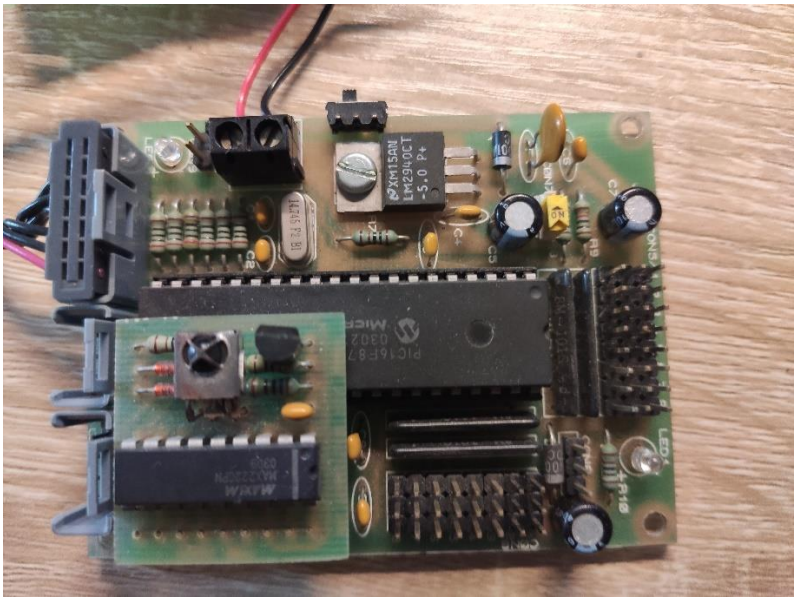
Εικόνα 11: Αρχική μορφή του ρομπότ (part 3)

Παρακάτω βλέπουμε μερικές φωτογραφίες από το αρχικό hardware.

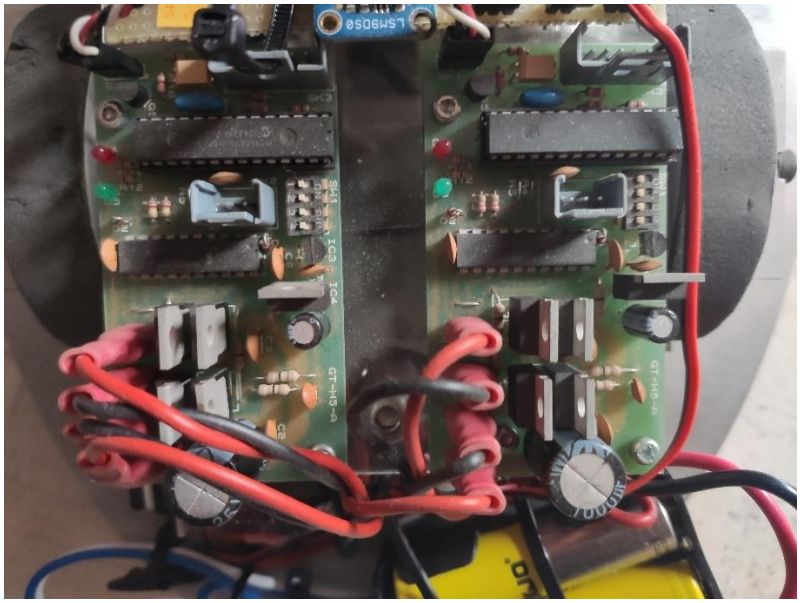
Εικόνα 12: Μοτέρ τροχού 12V



Εικόνα 13: Πλακέτα ελέγχου γυροσκοπίων και αισθητήρων



Εικόνα 14: Η κεντρική πλακέτα του ρομπότ

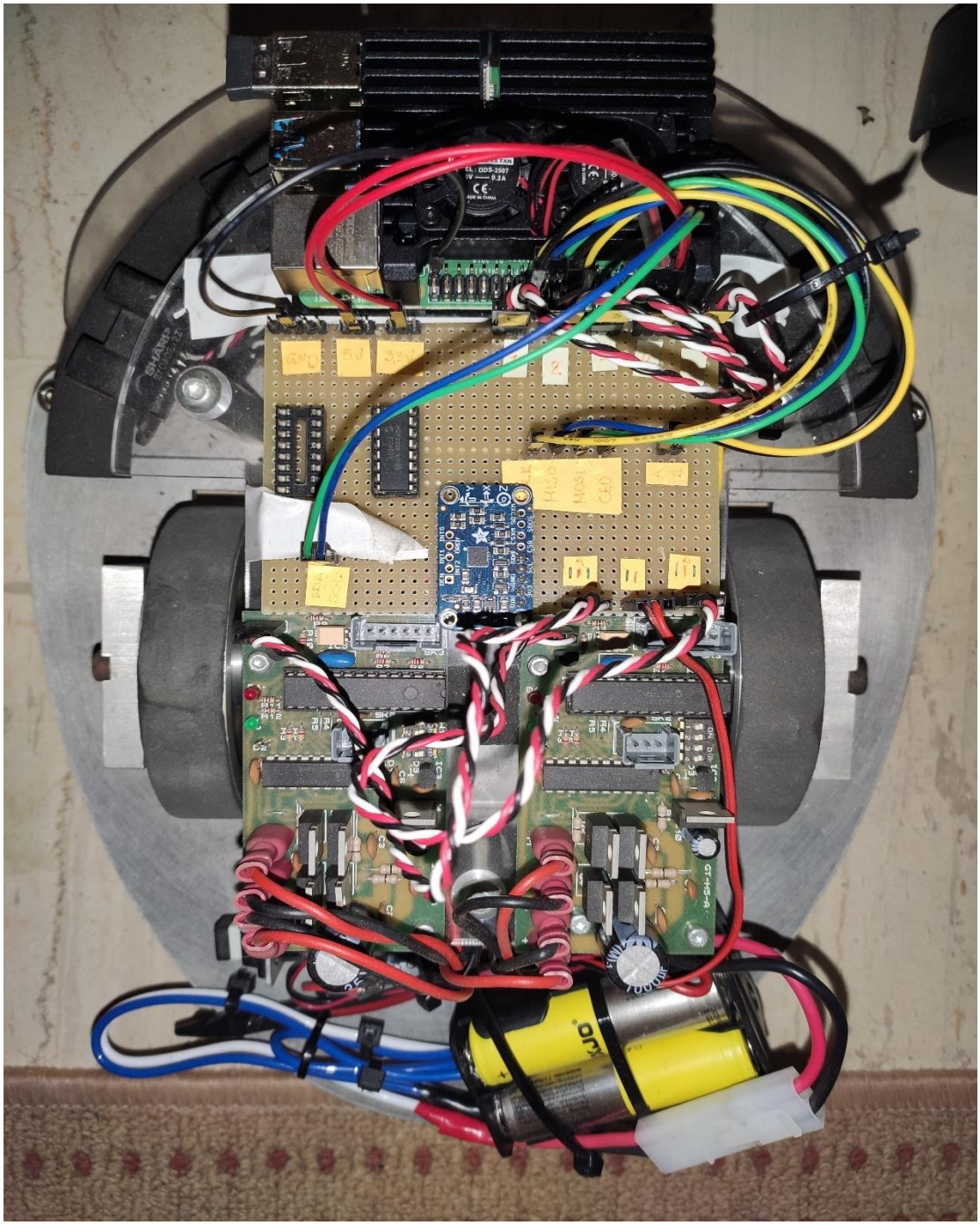


Εικόνα 15: Τα δύο motor drives

Εικόνα 16: Οι υπέρυθροι αισθητήρες SHARP

3.2 Το τελικό hardware του ρομπότ

Έπειτα από κάποιες τροποποιήσεις που δέχτηκε το αρχικό ρομπότ κατέληξε να του αφαιρεθεί ο παλιός επεξεργαστής και η πλακέτα ελέγχου των αισθητήρων και των γυροσκοπίων. Στην θέση αυτών τοποθετήθηκε ένα barebone (συγκεκριμένα ένα Raspberry Pi 4 Model B+), το οποίο είναι ουσιαστικά ο «εγκέφαλος» του, δηλαδή, είναι ένας μικρός υπολογιστής με μικρές δυνατότητες που «τρέχει» λειτουργικό σύστημα. Επιπλέον τοποθετήθηκε ένα voltage regulator (5V), ένα IMU και μαζί με μια handmade πλακέτα οπού έχει συγκεντρωμένα τα σήματα, τις τροφοδοσίες (5V, 3V3, GND), ένα ολοκληρωμένο MCP3008 και κάποια pin headers για να γίνει η διασύνδεση μεταξύ του RPi, των αισθητήρων και των motor drives. Στην ενότητα 3.2.4 θα δούμε τι είναι το IMU που αναφέρθηκε.



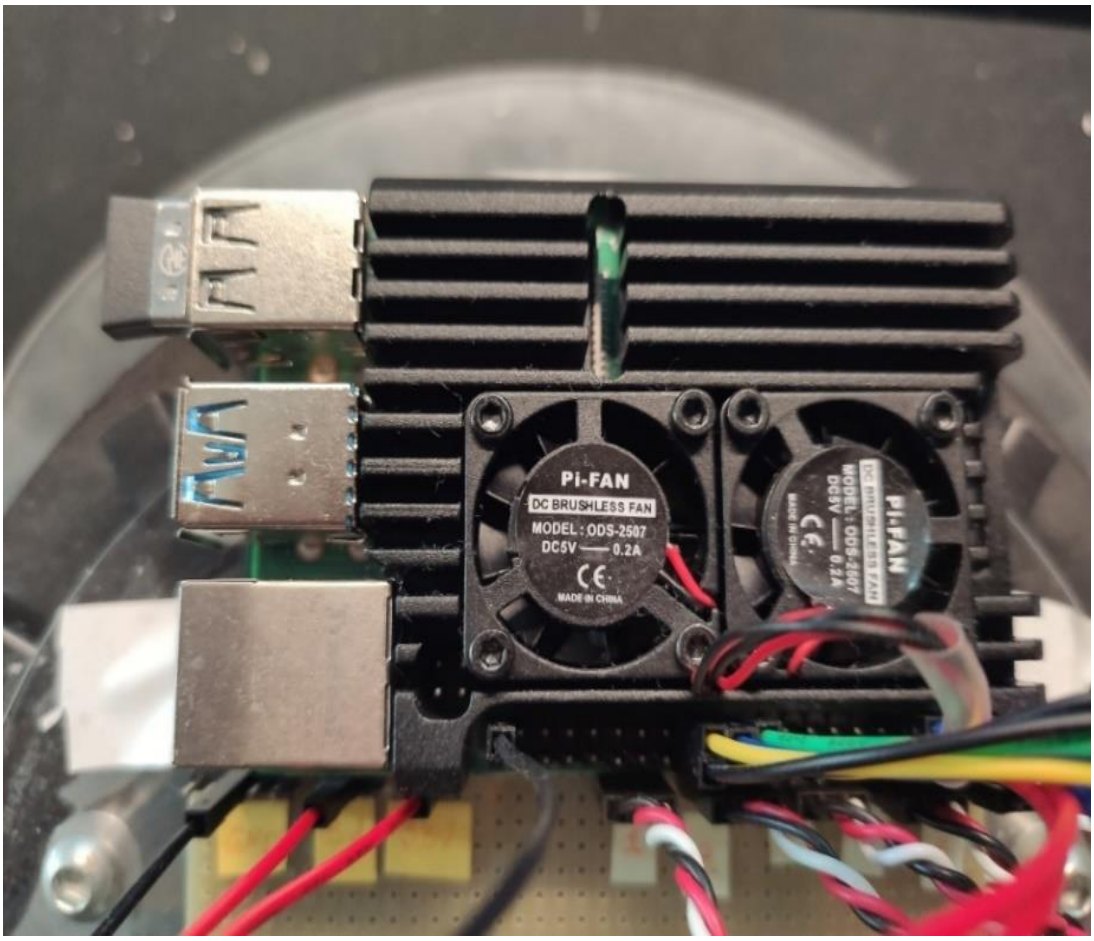
Εικόνα 17: Το τελικό ρομπότ

3.2.1 Το Raspberry Pi 4 Model B+

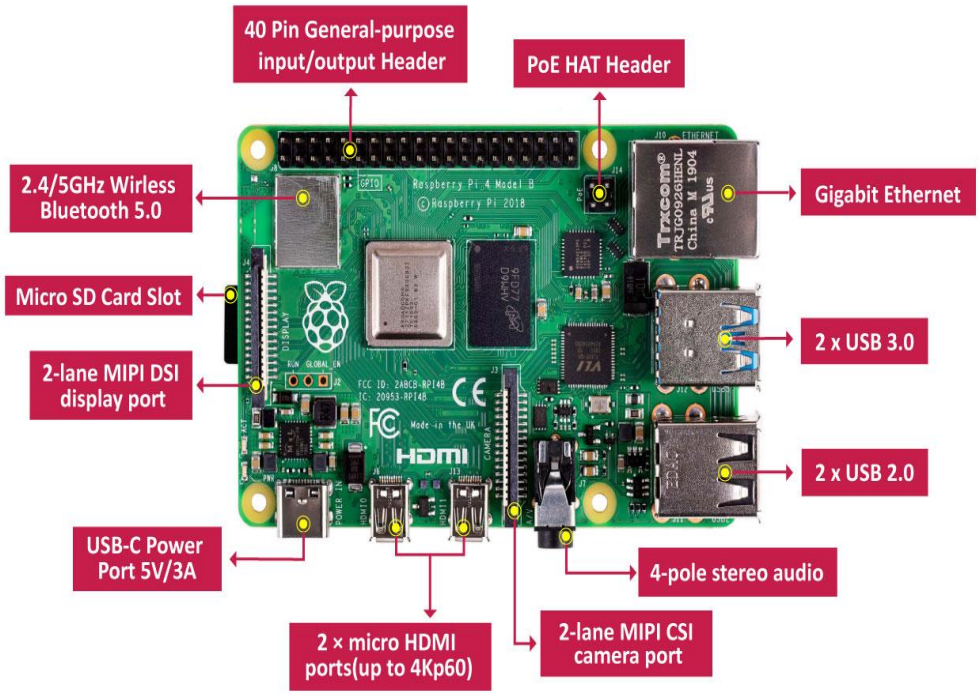
Το RPi είναι ένας open source μικροϋπολογιστής που μπορεί να δεχτεί λειτουργικά συστήματα βασισμένα σε Linux και Windows. Επίσης με την ιδιαιτερότητα που έχει να υποστηρίζει pins για την σύνδεση με μικροηλεκτρονικά το κάνει ιδιαίτερα γνωστό σε developers και hobbyists για την δημιουργία διαφόρων projects κυρίως για έξυπνα σπίτια, drones, αυτόνομα οχήματα, ρομποτικούς βραχίονες και διάφορα άλλα.

Το RPi τροφοδοτείται με 5V για να λειτουργήσει, είτε αυτά μπορούμε να τα παρέχουμε με τον μετασχηματιστή που συνδέεται στο USB-C Port είτε από τα pins στο Input/output header (pins 2 ή 4), το συγκεκριμένο RPi έχει 40 pin γενικού σκοπού (GPIO). Επίσης το RPi έχει ενσωματωμένες δυο θύρες micro HDMI, δυο θύρες USB 2.0, δυο θύρες USB 3.0, μια θύρα για ethernet. Τέλος να αναφέρουμε ότι το RPi επίσης έχει ενσωματωμένο Bluetooth και WIFI, και το σημαντικότερο από όλα είναι η υποδοχή για SD Card όπου εκεί θα γίνει και η εγκατάσταση του λειτουργικού συστήματος.

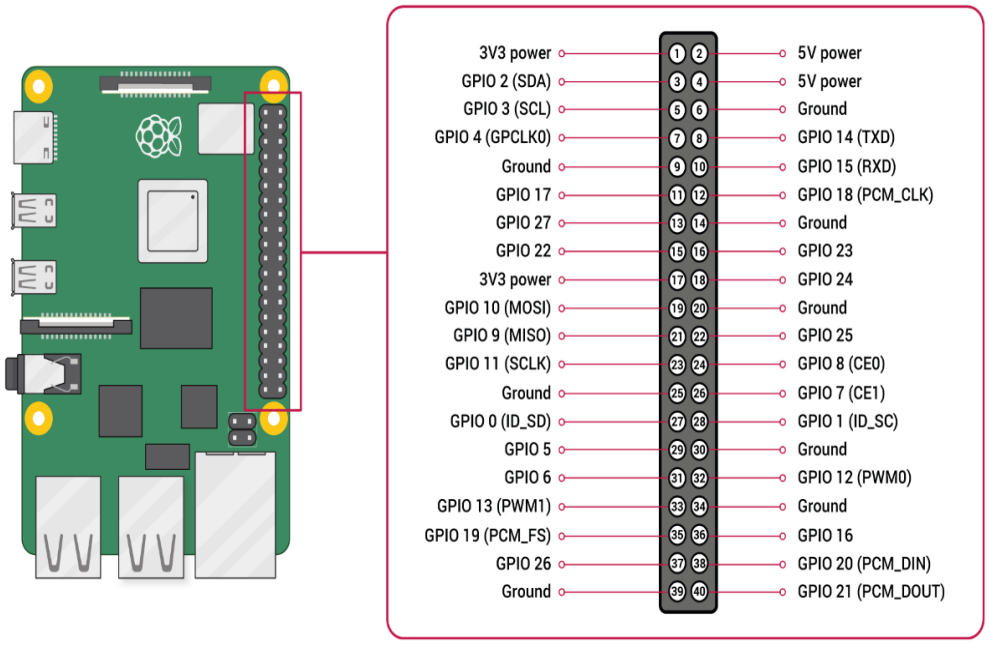
Στις παρακάτω εικόνες βλέπουμε όλα αυτά που αναφέραμε παραπάνω, καθώς και ένα λεπτομερές σχέδιο με τι είναι το κάθε pin του header όπως επίσης και το RPi που τοποθετήθηκε στο ρομπότ.



Εικόνα 18: Το Raspberry Pi 4 (μαζί με armor case και fans)



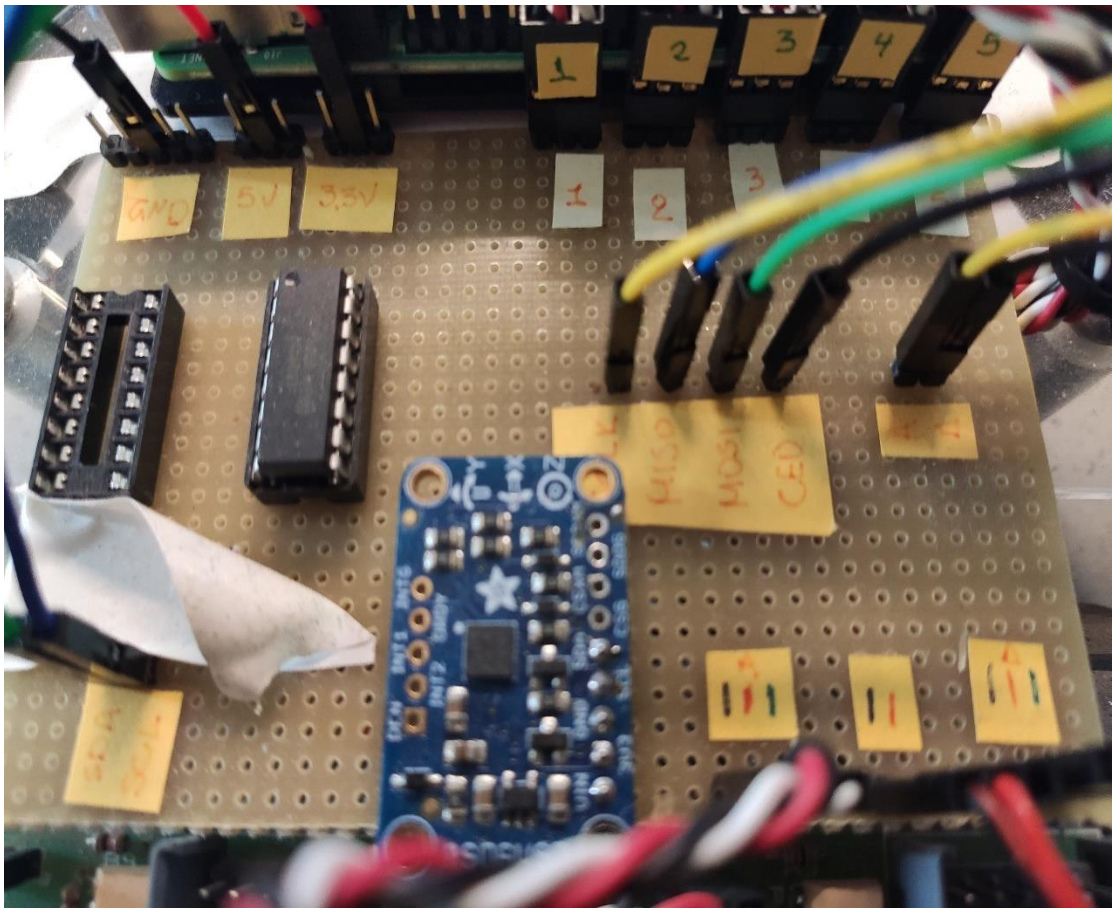
Εικόνα 19: Το hardware του RPi4 [www.google.gr]



Εικόνα 20: Τα GPIOs του RPi [www.google.gr]

3.2.2 Η επιπρόσθετη handmade πλακέτα

Για να γίνει η διασύνδεση μεταξύ του RPi με τους πέντε αισθητήρες, τα δύο motor drives, το IMU και την τροφοδοσία που προέρχεται από τέσσερεις μπαταρίες (σύνολο ~15V), θα χρειαστούμε μια επιπρόσθετη πλακέτα για να συγκεντρώσουμε τα σήματα ώστε να αποφύγουμε να έχουμε πάρα πολλά καλώδια στον αέρα. Την πλακέτα αυτή την βλέπουμε στην επόμενη εικόνα 21.



Εικόνα 21: Η handmade πλακέτα διασύνδεσης

Όπως μπορούμε να δούμε και στην παραπάνω φωτογραφία που φαίνεται η πλακέτα διασύνδεσης, αυτή περιλαμβάνει το IMU που το έχω στηρίξει πάνω της με κόλληση, ένα MCP3008 (ADC Converter) όπου το έχω στηρίξει σε μια βάση 16 pins για DIP, στην επόμενη ενότητα θα δούμε τι είναι το MCP3008. Από εκεί και πέρα έχει γίνει ομαδοποίηση των τροφοδοσιών και διασύνδεση των σημάτων του SPI και του I2C.

Κάτω δεξιά στην πλακέτα βλέπουμε τρία dupont (δύο τριπολικά και ένα διπολικό) με σειρά από αριστερά προς δεξιά, το αριστερό συνδέεται με την πλακέτα (motor drive) του αριστερού τροχού και είναι τριπολικό όπου τα δυο είναι για την τροφοδοσία των 5V της πλακέτας και το τρίτο για τα data, το δεξί συνδέεται αντίστοιχα για τον δεξί τροχό και τέλος το μεσαίο που είναι διπολικό είναι η τροφοδοσία των 5V που έρχεται από τον voltage regulator.

Κάτω αριστερά στην πλακέτα βλέπουμε δυο μονοπολικά dupont με σειρά από αριστερά προς δεξιά, το αριστερό είναι για το σήμα SDA (I2C) που προέρχεται από την πλακέτα IMU και συνδέεται στο GPIO 2 του RPi και το δεξί είναι για το σήμα SCL (I2C) και συνδέεται στο GPIO 3 του RPi.

Στο 4^{ο} κεφάλαιο που είναι για το software, θα δούμε τι είναι τα πρωτόκολλα επικοινωνίας SPI και I2C.

Στην μέση και δεξιά της πλακέτας βλέπουμε έξι μονοπολικά dupont με σειρά από αριστερά προς δεξιά, τα τέσσερα πρώτα είναι για το πρωτόκολλο επικοινωνίας SPI με τα

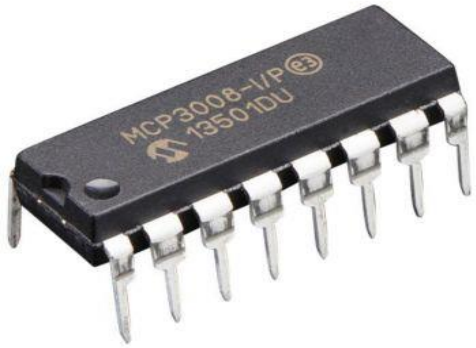
σήματα SCLK, MISO, MOSI, CEO τα οποία προέρχονται από το MCP3008 και συνδέονται στο RPi στα GPIO 11, 9, 10, 8 αντίστοιχα. Και τα δυο τελευταία είναι data από τους δυο τροχούς και τα οποία συνδέονται με τα GPIO 23 και 24 όπου το 23 είναι για τον αριστερό και το 24 για τον δεξί.

Έπειτα πάνω δεξιά βλέπουμε πέντε τριπολικά dupont με μια αρίθμηση από ένα έως πέντε, το κάθε ένα από αυτά είναι για τους πέντε υπέρυθρους αισθητήρες, όπου το κάθε ένα έχει τρία pins, τα δυο είναι για την τροφοδοσία και το τρίτο για τα data.

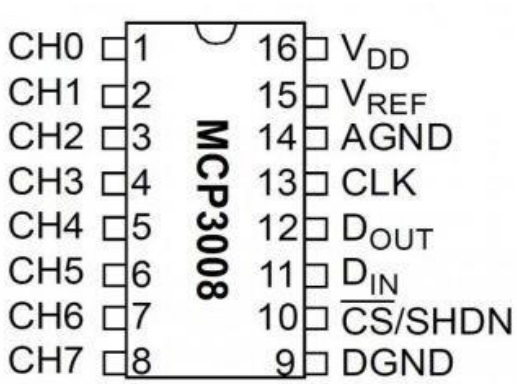
Τέλος πάνω αριστερά είναι οι τροφοδοσίες, από αριστερά προς δεξιά έχουμε το GND που συνδέεται με το GND του RPi, μετά είναι τα 5V που συνδέεται πάλι με το RPi, στην ουσία αυτά τα δύο είναι η τροφοδοσία για το RPi και τέλος έχουμε και τα 3V3 που προέρχονται από το RPi για να τροφοδοτήσουμε το MCP3008. Βέβαια θα μπορούσαμε να μην χρησιμοποιήσουμε τα 3V3 του RPi για να τροφοδοτήσουμε το ολοκληρωμένο και να χρησιμοποιούσαμε αυτά από το IMU, αλλά το IMU υποστηρίζει μέγιστο φορτίο μέχρι 100mA ενώ το RPi που είναι τελευταίας γενιάς μπορεί να δώσει μέχρι 500mA που μπορεί να φανούν χρήσιμα σε μελλοντικές προσθήκες.

3.2.3 Υπέρυθροι (Infrared) αισθητήρες μαζί με το MCP3008 (ADC Converter 10bit 8 channels)

Το MCP3008 είναι ένας 10bit μετατροπέας αναλογικού σήματος σε ψηφιακό οκτώ καναλιών, δηλαδή, μπορεί να δεχτεί οκτώ εισόδους αναλογικού σήματος, και έχει ως πρωτόκολλο επικοινωνίας το SPI το οποίο απαιτεί τέσσερα pins για την επικοινωνία. Η επιλογή του συγκεκριμένου IC (integrated circuit) έγινε γιατί ταιριάζει απόλυτα με ένα RPi, διότι το RPi δεν υποστηρίζει αναλογικά σήματα. Η τροφοδοσία του έχει ένα εύρος από 2.7V έως 5.5V. Στην εικόνα 22 βλέπουμε πως είναι ένα IC MCP3008.

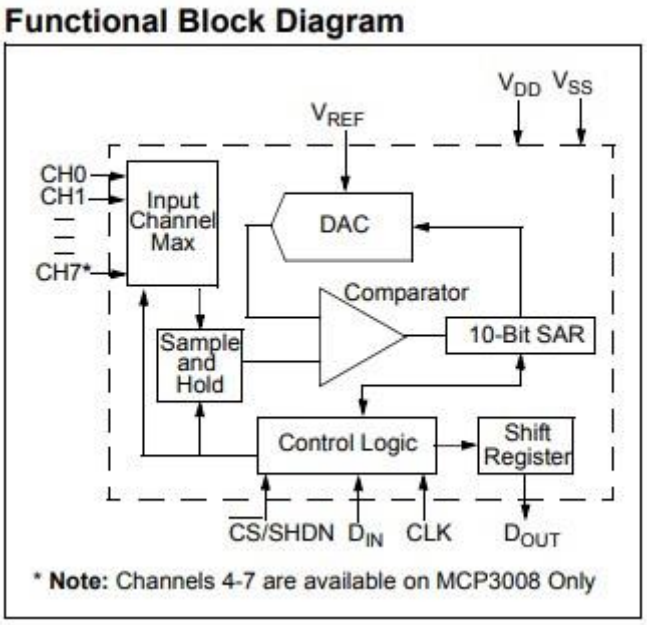


Εικόνα 22: Ένα through - hole IC MCP3008 της Microchip [www.google.gr]



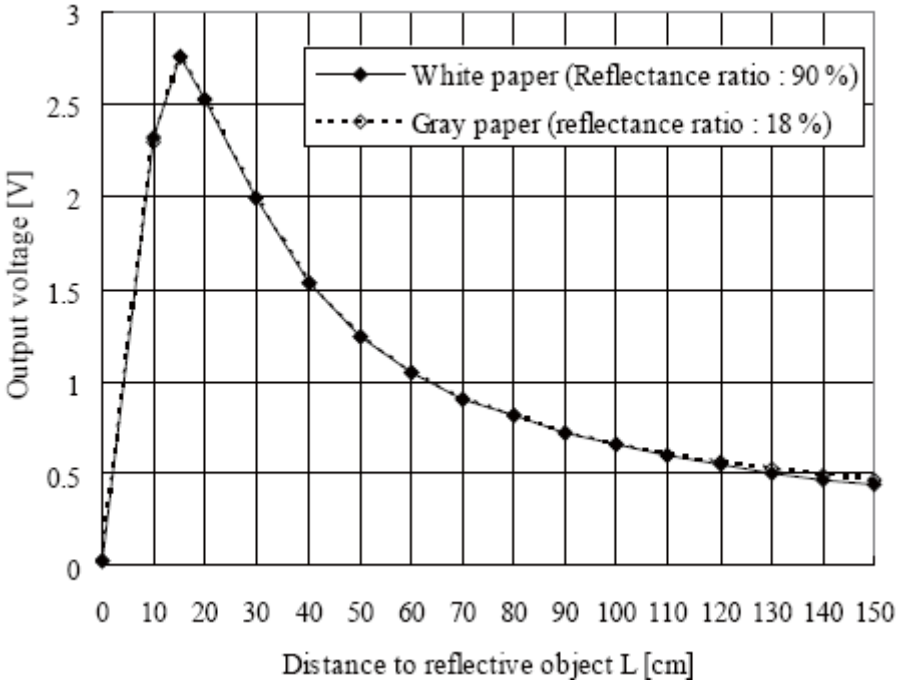
Εικόνα 23: Τα pins του MCP3008 [www.google.gr]

Στην παραπάνω εικόνα 23 βλέπουμε το διάγραμμα των 16 pins του IC. Από το 1 έως 8 είναι τα οκτώ κανάλια αναλογικών εισόδων που υποστηρίζει, το 9 και 14 είναι για το GND, στα 15 και 16 συνδέουμε την τροφοδοσία (3.3V). Τέλος τα υπόλοιπα 10 έως 13 είναι για την επικοινωνία μέσω του πρωτοκόλλου SPI. Στην επόμενη εικόνα 24 βλέπουμε πως είναι εσωτερικά το σχηματικό διάγραμμα μέσω blocks.



Εικόνα 24: Το functional block diagram του MCP3008 [Datasheet από Microchip]

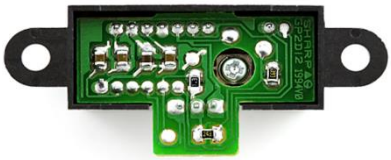
Οι υπέρυθροι αισθητήρες είναι της εταιρίας SHARP και συγκεκριμένα το μοντέλο 2Y0A02. Ο υπέρυθρος αισθητήρας χρειάζεται μια τάση μεταξύ 4.5V και 5.5V ως είσοδο, επομένως τα 5V είναι αρκετά για την λειτουργία του. Στην εικόνα 25 βλέπουμε πόση τάση βγάζει το pin του αισθητήρα βάση το πόσο μακριά είναι ένα αντικείμενο.



Εικόνα 25: Μετρούμενη απόσταση σε volts [Datasheet της Sharp]

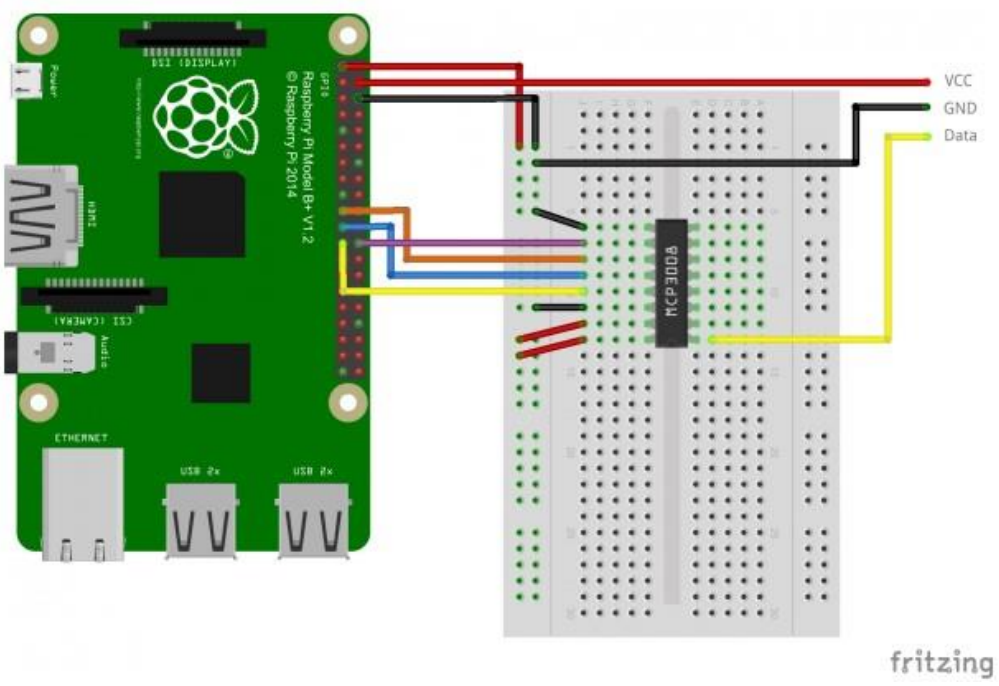
Στις δύο επόμενες εικόνες 26 και 27 βλέπουμε πως είναι ένας τέτοιος αισθητήρας. Αυτός ο αισθητήρας «βγάζει» τρία pins, τα δύο για την τροφοδοσία και το τρίτο για τα data.

Εικόνα 26: Εξωτερική εμφάνιση του Sharp 2Y0A02 [www.google.gr]



Εικόνα 27: Η πλακέτα του Sharp 2Y0A02 [www.google.gr]

Στις δυο επόμενες εικόνες 28 και 29 βλέπουμε την σύνδεση που απαιτείται μεταξύ του RPi, του αισθητήρα και του MCP3008. (Σημείωση ότι το RPi που φαίνεται στην εικόνα δεν είναι το ίδιο με αυτό που χρησιμοποιήσαμε στο project, παρά μόνο ταιριάζουν τα GPIO).



Εικόνα 28: Διασύνδεση για την ορθή λειτουργία του αισθητήρα [www.google.gr]

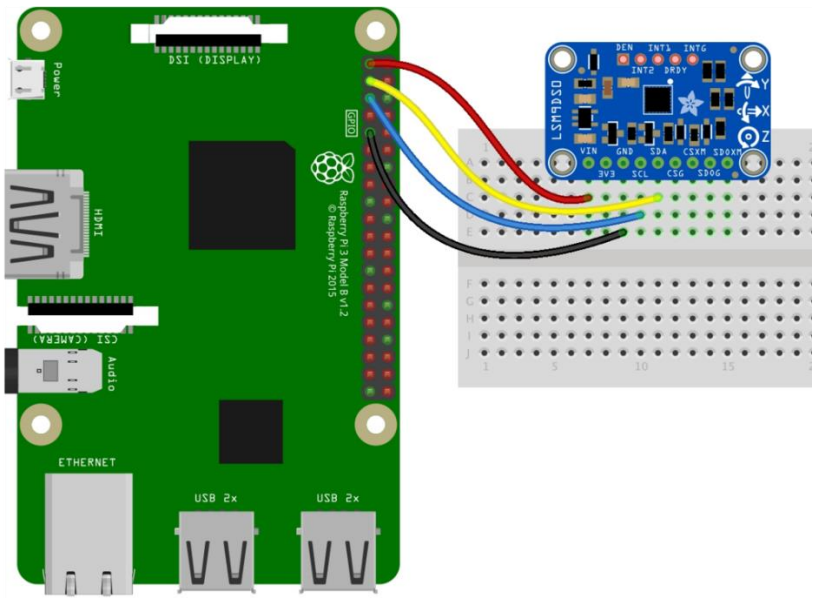
RaspberryPi	MCP3008
Pin 1 (3.3V)	Pin 16 (VDD)
Pin 1 (3.3V)	Pin 15 (VREF)
Pin 6 (GND)	Pin 14 (AGND)
Pin 23 (SCLK)	Pin 13 (CLK)
Pin 21 (MISO)	Pin 12 (DOUT)
Pin 19 (MOSI)	Pin 11 (DIN)
Pin 24 (CE0)	Pin 10 (CS/SHDN)
Pin 6 (GND)	Pin 9 (DGND)

Εικόνα 29: Σύνδεση των GPIOs και του MCP3008 [www.google.gr]

3.2.4 IMU 9 DOF (Inertial Measurement Unit)

Το IMU προέρχεται από τις λέξεις μονάδα μέτρησης αδρανειών, δηλαδή, Inertial Measurement Unit. Αυτό είναι μια ηλεκτρονική συσκευή η οποία μετράει και αναφέρει την δύναμη ενός σώματος, την γωνιακή ταχύτητα και τον προσανατολισμό. Αυτό γίνεται μέσα από τον συνδυασμό επιταχυνσιομέτρου, γυροσκόπιού και μαγνητόμετρου.

Το συγκεκριμένο IMU εννιά βαθμών ελευθερίας, είναι της εταιρίας Adafruit και το μοντέλο είναι το LSM9DS0 και θα χρησιμοποιήσουμε το πρωτόκολλο επικοινωνίας I2C. Η σύνδεση με το RPi είναι πολύ απλή, στην εικόνα 30 βλέπουμε την σύνδεση τους.



Εικόνα 30: Σύνδεση RPi με το IMU [www.google.gr]

- **RPi 3V3 (Pin 1)** στο **VIN** του IMU
- **RPi GND** στο **GND** του IMU
- **RPi SCL (GPIO 3)** στο **SCL** του IMU
- **RPi SDA (GPIO 2)** στο **SDA** του IMU

Σύμφωνα με το datasheet του LSM9DS0 (το οποίο είναι ενσωματωμένο πάνω σε board) δέχεται τάση εισόδου 3.3V, αλλά πάνω στο board υπάρχει επίσης ένα ενσωματωμένο voltage regulator που παρέχει σταθερά 3.3V σε περίπτωση που επιθυμούσαμε να του δώσουμε 5V. Στην επόμενη εικόνα 31 βλέπουμε πως είναι στην πραγματικότητα ένα IMU.

Εικόνα 31: Το πραγματικό IMU [www.google.gr]

4 Software

4.1 Επιπρόσθετες πληροφορίες για τα Ubuntu και το ROS

Στο Raspberry Pi (RPi) έχουμε προ εγκαταστήσει το λειτουργικό σύστημα (Ubuntu 20.04.5 Server) βασισμένο σε Linux. Κατά κύριο λόγο η χρήση των Ubuntu γίνεται από προγραμματιστές, πληροφορικούς και διάφορα άλλα επαγγέλματα σχετικά με την τεχνολογία πληροφοριών (IT). Η πλοήγηση στα Ubuntu γίνεται συνήθως μέσω του τερματικού (terminal) για χρήστες που είναι έμπειροι και αρκετά έμπειροι.

Για να είσαι σε θέση να τρέξεις εφαρμογές ρομποτικής δεν σου αρκεί μόνο το λειτουργικό σύστημα Ubuntu, θα πρέπει να εγκαταστήσεις και το σύστημα διαχείρισης των ρομπότ ROS στα Ubuntu. Είναι μια διαδικασία σχετικά εύκολη καθώς υπάρχουν έτοιμες οδηγίες από την κοινότητα του ROS. Κάθε έκδοση του ROS θέλει και την αντίστοιχη έκδοση των Ubuntu, την παρούσα στιγμή η πιο σταθερή (stable) έκδοση και καλά δοκιμασμένη είναι η έκδοση Noetic του ROS. Γι' αυτό τον λόγο επιλέξαμε και την έκδοση 20.04.5 των Ubuntu, παρότι παλαιότερη δουλεύει άψογα με την έκδοση Noetic.

Για να φανεί χρήσιμο το RPi ώστε να γίνει χρήση του SPI και του I2C θα πρέπει να γίνει εγκατάσταση και του **raspi-config**, ένα εργαλείο που δεν υπάρχει από μόνο του στα Ubuntu παρά μόνο στο λειτουργικό που είναι προορισμένο για το RPi (το Raspberry Pi OS).

Για να κάνουμε χρήση των GPIOs θα πρέπει να κάνουμε και εγκατάσταση μιας βιβλιοθήκης που λέγεται **pigpio**, η οποία είναι διαθέσιμη στο GitHub.

4.2 Λειτουργία των πακέτων του ROS

Το ρομπότ θέλουμε να πραγματοποιεί τυχαία κίνηση στον χώρο και να αποφεύγει τα εμπόδια που εντοπίζουν οι υπέρυθροι αισθητήρες που έχει πάνω του. Όλο αυτό υλοποιείται με την παρουσία πέντε πακέτων στο περιβάλλον εργασίας του ROS, κάθε πακέτο έχει την δική του λειτουργία. Ας δούμε παρακάτω αναλυτικά τα πέντε πακέτα, όπου το κάθε πακέτο περιέχει αρχεία script και launch.

4.2.1 Obstacle sensor

Στο αρχείο python, το node **obstacle_sensor** χρησιμοποιεί την βιβλιοθήκη SpiDev για να διαβάζει από τους πέντε υπέρυθρους αισθητήρες αποφυγής εμποδίων (infrared sensor) και κάνει publish αυτά στο topic /sensor/sonar_X, όπου X παίρνει τιμές L, R, FL, FR και F και το topic αυτό γίνεται subscribe από το node (obstacle_avoiding_random_walker).

Το node συνδέεται σε ένα κανάλι του SPI και διαβάζει από τους αντίστοιχους αισθητήρες και κάνει publish τα data στο αντίστοιχο topic το οποίο έχει ένα μοναδικό όνομα για κάθε αισθητήρα. Στο αρχείο launch, έχω ομαδοποιήσει όλα τα nodes σε ένα όπου κάθε φόρα παίρνει διαφορετικό όνομα.

- sensor/sonar_L
- sensor/sonar_FL
- sensor/sonar_F
- sensor/sonar_FR
- sensor/sonar_R

4.2.2 Obstacle Avoiding Random Walker

Στο αρχείο python obstacle_solver, το node κάνει subscribe σε όλα τα topic αισθητήρων και κρατάει αυτά ενημερωμένα για τον υπολογισμό αποφυγής εμποδίων. Αυτό το node φιλοξένει μια υπηρεσία ROS που ονομάζεται obstacle_free_direction. Το service αυτό ελέγχει τους τρεις αισθητήρες F, L, R για την παρουσία εμποδίων. Το service τότε απαντά με ένα string που περιέχει τις ελεύθερες κατευθύνσεις, επίσης για να σταματήσει η κίνηση του ρομπότ χρησιμοποιούνται ο μπροστινός, ο μπροστινός αριστερός και ο μπροστινός δεξιός.

Στο αρχείο launch obstacle_solver, έχω δημιουργήσει 5 παραμέτρους για την απόσταση που θέλουμε να ελέγξουμε για τον κάθε αισθητήρα ώστε να μπορούν να παραμετροποιηθούν εύκολα.

Στο αρχείο python improved_random_walker, το node καλεί την υπηρεσία obstacle_free_direction για να λάβει οδηγίες χωρίς εμπόδια ως string και να επιλέξει μια τυχαία κατεύθυνση από αυτές τις επιλογές. Οι μόνες δυνατές ενέργειες είναι να στρίψει αριστερά, να στρίψει δεξιά ή να μην στρίψει και να συνεχίσει να κινείται. Μόλις επιλέξουμε την απαιτούμενη ενέργεια, το node κάνει publish τις απαιτούμενες βασικές ταχύτητες για αυτήν την ενέργεια με την απαιτούμενη ακολουθία και ολόκληρη αυτή η διαδικασία επαναλαμβάνεται για πάντα.

Η επιλογή των κατευθύνσεων περιορίζεται στον μπροστινό, αριστερό και δεξιό αισθητήρα, οφείλεται στο γεγονός ότι αυτοί είναι 3 σαφείς διευθύνσεις και η λογική στροφής θα είναι απλή (απλώς στρίψτε μέχρι τις 90 μοίρες χρησιμοποιώντας την ταχύτητα στροφής για τη διάρκεια της στροφής).

Όλοι οι υπολογισμοί ταχύτητας γίνονται θεωρώντας ότι το ρομπότ είναι διαφορικής οδήγησης. Επομένως, η βάση που στρέφεται στα αριστερά ή στα δεξιά θα είναι σε ακτίνα μετατόπισης έξω από την βάση, αντί να στρέφεται στον άξονα εντός της βάσης. Ο υπολογισμός της ταχύτητας επίσης δεν λαμβάνει υπόψη την οπίσθια κίνηση για να μειώσει αυτούς τους λογικούς περιορισμούς του διαφορικού ελέγχου για την κίνηση προς τα πίσω και την αδυναμία του ρομπότ να αντιληφθεί την παρουσία τυχόν εμποδίων πίσω από αυτό.

4.2.3 Motor Controller

Στο αρχείο python, το node χρησιμοποιεί την βιβλιοθήκη pigpio για να στέλνει σήματα στους motor drives για να ελέγχει την ταχύτητα στον κάθε τροχό. Αυτό το node κάνει subscribe στα topics του αριστερού και δεξιού pwm από τις ελεγχόμενες τιμές των pwm και περνάει αυτές τις τιμές στα gpio pins. Το node επίσης έχει έναν παράγοντα ασφάλειας για να σταματάει το ρομπότ και λέγεται safety_cut. Αυτό γίνεται μέσω των παραμέτρων του ROS, δηλαδή **rosparam set /start_robot 1** για να ξεκινήσει το ρομπότ και 0 για να σταματήσει κατά την διάρκεια της διαδικασίας λειτουργίας.

4.2.4 IMU data

Το αρχείο python περιέχει την βιβλιοθήκη Adafruit του κατασκευαστή της πλακέτας IMU LSM9DS0. Επίσης υπάρχει το node με το όνομα **talker** και τα μηνύματα του IMU που γίνονται publish στο topic **/imu/data_raw**.

4.2.5 Speed Convertor

Στο αρχείο python, το node κάνει subscribe στο topic base/velocity, όπως επίσης κάνει publish τα pwm για κάθε τροχό.

Για να υπολογίσουμε το pwm που στέλνεται στα motor drives, θα κάνουμε mapping με την ταχύτητα βάσης. Αυτό θα γίνει ως εξής, θα υπολογίσουμε πρώτα το εύρος της ταχύτητας βάσης μετά το εύρος των τιμών του pwm, μετά βρίσκουμε την κλίμακα του εύρους και τέλος υπολογίζονται τα pwm.

Στο αρχείο launch, έχω δημιουργήσει δύο group, ένα για τις παραμέτρους της βάσης και το άλλο για τις παραμέτρους των τροχών. Επιπλέον, δημιουργώ κάποιες παραμέτρους ως topic και δίνω ως values τα εξής:

- wheel/left/vel
- wheel/left/pwm
- wheel/right/vel
- wheel/right/pwm
- base/velocity

5 Περιβάλλον προσομοίωσης (Gazebo)

5.1 Εισαγωγή στο Gazebo

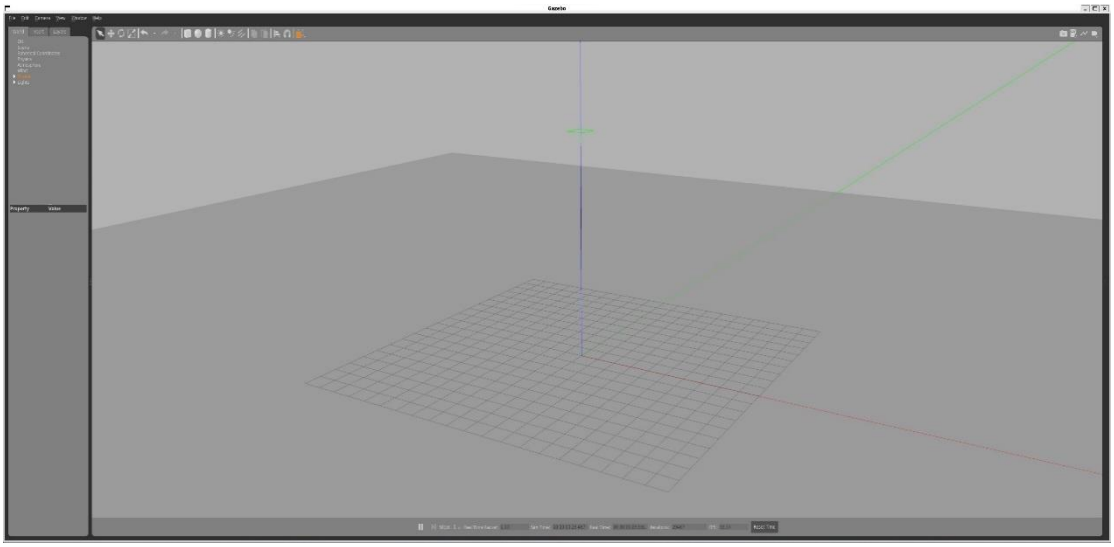
Το Gazebo είναι ένα περιβάλλον προσομοίωσης για ρομποτικές εφαρμογές. Αυτό μας επιτρέπει να ελέγξουμε τους αλγορίθμους που έχουμε ετοιμάσει, να σχεδιάσουμε ρομπότ και να κάνουμε εκπαίδευση σε πιο εξειδικευμένα συστήματα.

Το Gazebo έρχεται προ εγκατεστημένο με το ROS εάν επιλέξουμε την φουλ έκδοση, όπου είναι και ο προεπιλεγμένος simulator. Αυτό μας παρέχει πλήρη επιλογή για αισθητήρες, 3D γραφικά, έτοιμα ρομποτικά μοντέλα και πρόσθετα για τα ρομπότ.

Τρέχοντας μια προσομοίωση μας παρέχει όλες τις πληροφορίες που χρειαζόμαστε σχετικά με το περιβάλλον και τα ρομπότ. Επομένως δεν χρειάζεται να τεστάρουμε τον κώδικα μας σε ένα πραγματικό ρομπότ για να βρούμε προβλήματα και να κάνουμε ρύθμιση των παραμέτρων [7],[8].

5.2 Το περιβάλλον του Gazebo

Το περιβάλλον του Gazebo φαίνεται στην επόμενη εικόνα 32 και για να το εκκινήσουμε τρέχουμε την εντολή **gazebo** στο terminal, αυτό θα μας φορτώσει ένα άδειο περιβάλλον χωρίς κάποιο ρομπότ και εμπόδια για παράδειγμα.



Εικόνα 32: Gazebo

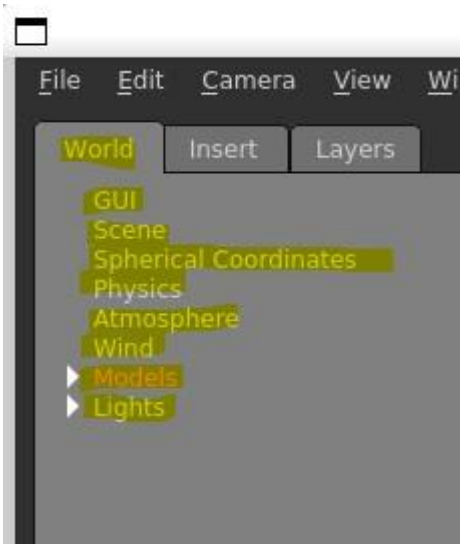
Το Gazebo αποτελείται από διάφορα αντικείμενα, όπως το **World**, τα **Models**, τα **Plugins** και άλλα. Επίσης έχει και διάφορα εργαλεία που είναι πολύ χρήσιμα, όπως την μπάρα εργαλείων, τον επιλογέα (selector), το εργαλείο μετατόπισης αντικειμένου, το εργαλείο περιστροφής και το εργαλείο για αλλαγή μεγέθους ενός αντικειμένου. Μπορούμε επίσης να αποθηκεύσουμε ένα στιγμιότυπο, να κάνουμε καταγραφή σε βίντεο της προσομοίωσης, να πάρουμε αρχείο καταγραφής (log) και ακόμα να κάνουμε plot από το Gazebo.

- **World και Models**

  Το world είναι η προσομοίωση ολόκληρου το περιβάλλοντος, αυτό περιέχει τα πάντα και περιγράφεται με την κατάληξη "**.world**" σε αρχείο, η φόρμα του αρχείου είναι SDF (Simulation Description Format). Η δομή του SDF είναι ένα περιβάλλον στο

οποίο τα μοντέλα μπορούν να δημιουργηθούν και να προσομοιωθούν από μια μηχανή φυσικής.

Στην επόμενη εικόνα 33 βλέπουμε τι περιλαμβάνει το **world**, στην ουσία τι περιγράφεται μέσα στο αρχείο "**.world**".



Εικόνα 33: Το world στο περιβάλλον προσομοίωσης

- **Plugins**

Τα πρόσθετα του Gazebo χρησιμοποιούνται για την διασύνδεση με εξωτερικά προγράμματα. Υπάρχουν δυο τρόποι για να εκκινήσουμε τέτοια πρόσθετα, είτε μέσω command line, είτε με τα SDF αρχεία. Όταν δουλεύουμε με το ROS, τα πρόσθετα φορτώνονται στο αρχείο SDF αφού χρησιμοποιούμε τα launch αρχεία για να ξεκινήσουμε μια προσομοίωση. Αυτά τα πρόσθετα είναι χρήσιμα για τη διασύνδεση της προσομοίωσης του Gazebo και των προγραμμάτων στο ROS.

- **Toolbar**

Υπάρχει μια μπάρα εργαλείων στην κορυφή του παράθυρου προσομοίωσης, αυτή φαίνεται στην επόμενη εικόνα 34.

Εικόνα 34: Toolbar

Εμείς θα δούμε τις πρώτες τέσσερεις από αυτές στο toolbar, με σειρά από αριστερά προς δεξιά. Η πρώτη είναι ο επιλογέας (selector) όπου είναι μια λειτουργία επιλογής, η δεύτερη είναι η λειτουργία μετατόπισης για να μετακινούμε τα αντικείμενα, η τρίτη είναι η λειτουργία περιστροφής για να περιστρέφουμε τα αντικείμενα στους τρεις άξονες, και η τέταρτη είναι η λειτουργία αλλαγής μεγέθους για το μεγάλωμα ή σμίκρυνση ενός αντικειμένου. Τέλος με το ποντίκι υπάρχουν άλλες τρεις επιλογές, με το αριστερό κουμπί μετακινούμε το περιβάλλον, με το δεξί κάνουμε zoom-in και zoom-out το οποίο μπορεί να γίνει και με τα ροδέλα και με το πάτημα της ροδέλας γίνεται περιστροφή γύρω από κάποιο σημείο.

5.3 Δημιουργία ενός ρομπότ

Το Gazebo χρησιμοποιεί την δομή SDF για να περιγράψει τα worlds, models, robots και άλλα απαραίτητα στοιχεία για να χτιστεί μια ολόκληρη προσομοίωση. Τα εργαλεία του ROS είναι φτιαγμένα για URDF (Unified Robotic Description Format), όπως για παράδειγμα το Rviz. Το Gazebo μετατρέπει τα URDF αρχεία σε SDF οπότε εμφανίζεται ένα ρομπότ ορισμένο σε URDF στην προσομοίωση.

5.3.1 Αρχεία URDF και Xacro

Τα URDF είναι μια συλλογή αρχείων που περιγράφουν τη φυσική περιγραφή ενός ρομπότ στο ROS. Αυτά τα αρχεία χρησιμοποιούνται από το ROS για να ξέρει ο υπολογιστής στην δική του γλώσσα πώς μοιάζει πραγματικά το ρομπότ στην πραγματική ζωή.

Τα αρχεία URDF χρειάζονται προκειμένου το ROS να κατανοήσει και να είναι σε θέση να προσομοιώσει καταστάσεις με το ρομπότ προτού κάποιος μηχανικός ασχοληθεί πραγματικά με το ρομπότ.

Η λέξη xacro προέρχεται από την σύνθεση των λέξεων XML και Macro. Αυτά τα αρχεία καθιστούν δυνατή την δημιουργία πιο οργανωμένων και συντομότερων αρχείων XML, δηλαδή περιλαμβάνουν την περιγραφή του ρομπότ, για παράδειγμα, το όνομα του ρομπότ, τις φυσικές ιδιότητες του ρομπότ όπως το σασί, τους τροχούς και το βοηθητικό τροχό (caster wheel) του ρομπότ. Επίσης περιλαμβάνει τις φυσικές ιδιότητες των αισθητήρων, του IMU. Όλα αυτά μεταξύ τους συνδέονται με τις ονομασίες link και joint.

Στην επόμενη εικόνα 35 βλέπουμε ένα δείγμα για το τι είναι αυτά τα αρχεία και πως μοιάζουν.

```
<xacro:property name="chassisHeight" value="0.1"/>
<xacro:property name="chassisLength" value="0.4"/>
<xacro:property name="chassisWidth" value="0.2"/>
<xacro:property name="chassisMass" value="50"/>

<xacro:property name="casterRadius" value="0.05"/>
<xacro:property name="casterMass" value="5"/>

<xacro:property name="wheelWidth" value="0.05"/>
<xacro:property name="wheelRadius" value="0.1"/>
<xacro:property name="wheelPos" value="0.2"/>
<xacro:property name="wheelMass" value="5"/>

<xacro:property name="imuHeight" value="0.005"/>
<xacro:property name="imuSize" value="0.05"/>
<xacro:property name="imuMass" value="0.01"/>

<xacro:property name="sonarHeight" value="0.025"/>
<xacro:property name="sonarThickness" value="0.005"/>
<xacro:property name="sonarWidth" value="0.05"/>
<xacro:property name="sonarMass" value="0.02"/>
```

Εικόνα 35: Οι ιδιότητες του αρχείου xacro

5.3.2 Σύνδεσμοι (links) και Αρθρώσεις (joints)

Οι σύνδεσμοι πρέπει να περιέχουν τα εξής χαρακτηρίστηκα, inertial, collision και visual. Ας δούμε τι είναι αυτά λεπτομερώς. Στην εικόνα 36 βλέπουμε ένα παράδειγμα πως είναι αυτά.

- **Inertial:** Το inertial περιέχει την μάζα (mass), την θέση και τον προσανατολισμό (origin) και τον πίνακα αδράνειας (inertia matrix).

- **Collision και Visual:** Το collision και το visual είναι ταυτόσημα χαρακτηρίστηκα, αλλά έχουν διαφορετικούς σκοπούς. Το πρώτο είναι για την γεωμετρία του αντικειμένου, για παράδειγμα, ενός τοίχου και το δεύτερο για την γεωμετρία της απεικόνισης. Αυτά πρέπει να έχουν ίδιες τιμές στο tag **geometry**, για να μην δημιουργηθεί σύγχυση, για παράδειγμα, ενώ μπορεί να φαίνεται στο περιβάλλον προσομοίωσης ένας τοίχος ως εμπόδιο αλλά να δούμε το ρομπότ μας να περνάει μέσα από τον τοίχο.

```
<link name="chassis">
  <collision>
    <origin xyz="0 0 ${wheelRadius}" rpy="0 0 0"/>
    <geometry>
<box size="${chassisLength} ${chassisWidth} ${chassisHeight}"/>
    </geometry>
  </collision>

  <visual>
    <origin xyz="0 0 ${wheelRadius}" rpy="0 0 0"/>
    <geometry>
<box size="${chassisLength} ${chassisWidth} ${chassisHeight}"/>
    </geometry>
    <material name="orange"/>
  </visual>

  <inertial>
    <origin xyz="0 0 ${wheelRadius}" rpy="0 0 0"/>
    <mass value="${chassisMass}"/>
    <xacro:box_inertia m="${chassisMass}" x="${chassisLength}" y="${chassisWidth}" z="${chassisHeight}"/>
  </inertial>
</link>
```

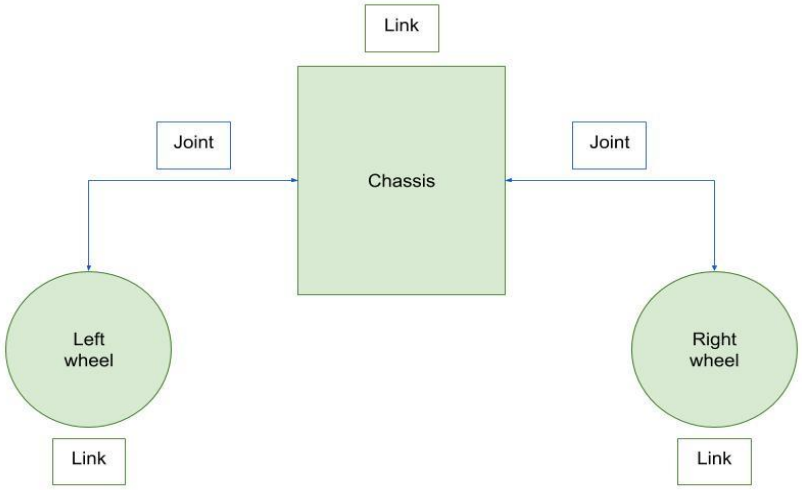
Εικόνα 36: Περιγραφή ενός link

Οι αρθρώσεις χρησιμοποιούνται για τη σύνδεση των συνδέσμων και τη δημιουργία μιας σχέσης κίνησης μεταξύ τους, για παράδειγμα, στην επόμενη εικόνα 37 βλέπουμε πως θα ενώσουμε έναν τροχό με το σασί.

```
<joint name="base_joint" type="fixed">
  <parent link="footprint"/>
  <child link="chassis"/>
</joint>
```

Εικόνα 37: Περιγραφή ενός joint

Στην επόμενη εικόνα 38 βλέπουμε ένα γράφημα σύνδεσης μεταξύ links και joints.



Εικόνα 38: Γράφημα διασύνδεσης μεταξύ links και joints [www.google.gr]

5.3.3 Αρχείο Gazebo

Στα αρχεία αυτά βλέπουμε τα plugin και περιλαμβάνονται με το tag gazebo, για να συνδέσουμε το ROS με το Gazebo και να κάνουμε δυνατή την μετακίνηση του ρομπότ κάνοντας publish σε ένα ROS topic.

Ένα από αυτά τα plugin είναι σίγουρα ο **differential_drive_controller**, όπου βλέπουμε ότι περιέχει σίγουρα το topic για τις ταχύτητες και το topic για την οδομετρία και προέρχεται από το αρχείο **libgazebo_ros_diff_drive.so**. Κάτι ακόμα που περιέχει είναι το **leftJoint** και **rightJoint** όπου είναι τα ονόματα των joints που έχουν οριστεί στο αρχείο **macros**. Κάτι άλλο επίσης που περιέχει είναι το **wheelSeparation** και **wheelDiameter** όπου είναι η διάμετρος των τροχών και η απόσταση των τροχών μεταξύ τους και έχουν οριστεί στο αρχικό αρχείο. Όλα αυτά που αναφέρθηκαν μπορούμε να τα δούμε στην επόμενη εικόνα 39.

```
<gazebo>
 <plugin name="differential_drive_controller" filename="libgazebo_ros_diff_drive.so">
   <alwaysOn>true</alwaysOn>
   <updateRate>100</updateRate>
   <leftJoint>left_wheel_hinge</leftJoint>
   <rightJoint>right_wheel_hinge</rightJoint>
   <wheelSeparation>${chassisWidth+wheelWidth}</wheelSeparation>
   <wheelDiameter>${2*wheelRadius}</wheelDiameter>
   <torque>20</torque>
   <commandTopic>mymobibot/cmd_vel</commandTopic>
   <odometryTopic>mymobibot/odom_diffdrive</odometryTopic>
   <odometryFrame>odom</odometryFrame>
   <robotBaseFrame>footprint</robotBaseFrame>

   <publishWheelTF>false</publishWheelTF>
   <publishWheelJointState>false</publishWheelJointState>
   <rosDebugLevel>na</rosDebugLevel>
   <wheelAcceleration>0</wheelAcceleration>
   <wheelTorque>5</wheelTorque>
   <publishOdomTF>true</publishOdomTF>
   <odometrySource>world</odometrySource>
   <publishTf>1</publishTf>

 </plugin>
</gazebo>
```

Εικόνα 39: Το plugin differential drive controller

Ένα άλλο plugin είναι το **gazebo_ros_control**, το οποίο προέρχεται από το αρχείο **libgazebo_ros_control.so**. Αυτό το plugin είναι για το όνομα του ρομπότ μας στο Gazebo ώστε να επικοινωνεί με το **ros_control**. Επιπλέον βλέπουμε το plugin που είναι χρήσιμο για την οδομετρία και ονομάζεται **ground_truth** και προέρχεται από την βιβλιοθήκη **libgazebo_ros_p3d.so**. Στην επομένη εικόνα 40 βλέπουμε αυτά που περιεγράφηκαν παραπάνω.

```
<gazebo>
  <plugin name="gazebo_ros_control" filename="libgazebo_ros_control.so">
    <robotNamespace>/mymobibot</robotNamespace>
  </plugin>
</gazebo>

<gazebo>
 <plugin name="ground_truth" filename="libgazebo_ros_p3d.so">
   <frameName>map</frameName>
   <bodyName>chassis</bodyName>
   <topicName>mymobibot/odom</topicName>
   <updateRate>30.0</updateRate>
 </plugin>
</gazebo>
```

Εικόνα 40: Τα plugins gazebo ros control και ground truth

5.3.4 Φάκελος control και PID gains

Ο φάκελος control περιέχει αρχεία για την διασύνδεση μεταξύ των ROS controllers και του Gazebo. Στον φάκελο αυτόν υπάρχουν δύο ακόμα φάκελοι, ο ένας είναι ο **config** και ο άλλος είναι ο **launch**. Στον πρώτο φάκελο υπάρχει ένα αρχείο που **yaml**, αυτό το αρχείο είναι υπεύθυνο για την αποθήκευση των gains του PID και τις ρυθμίσεις του controller, τα οποία φορτώνονται στο param server μέσω του αρχείου **launch** που βρίσκεται μέσα στον φάκελο launch. Στην επόμενη εικόνα 41 βλέπουμε αυτά που αναφέραμε.

```
mymobibot:
  # Publish all joint states -----------------------------------
  joint_state_controller:
    type: joint_state_controller/JointStateController
    publish_rate: 50


  # Effort Controllers ---------------------------------------
  leftWheel_effort_controller:
    type: effort_controllers/JointEffortController
    joint: left_wheel_hinge
    pid: {p: 100.0, i: 0.1, d: 10.0}
  rightWheel_effort_controller:
    type: effort_controllers/JointEffortController
    joint: right_wheel_hinge
    pid: {p: 100.0, i: 0.1, d: 10.0}
```

Εικόνα 41: Αρχείο yaml

Στην εικόνα 42 βλέπουμε ότι στην 4η γραμμή το **rosparam** φορτώνει τις ρυθμίσεις του controller στον server παραμέτρων από το αρχείο **yaml**. Έπειτα βλέπουμε ότι το node με όνομα **controller_spawner** εκκινεί τα δυο joints effort controller **rightWheel** και **leftWheel** τρέχοντας ένα script python το οποίο πραγματοποιεί ένα service call στον **controller_manager** και του λέει ποιους controllers θέλει. Επίσης εκκινεί έναν τρίτο controller ο οποίος κάνει publish τα joint states από όλα τα joints.

```
<launch>

  <!-- Load joint controller configurations from YAML file to parameter server -->
  <rosparam file="$(find mymobibot_control)/config/mymobibot_control.yaml" command="load"/>

  <!-- load the controllers -->
  <node name="controller_spawner" pkg="controller_manager" type="spawner" respawn="false"
  output="screen" ns="/mymobibot" args="joint_state_controller
                rightWheel_effort_controller
                leftWheel_effort_controller"/>


  <!-- convert joint states to TF transforms for rviz, etc -->
  <node name="robot_state_publisher" pkg="robot_state_publisher" type="robot_state_publisher"
  respawn="false" output="screen">
    <param name="robot_description" command="$(find xacro)/xacro '$(find mymobibot_description)/urdf/mymobibot.xacro'" />
    <remap from="/joint_states" to="/mymobibot/joint_states" />
  </node>

</launch>
```

Εικόνα 42: Αρχείο launch

5.3.5 Φάκελος Gazebo

Σε αυτόν τον φάκελο κατασκευάζουμε τον κόσμο (world) του gazebo, δηλαδή, δημιουργούμε τα αντικείμενα που θα υπάρχουν μέσα στο περιβάλλον της προσομοίωσης όπως οι τοίχοι για παράδειγμα. Αυτά τα αντικείμενα μπορεί να είναι οτιδήποτε, από αυτοκίνητα μέχρι κάποιον ταινιόδρομο σε μια παραγωγική μονάδα όπου ένας ρομποτικός βραχίονας κάνει palletizing. Για να γίνει spawn το περιβάλλον θα χρειαστούμε και το αρχείο **launch** που υπάρχει μέσα σε έναν φάκελο launch.

Το περιβάλλον προσομοίωσης το χτίζουμε μέσα στον φάκελο world και αυτό γίνεται με τα αρχεία **world**. Τα αρχεία αυτά περιλαμβάνουν το **ground_plane**, το **sun** και τα αντικείμενα που θα υπάρχουν μέσα στο περιβάλλον προσομοίωσης. Αυτά τα αντικείμενα θα είναι οι τέσσερεις τοίχοι που περιφράζουν το ρομπότ, ακόμα δυο ενδιάμεσοι τοίχοι και δυο κύλινδροι ως εμπόδια.

```
<world name="default">
  <include>
    <uri>model://ground_plane</uri>
  </include>

  <!-- Global light source -->
  <include>
    <uri>model://sun</uri>
  </include>
```

Εικόνα 43: Κατασκευή αρχείου world

5.4 Περιγραφή του αλγόριθμου obstacle avoiding

Όπως αναλύσαμε και στο τέταρτο κεφάλαιο στην ενότητα 4.2.2, ο αλγόριθμος αυτός κάνει το ρομπότ να πραγματοποιεί τυχαία κίνηση στον χώρο και να αποφεύγει τα εμπόδια που εντοπίζουν οι υπέρυθροι αισθητήρες που έχει πάνω του. Στις επόμενες τρεις ενότητες θα δούμε την δομή του.

5.4.1 Φάκελος launch

Στον φάκελο launch έχουμε τα αρχεία εκκίνησης "**improved_random_walkder.launch**" και "**obstacle_solver.launch**". Στο αρχείο **obstacle_solver** έχουμε τις παραμέτρους για τον έλεγχο απόστασης τον εμποδίων με τα εύρη δράσης των αισθητήρων για να συγκριθούν με τις πραγματικές αποστάσεις που μετράνε οι αισθητήρες. Επίσης έχουμε και τις τιμές των παραμέτρων για κάθε topic των αισθητήρων. Επιπλέον κάνουμε εκκίνηση του node **obstacle_solver** που βρίσκεται στο αρχείο python.

Στο αρχείο **improved_random_walker** περιλαμβάνουμε το προηγούμενο αρχείο launch, για να κάνουμε εκκίνηση του node **random_walker** που βρίσκεται στο αρχείο python και επιπλέον έχουμε βάλει κάποιες παραμέτρους για το tuning του αλγορίθμου. Στις παρακάτω εικόνες 44 και 45 βλέπουμε τι περιέχουν τα αρχεία.

```
<launch>

    <param name="aRange_f" value="1.0" />
    <param name="aRange_l" value="0.2" />
    <param name="aRange_r" value="0.2" />
    <param name="aRange_fl" value="0.2" />
    <param name="aRange_fr" value="0.2" />

    <param name="left_sensor_topic" value="/sensor/sonar_L"/>
    <param name="frontleft_sensor_topic" value="/sensor/sonar_FL"/>
    <param name="front_sensor_topic" value="/sensor/sonar_F"/>
    <param name="frontright_sensor_topic" value="/sensor/sonar_FR"/>
    <param name="right_sensor_topic" value="/sensor/sonar_R"/>

    <node pkg="obstacle_avoiding_random_walker" type="obstacle_solver.py" name="obstale_solver" output="screen" />

</launch>
```

Εικόνα 44: Αρχείο obstacle_solver.launch

```
<launch>

    <include file="$(find obstacle_avoiding_random_walker)/launch/obstacle_solver.launch" />

    <param name="cmd_vel_topic" value="/mymobibot/cmd_vel" />
    <param name="straight_vel" value="1.0" />
    <param name="turn_vel" value="1.0" />
    <param name="wait_time" value="3.0" />
    <param name="turn_time" value="3.0" />
    <param name="straight_time" value="1.0" />
    <param name="brake_time" value="0.1" />

    <node pkg="obstacle_avoiding_random_walker" type="improved_random_walker.py" name="random_walker" output="screen" />

</launch>
```

Εικόνα 45: Αρχείο improved_random_walker.launch

Παρακάτω βλέπουμε τις παραμέτρους που αναφέραμε προηγουμένως:

- **cmd_vel_topic**: Αυτό είναι το topic που παίρνει την τιμή "**/mymobibot/cmd_vel**" για να στείλει τις τιμές ταχύτητας για την προσομοίωση, ενώ για το πραγματικό ρομπότ είναι "**/base/velocity**".
- **straight_vel**: Είναι η ταχύτητα βάσης για να κινηθεί ευθεία και μετράτε σε (m/s).
- **turn_vel**: Είναι η ταχύτητα περιστροφής της βάσης και μετράτε σε (rad/s).
- **wait_time**: Είναι ένας χρόνος καθυστέρησης που μπορεί να χρησιμοποιηθεί σε οποιαδήποτε περίπτωση μελλοντικά.
- **turn_time**: Η χρονική διάρκεια περιστροφής του ρομπότ αφού ξεκινήσει να στρίβει.
- **straight_time**: Η χρονική διάρκεια κίνησης του ρομπότ αφού ξεκινήσει να κινείται ευθεία.
- **brake_time**: Η χρονική διάρκεια που θα φρενάρει το ρομπότ με την εφαρμογή αντίθετης ροπής στους τροχούς.

5.4.2 Φάκελος scripts

Στον φάκελο scripts έχουμε τα αρχεία python **improved_random_walker.py** και **obstacle_solver.py**. Το αρχείο **obstacle_solver.py** περιλαμβάνει τον κώδικα για την επίλυση των λειτουργιών κάθε αισθητήρα από τους πέντε, επίσης περιλαμβάνει και τα topic τους που γίνονται subscribe. Επιπλέον γίνεται χρήση της υπηρεσίας (service) για να γίνει έλεγχος για εμπόδια από τους αισθητήρες. Στην επόμενη εικόνα 46 βλέπουμε την function στην οποία γίνεται ο έλεγχος για εμπόδια.

```
def check_for_obstacles(self, req):
    # Service callback function that checks for any obstacles around the robot using the current values of ranges from those sensors
    # and return the obstacle-free directions
    rospy.loginfo('Checking for Obstacles..')
    response = DirectionsResponse()                  # Creating a service response data type
    response.directions.data = ""                    # Emptying the String for response
    if self.ranges['l'] > self.action_ranges['l']:   # Checking if the left sensor range is greater than its action range
        response.directions.data += 'l'              # if yes, then add that direction as obstacle-free
    # Checking if the front, front_left and front_right sensors range is greater than its action range
    if self.ranges['f'] > self.action_ranges['f'] and self.ranges['fl'] > self.action_ranges['f'] and self.ranges['fr'] > self.action_ranges['f']:
        response.directions.data += 'f'
    if self.ranges['r'] > self.action_ranges['r']:   # Checking if the right sensor range is greater than its action range
        response.directions.data += 'r'              # if yes, then add that direction as obstacle-free
    rospy.loginfo('Obstacle-free directions : ' + response.directions.data + '\n')
    return response                                  # return the string of obstacle-free direction as response
```

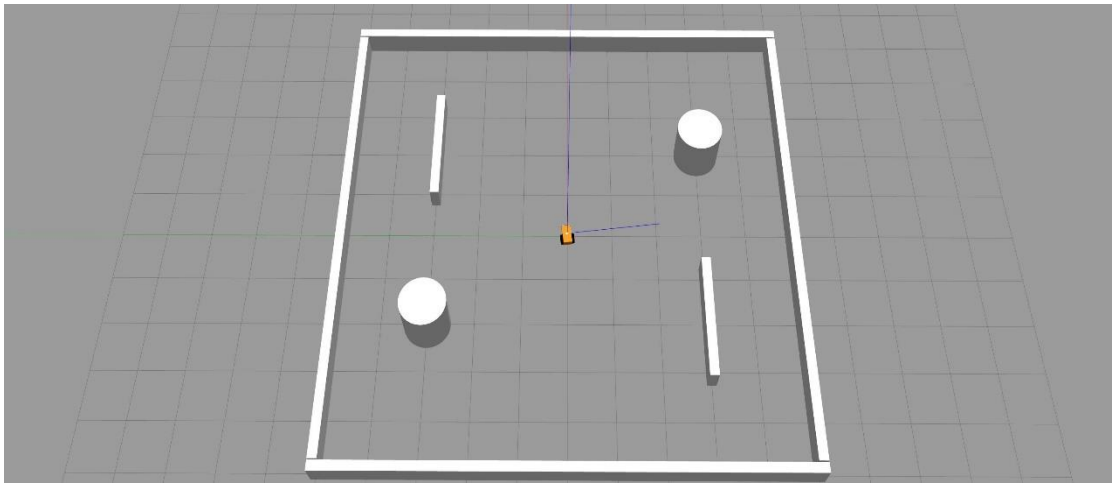
Εικόνα 46: Η συνάρτηση για τον έλεγχο εμποδίων

Στην συγκεκριμένη function βλέπουμε ότι συγκρίνει τις πραγματικές μετρήσεις του αριστερού, του δεξιού και του εμπρόσθιου αισθητήρα με αυτές που το έχουμε ορίσει στα **action_ranges**, εάν μια από τις πραγματικές είναι μεγαλύτερη από την ορισμένη τότε του λέει να στρίψει ανάλογα δεξιά, αριστερά ή να συνεχίσει να κινείται ευθεία για επιλογή κατεύθυνσης αφού έχει σταματήσει από κάποιο εμπόδιο.

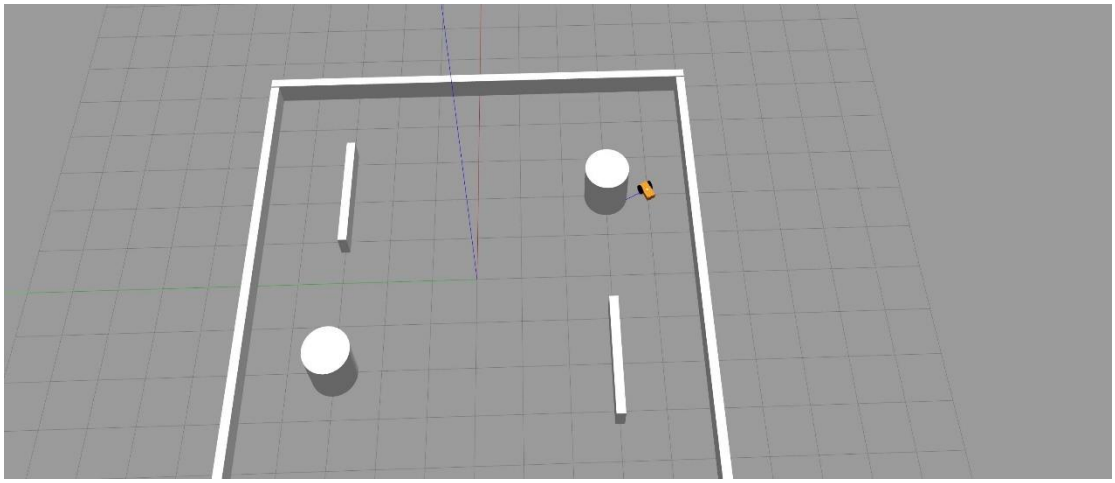
Το αρχείο "**improved_random_walker.py**" περιλαμβάνει μια συνάρτηση η οποία αναθέτει τις τιμές των παραμέτρων από το launch στον παρόν αρχείο, επίσης ακόμα μια συνάρτηση η οποία δέχεται τα response των κατευθύνσεων από το άλλο αρχείο python και τα μετατρέπει σε κίνηση. Επιπλέον μια συνάρτηση για την τυχαία επιλογή κατεύθυνσης, ακόμα μια συνάρτηση για το σταμάτημα της κίνησης με την εφαρμογή ανάποδης ροπής στους τροχούς με αποτέλεσμα να φρενάρει. Τέλος έχουμε και την συνάρτηση για να κάνει publish τις εντολές ταχύτητας και να αποφασίζει πως θα κινηθεί.

5.5 Στιγμιότυπα του Gazebo από το ρομπότ διαφορικής οδήγησης

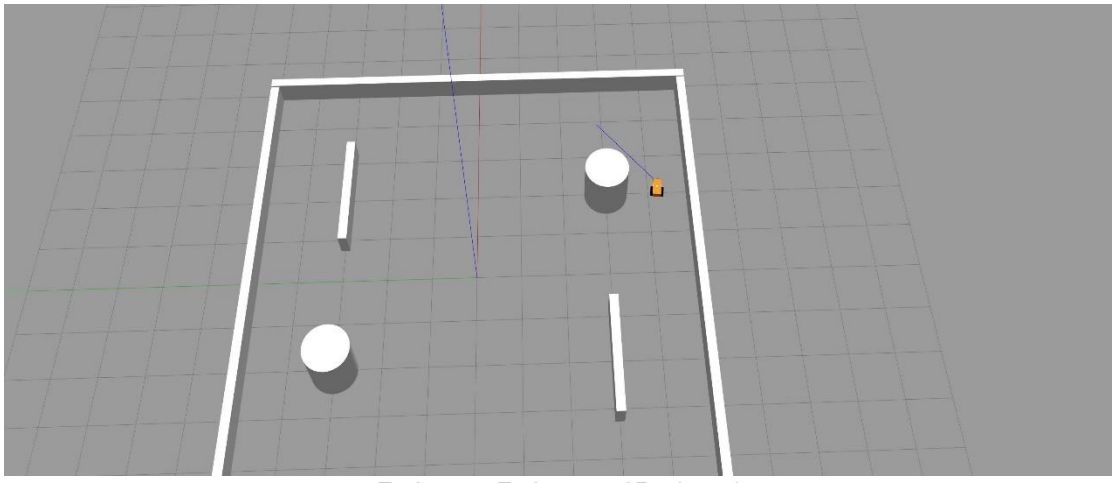
Σε αυτήν την ενότητα παραθέτω μερικά στιγμιότυπα από το Gazebo την ώρα της προσομοίωσης.



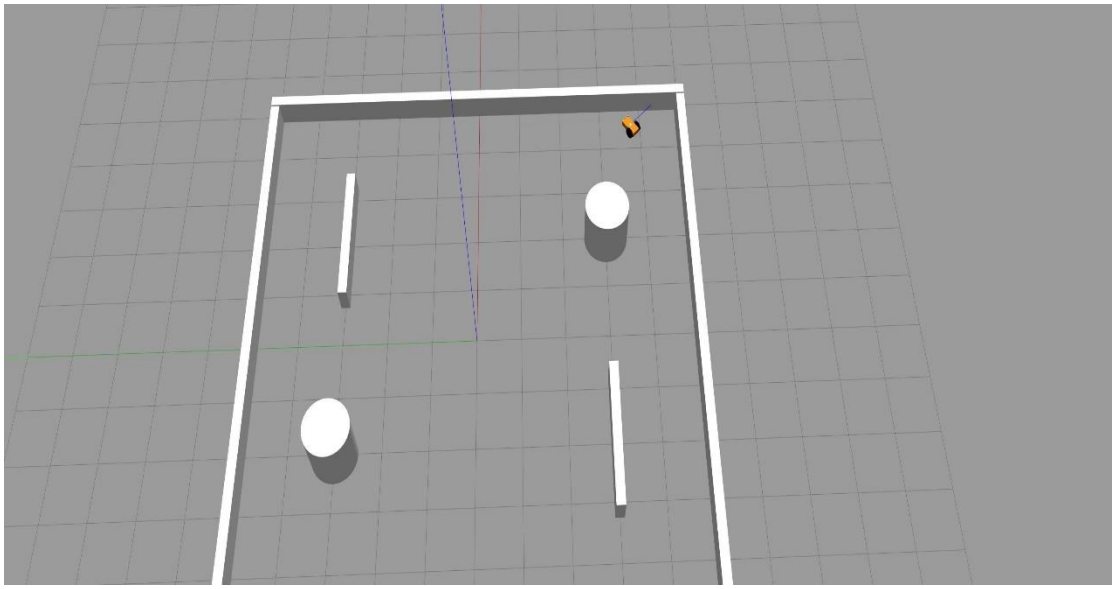
Εικόνα 47: Αρχικό σημείο



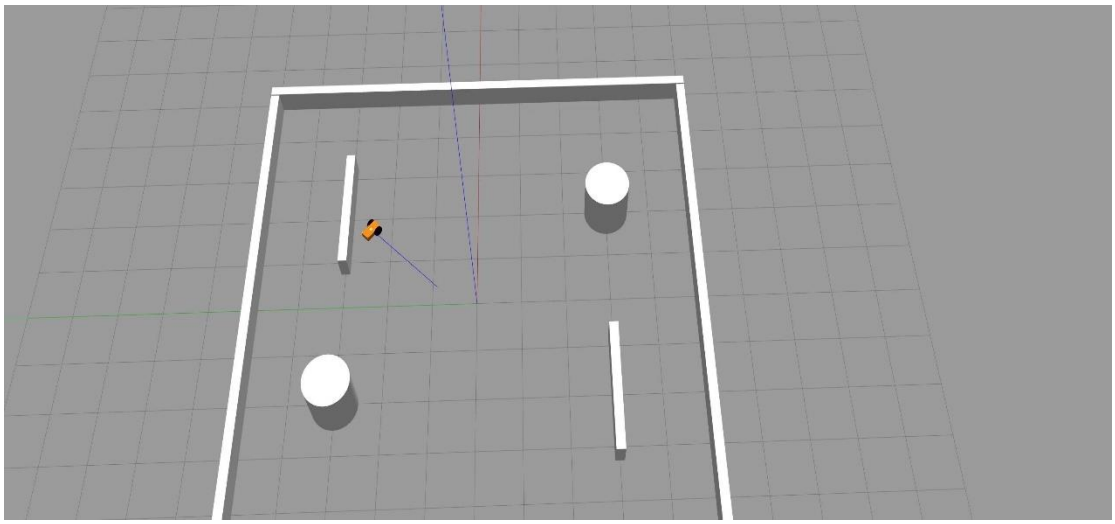
Εικόνα 48: Επόμενο στάδιο



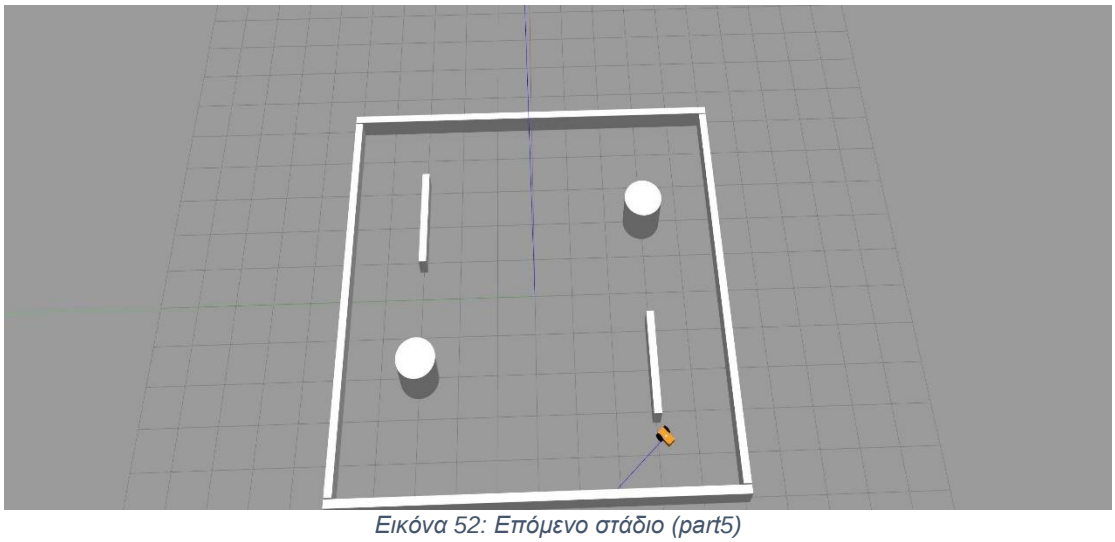
Εικόνα 49: Επόμενο στάδιο (part2)



Εικόνα 50: Επόμενο στάδιο (part3)



Εικόνα 51: Επόμενο στάδιο (part4)

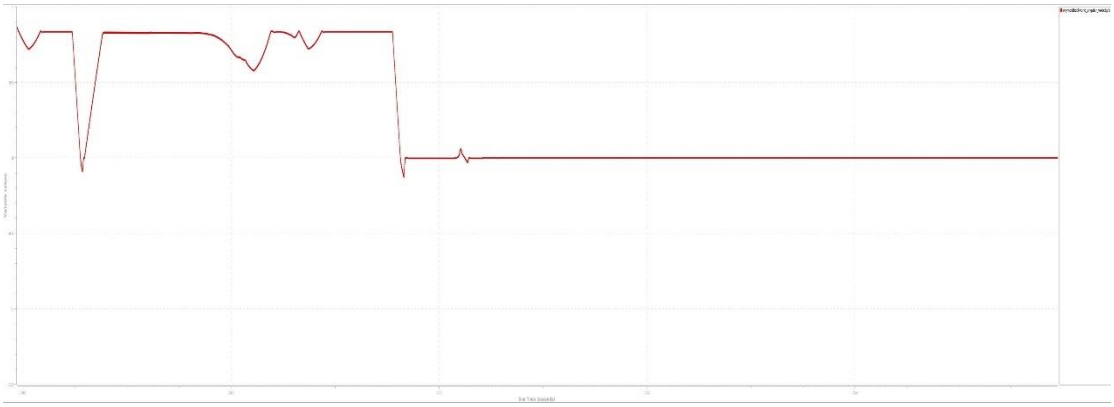


Εικόνα 52: Επόμενο στάδιο (part5)

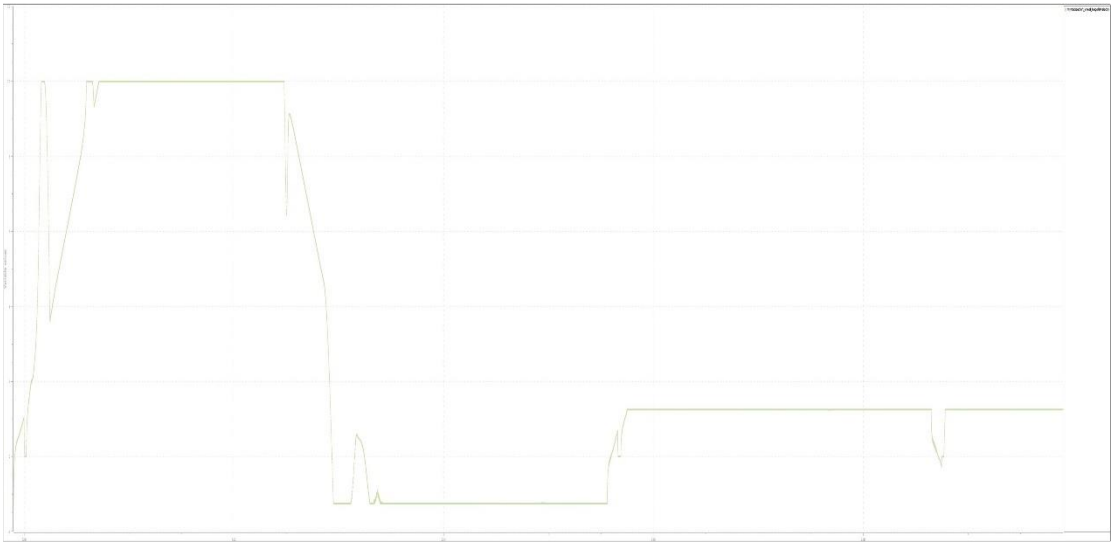
5.6 Plots

Τα plots προέρχονται από περιβάλλον του Gazebo.

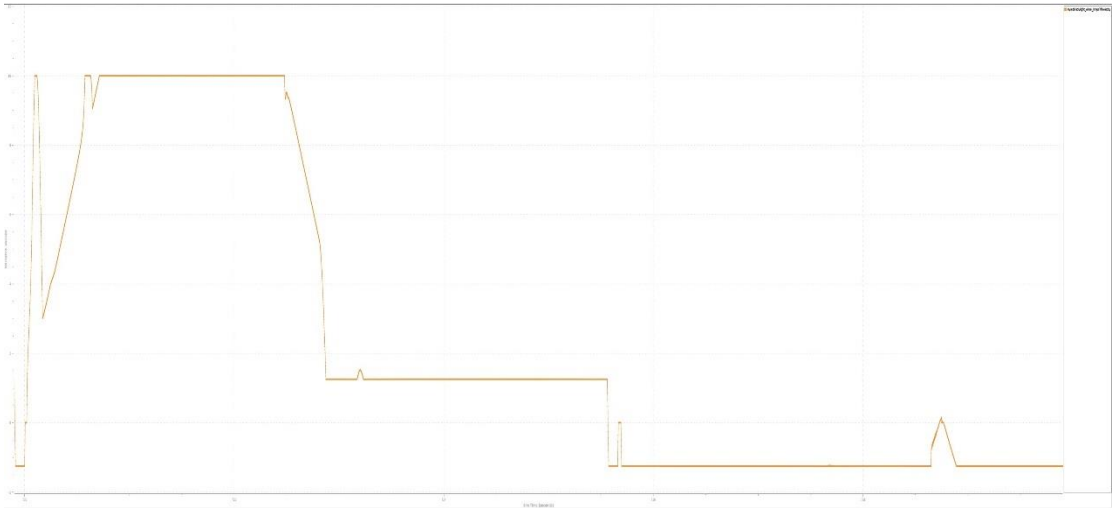
Εικόνα 53: Γραμμική ταχύτητα x



Εικόνα 54: Γωνιακή ταχύτητα z



Εικόνα 55: Ταχύτητα αριστερού τροχού



Εικόνα 56: Ταχύτητα δεξιού τροχού

Στις δύο πρώτες εικόνες 53 και 54 παρατηρούμε ότι η ταχύτητα βάσης του ρομπότ γίνεται ίση με 1 m/s, δηλαδή με αυτήν που του έχουμε ορίσει στις παραμέτρους. Επίσης η γωνιακή ταχύτητα θα είναι περίπου 0,82 rad/s, βάσει τον χρόνο που έχουμε βάλει 3 sec θα κάνει στροφή περίπου 141 μοίρες.

Στις άλλες δύο εικόνες 55 και 56 βλέπουμε ότι ταχύτητες των τροχών είναι 10, αυτό το δέκα είναι σε revolution/sec. Η ακτίνα του τροχού στην προσομοίωση είναι 0.1m και η ταχύτητα βάσης είναι 1m/s.

6 Πειραματικές διαδικασίες

Στο κεφάλαιο αυτό θα δούμε τις πειραματικές μετρήσεις που πραγματοποιήθηκαν με την πειραματική διάταξη. Αρχικά θα υπολογίσουμε την ταχύτητα βάσης του ρομπότ με βάση τους παλμούς (pwm) που στείλαμε στους τροχούς.

Στο 4ο κεφάλαιο στην ενότητα 4.1 αναφερθήκαμε στην βιβλιοθήκη που χρειαστήκαμε την **pigpio,** από την ιστοσελίδα του GitHub, όπου την βρήκαμε συνοδεύεται και από κάποια παραδείγματα με scripts, ένα από αυτά είναι και ένα python script για γεννήτρια παλμών σερβοκινητήρα (Servo pulse generator). Το script αυτό δημιουργεί έναν παλμό για ένα ή περισσότερα GPIO, οι παλμοί μπορούν να κυμαίνονται μεταξύ 1000μs και 2000μs για να έχουμε διαφορετικές ταχύτητες ενώ αν στείλουμε παλμό 1500μs αυτό αντιστοιχεί σε μηδενική ταχύτητα. Για να βρούμε όμως ποια είναι η πραγματική ταχύτητα βάσης σε m/s θα πρέπει να διενεργήσουμε πειραματικές μετρήσεις στο οδόστρωμα (πάτωμα), δηλαδή τοποθετώντας μια μετροταινία παράλληλα με την διαδρομή του ρομπότ. Η διαδικασία φαίνεται στην επόμενη εικόνα 57.

Εικόνα 57: Πειραματική μέτρηση της ταχύτητας βάσης σε m/s

Να σημειώσουμε ότι για να είχαμε ακριβείς μετρήσεις θα έπρεπε να είμαστε σε ιδανικό περιβάλλον (αντιολισθητικό πάτωμα) για την αποφυγή ολίσθησης. Επιπλέον τα λάστιχα από τους τροχούς είναι φαγωμένα και φθαρμένα από την παλαίωση με αποτέλεσμα να άλλαζε πορεία σε μεγάλες αποστάσεις. Στον πίνακα 2 βλέπουμε τις μετρήσεις που έγιναν.

Παλμοί [μs]	**Ταχύτητα βάσης σε [m/s]**
1000	~1.3 (CCW)
1300	~0.6 (CCW)
1350	~0.33 (CCW)
1500	0
1650	~0.33 (CW)
1700	~0.6 (CW)
2000	~1.3 (CW)

Πίνακας 2: Πειραματικές μετρήσεις από παλμούς σε m/s

Βλέπουμε ότι από 1000 μέχρι 1499 θα έχουμε κίνηση προς τα πίσω (CCW – Counter Clockwise), στα 1500 θα έχουμε μηδενική κίνηση, ενώ από 1501 έως 2000 θα έχουμε εμπρόσθια κίνηση (CW – Clockwise). Σημειώνετε ότι το ρομπότ παρέμενε σε μηδενική κίνηση ακόμα και με 1520 και 1480, άρα τα υπάρχει μια απόκλιση.

Τώρα για να βρούμε και την περιστροφική ταχύτητα του κάθε τροχού αρκεί να κάνουμε χρήση των εξισώσεων που είδαμε στην ενότητα 2.3 στο 2ο κεφάλαιο, επομένως θα έχουμε:

$$V_r = \frac{2V + \omega L}{2R}$$

$$V_l = \frac{2V - \omega L}{2R}$$

Αφού η γωνιακή ταχύτητα είναι μηδέν λόγο του ότι έχουμε γραμμική ταχύτητα, οι εξισώσεις θα απλοποιηθούν σε γραμμική ταχύτητα V ως προς την ακτίνα των τροχών R, με το αποτέλεσμα να έχει μονάδες σε [r/s ή 1/s ή revolution/second], δηλαδή, περιστροφές το δευτερόλεπτο. Ας δούμε παρακάτω ένα παράδειγμα όπου θα επιλέξαμε τα 0.6[m/s] και η ακτίνα του τροχού είναι 0.04[m].

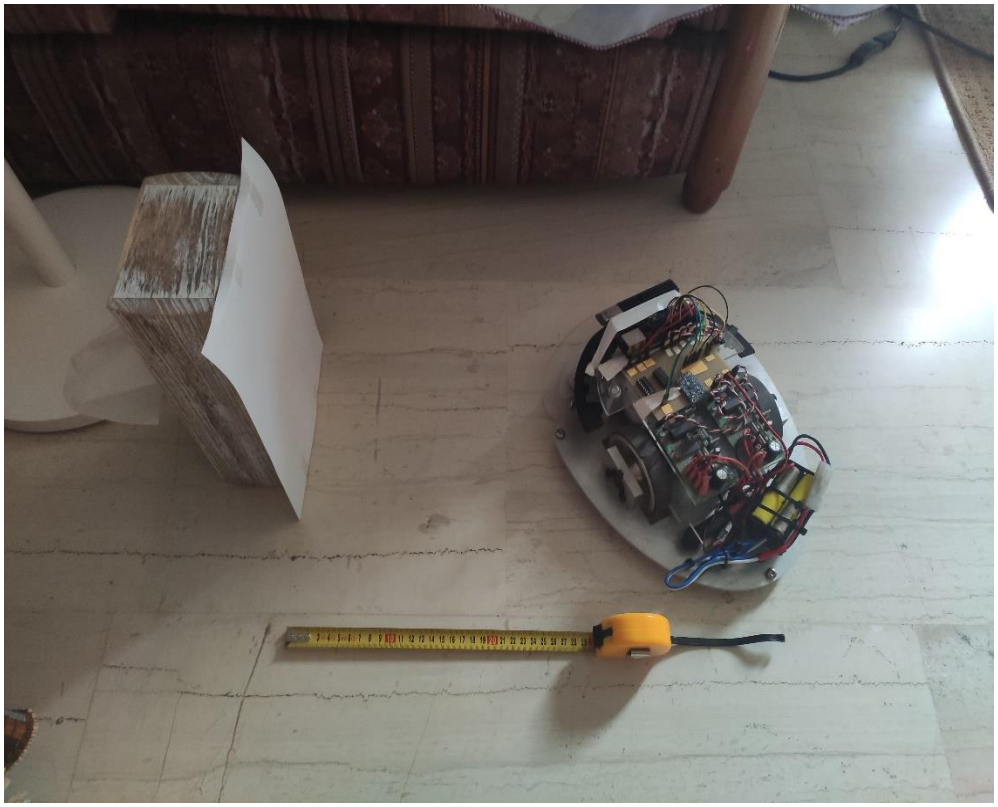
$$V_r = \frac{V}{R} = \frac{0.6\ m/s}{0.04\ m} = 15\ [rps]$$

$$V_l = \frac{V}{R} = \frac{0.6\ m/s}{0.04\ m} = 15\ [rps]$$

Τώρα για να έχουμε το αποτέλεσμα σε [rad/s] θα πολλαπλασιάσουμε με 6.28 που είναι το 1[rps], άρα θα έχουμε ότι η γωνιακή ταχύτητα του κάθε τροχού θα είναι 94.25 [rad/s].

Στην συνέχεια θα δούμε τις πειραματικές μετρήσεις που πραγματοποιήθηκαν για το πόσο ακριβείς είναι οι αισθητήρες. Οι μετρήσεις πραγματοποιήθηκαν για τους δυο πλευρικούς (δεξί και αριστερό) και τον μπροστινό και για τις αποστάσεις 30, 60, 90 και 120 cm. Στις επόμενες τρεις εικόνες 58, 59 και 60 βλέπουμε την πειραματική διαδικασία όπου τοποθετήθηκε εμπόδιο σε άσπρο χρώμα.

Εικόνα 58: Έλεγχος των αισθητήρων μέτρησης απόστασης (part 1)



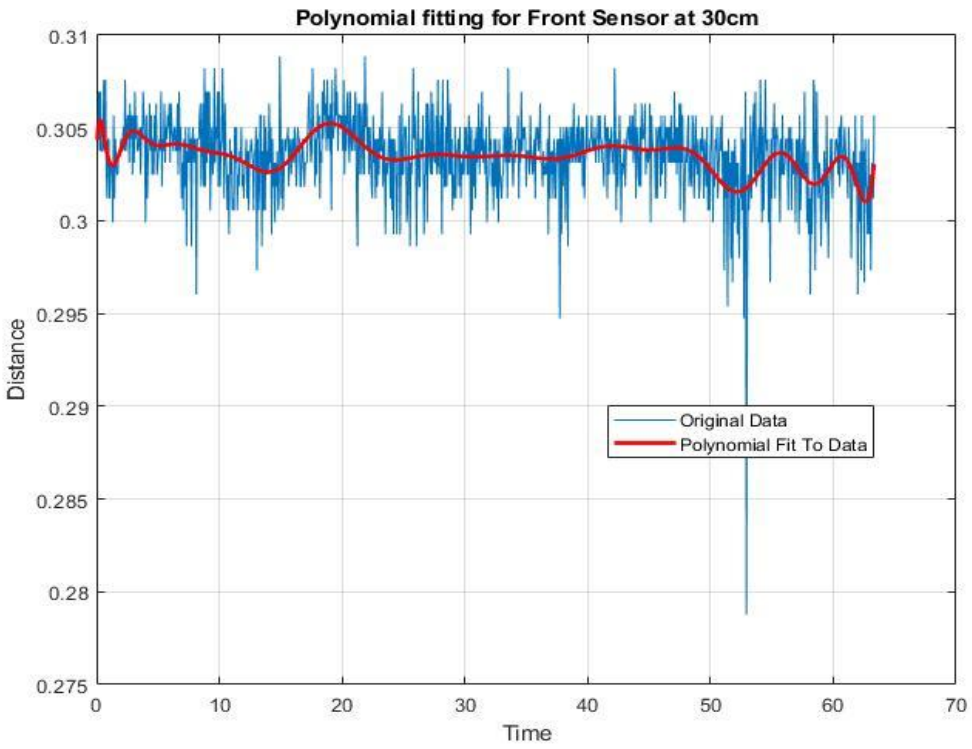
Εικόνα 59: Έλεγχος των αισθητήρων μέτρησης απόστασης (part 2)

Εικόνα 60: Έλεγχος των αισθητήρων μέτρησης απόστασης (part 3)

Έγινε χρήση του MATLAB με τις συναρτήσεις polyfit και polyval. Η polyfit είναι μια συνάρτηση που υπολογίζει το πολυώνυμο ελαχίστων τετραγώνων και η polyval κάνει εκτίμηση του πολυωνύμου για να ταιριάξει μια καμπύλη στα δεδομένα.

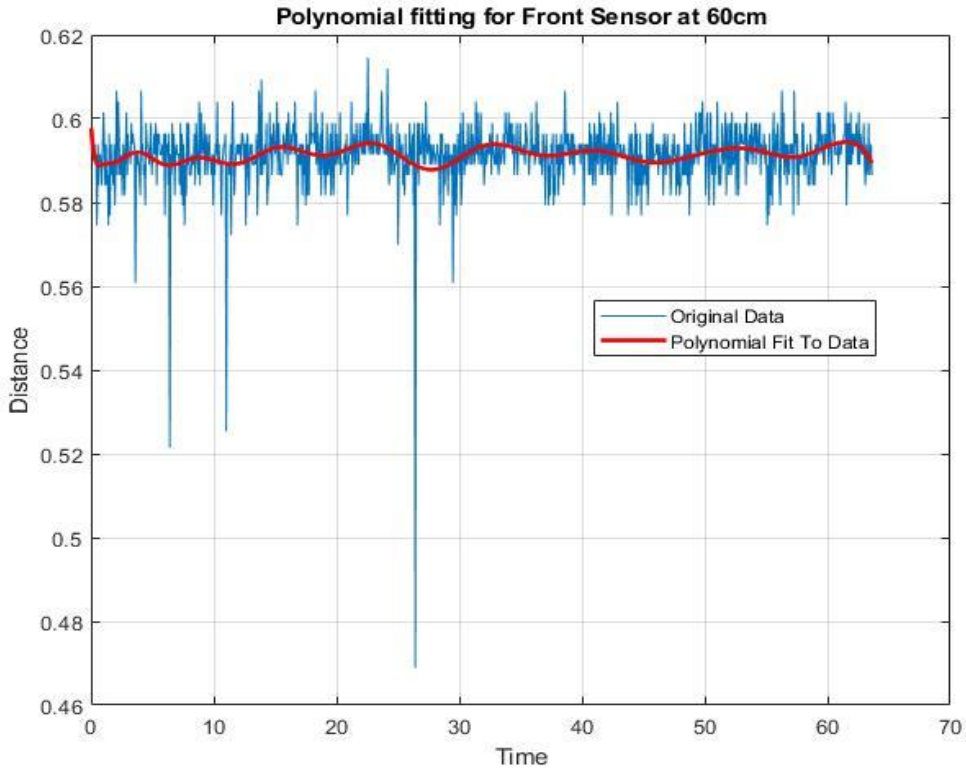
Μπροστινός αισθητήρας

Για τα 30 cm:



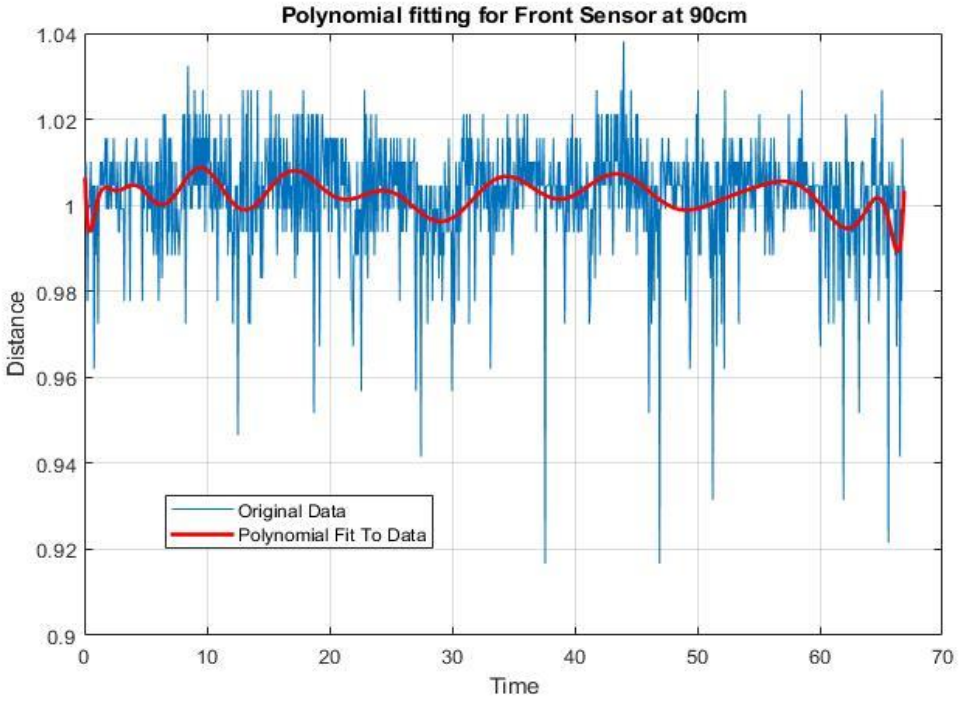
Εικόνα 61: Μπροστινός αισθητήρας στα 30cm

Για τα 60 cm:



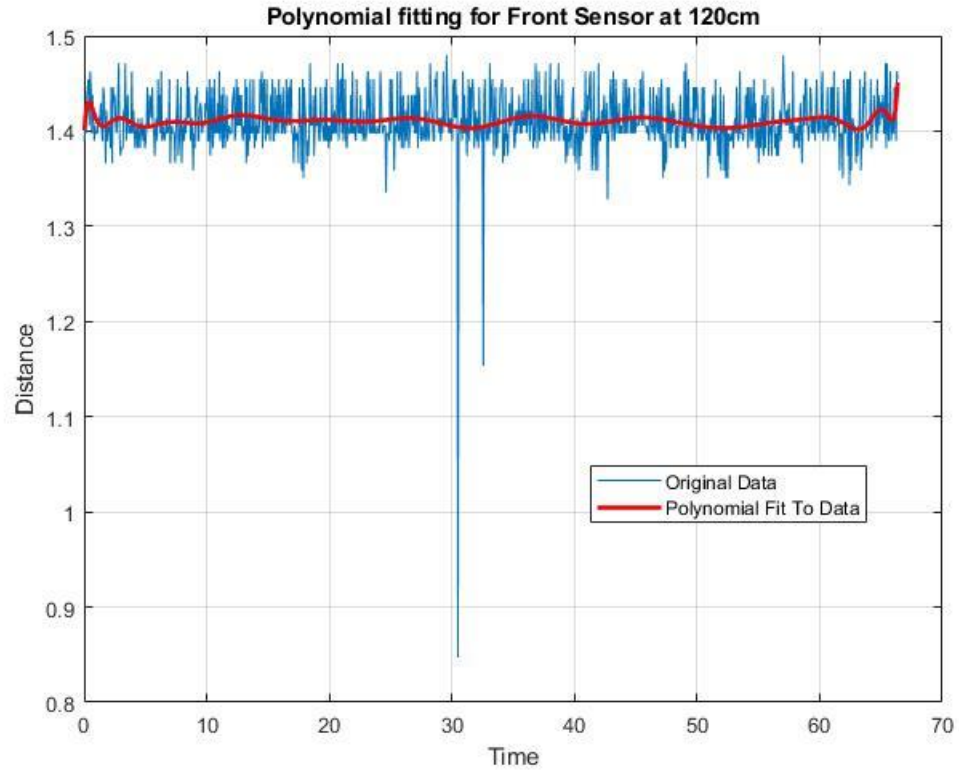
Εικόνα 62: Μπροστινός αισθητήρας στα 60 cm

Για τα 90 cm:



Εικόνα 63: Μπροστινός αισθητήρας στα 90 cm

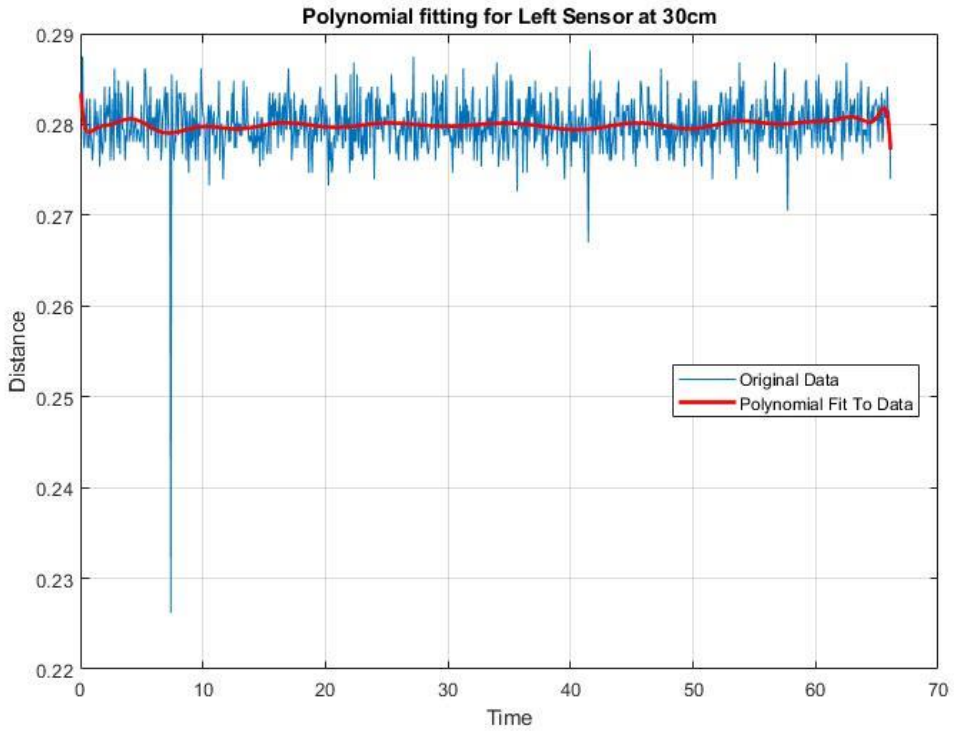
Για τα 120 cm:



Εικόνα 64: Μπροστινός αισθητήρας στα 120 cm

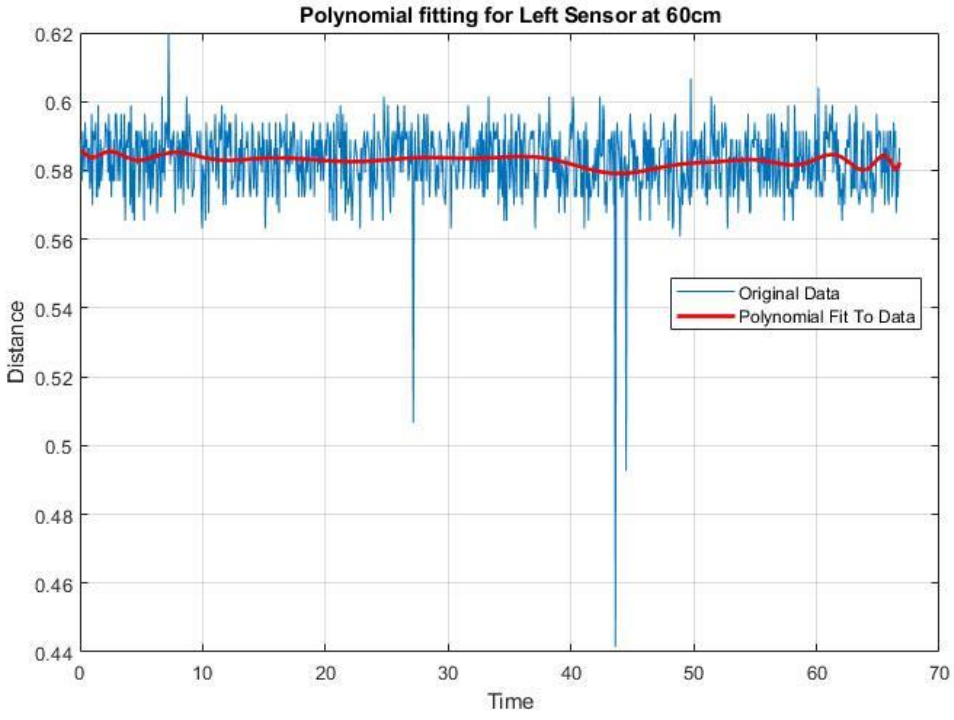
Αριστερός αισθητήρας

Για τα 30 cm:



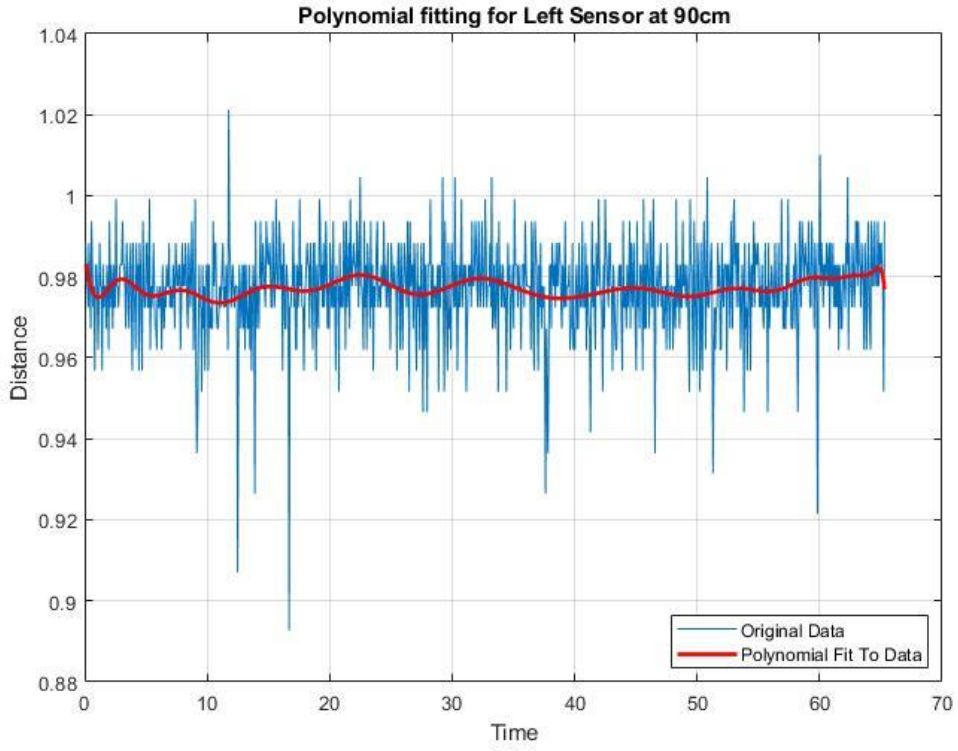
Εικόνα 65: Αριστερός αισθητήρας στα 30 cm

Για τα 60 cm:



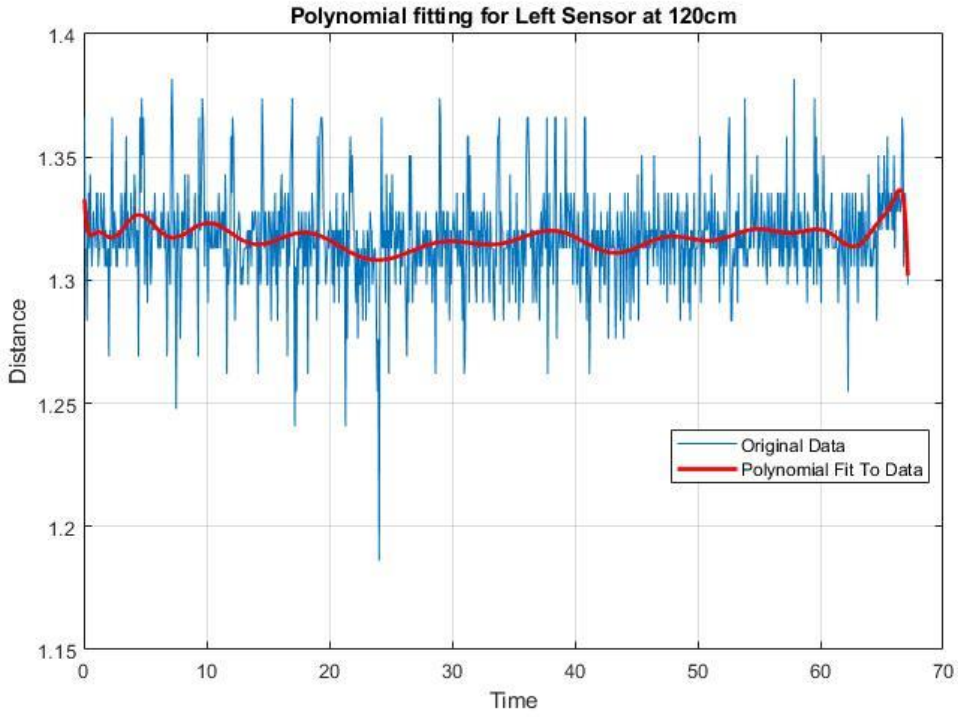
Εικόνα 66: Αριστερός αισθητήρας στα 60 cm

Για τα 90 cm:



Εικόνα 67: Αριστερός αισθητήρας στα 90 cm

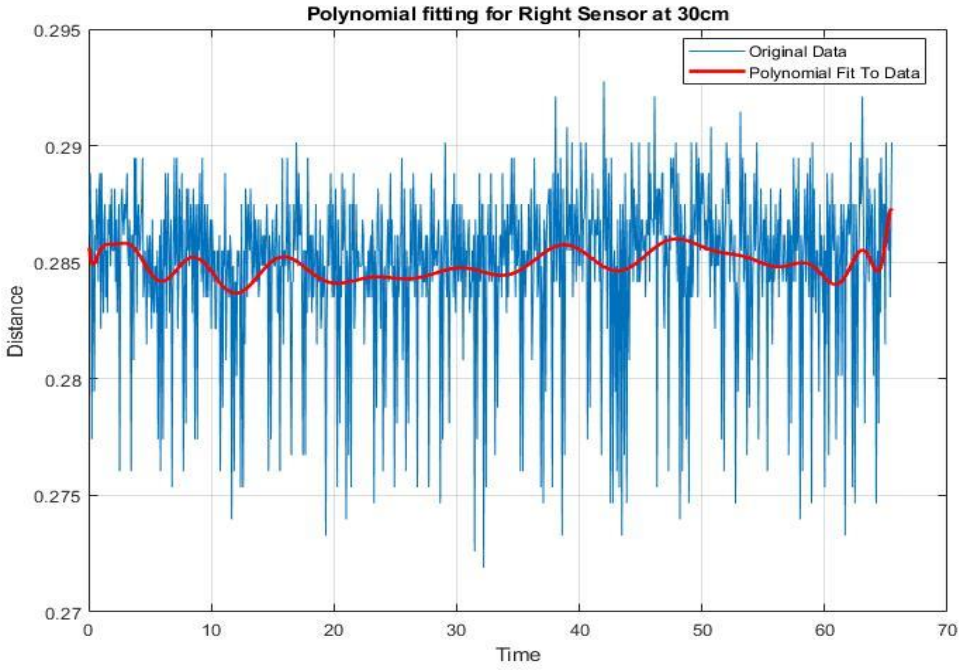
Για τα 120 cm:



Εικόνα 68: Αριστερός αισθητήρας στα 120 cm

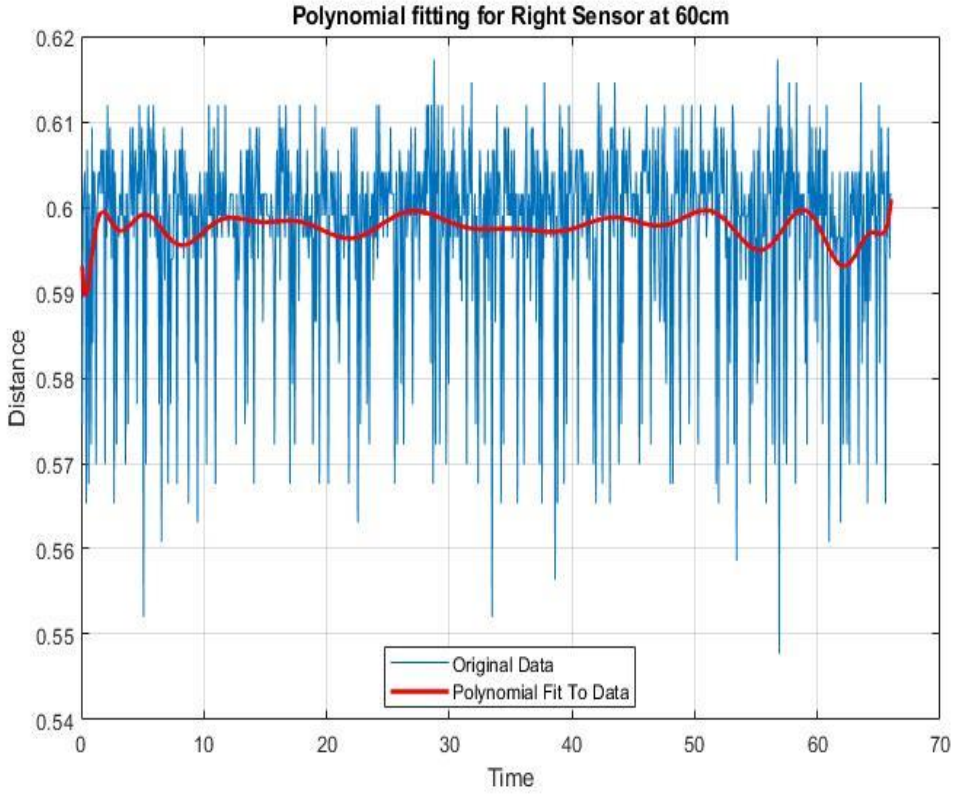
Δεξιός αισθητήρας

Για τα 30 cm:



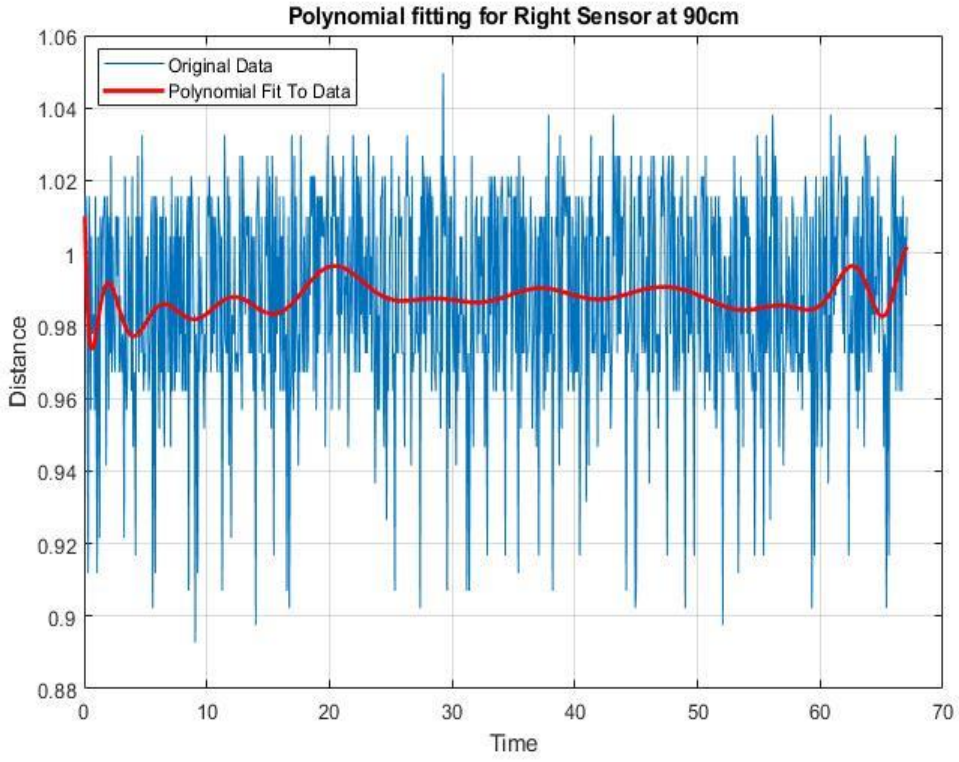
Εικόνα 69: Δεξιός αισθητήρας στα 30 cm

Για τα 60 cm:



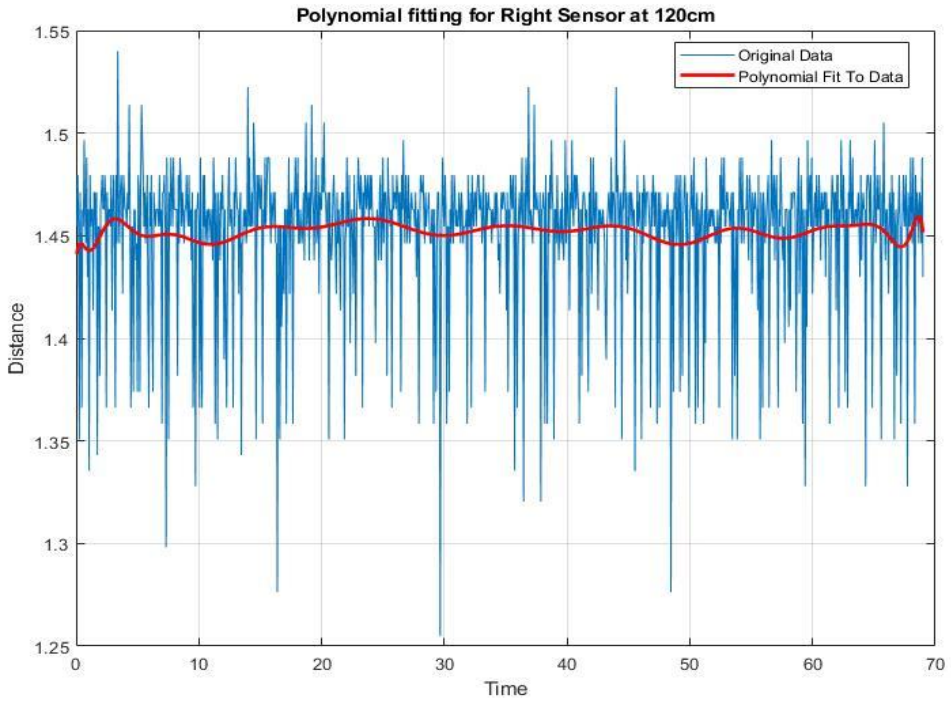
Εικόνα 70: Δεξιός αισθητήρας στα 60 cm

Για τα 90 cm:



Εικόνα 71: Δεξιός αισθητήρας στα 90 cm

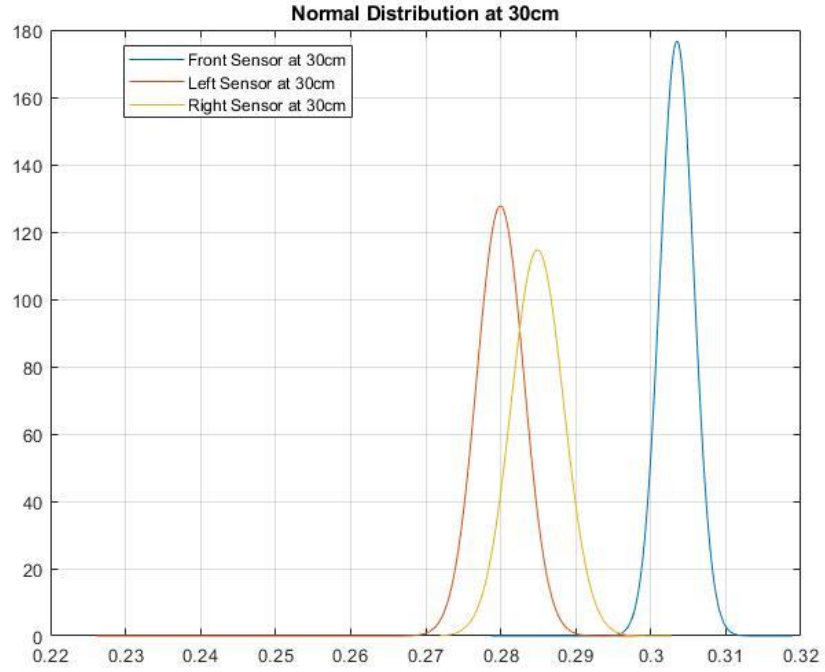
Για τα 120 cm:



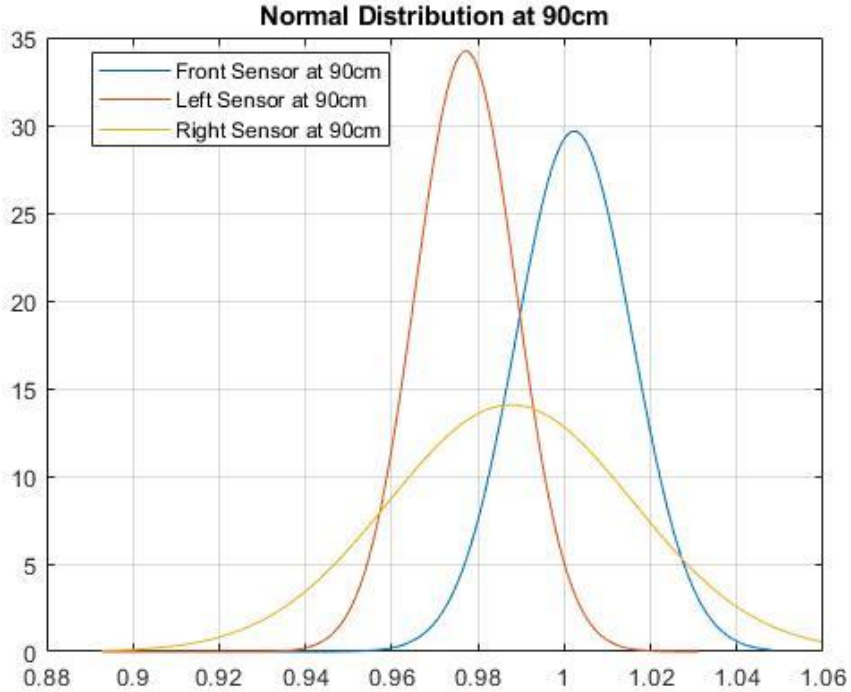
Εικόνα 72: Δεξιός αισθητήρας στα 120 cm

Στα plots παρατηρούμε ότι όσο πιο μεγάλη ήταν η απόσταση από το εμπόδιο τόσο περισσότερο υπάρχει απόκλιση στις μετρήσεις των αισθητήρων. Αυτό μπορεί να οφείλεται στα μοντέλα των αισθητήρων και το ηλεκτρονικό κύκλωμα σύνδεσης τους με το RPi, σύμφωνα με το διάγραμμα που φαίνεται στην εικόνα 25, στο 3ο κεφάλαιο στην ενότητα 3.2.3.

Στα επόμενα διαγράμματα βλέπουμε την κανονική κατανομή για τον μπροστινό, δεξιό και αριστερό αισθητήρα στα 30cm και στα 90cm.



Εικόνα 73: Κανονική κατανομή στα 30cm



Εικόνα 74: Κανονική κατανομή στα 90cm

7 Συμπεράσματα και Μελλοντική Εργασία

7.1 Συμπεράσματα

Τα συμπεράσματα από αυτήν την διπλωματική εργασία είναι η κατανόηση του θεωρητικού υπόβαθρου που υπάρχει πίσω από την διαφορική οδήγηση, για παράδειγμα οι εξισώσεις που έχουμε συμπεριλάβει στο 2ο κεφάλαιο στην ενότητα 2.3, βγαίνουν μετά από μεγάλη θεωρητική ανάλυση. Επίσης η εξοικείωση με το Gazebo ήταν σημαντική καθώς είναι ένα πολύ χρήσιμο εργαλείο για την προσομοίωση των ρομπότ.

Στην διπλωματική εργασία υπάρχουν πολλές διαφορές στο πραγματικό πείραμα σε σχέση με την προσομοίωση. Στο πειραματικό ρομπότ πρέπει να προβλέψουμε να δώσουμε μεγαλύτερα action ranges γιατί το ρομπότ είναι μεγαλύτερο και ογκώδες όπως και διαφορετική θέση των αισθητήρων και μεγαλύτερος χώρος πειραματισμού σε σχέση με την προσομοίωση. Επίσης πρέπει να προσέξουμε τι τιμές θα δώσουμε στις παραμέτρους turn_vel και turn_time καθώς και straight_vel γιατί στην προσομοίωση επειδή είναι ιδανικές οι συνθήκες θα έχουμε αμέσως επίδραση με ανεπιθύμητα αποτελέσματα όπως σύγκρουση με κάποιο εμπόδιο, ενώ στο πραγματικό ρομπότ λόγω βάρους ρομπότ, φθαρμένα λάστιχα χρειαζόμαστε λίγο μεγαλύτερες τιμές για να στρίβει πιο γρήγορα.

7.2 Μελλοντική Εργασία

Το πρακτικό κομμάτι αυτής της διπλωματικής εργασίας θα μπορούσε να αναπτυχθεί ή να βελτιωθεί περισσότερο. Για παράδειγμα, θα μπορούσε να αντικατασταθεί το RPi 4 με ένα Jetson Nano της Nvidia το οποίο παρέχει περισσότερες δυνατότητες. Θα μπορούσε επίσης να προστεθεί κάμερα για χαρτογράφηση της περιοχής. Ακόμα να προστεθούν και encoders στους τροχούς. Μια άλλη βελτίωση είναι να αναπτυχθεί περισσότερο η handmade πλακέτα με ένα PCB, ώστε να ελαχιστοποιήσουμε θορύβους, να μειωθεί ο χώρος που καταλαμβάνει πάνω στο ρομπότ και να είναι πιο ασφαλής από θέμα σταθερότητας, για παράδειγμα μην κοπεί κάποιο καλώδιο.

Βιβλιογραφία και αναφορές

[1] Roland Siegwart, Illah R. Nourbakhsh, Davide Scaramuzza “Introduction to Autonomous Mobile Robots”, 2004, The MIT Press, pp. 35, 36

[2] Δρ. Τζαφέστας Κωνσταντίνος, Παρουσίαση: «Ρομποτική ΙΙ, Ευφυή και Επιδέξια Ρομποτικά Συστήματα», pp.12

[3] Margarita Chli, Marco Hutter, Martin Rufli, Davide Scaramuzza, Roland Siegwart, Presentation: “Autonomous Mobile Robots”, Autonomous Systems Lab, ETH Zurich

[4] Roland Siegwart, Illah R. Nourbakhsh, Davide Scaramuzza “Introduction to Autonomous Mobile Robots”, 2004, The MIT Press, pp. 61

[5] Papadopoulos E. and Misailidis Michael, “On Differential Drive Robot Odometry with Application to Path Planning”, European Control Conference 2007, ECC 2007, ISBN: 978-960-89028-5-5

[6] Eren A., Dogan H., “Design and implementation of a cost effective vacuum cleaner robot”, Turkish Journal of Engineerin,2022, DOI: 10.31127/tuje.830282

[7] Lentin Joseph “Robot Operating System (ROS) for Absolute Beginners: Robotics Programming Made Easy”, Second Edition, Apress, 2022, Chapter 6, pp.255-260, https://doi.org/10.1007/978-1-4842-7750-8

[8] Morgan Quigley, Brian Gerkey, William D. Smart “Programming Robots with ROS: A Practical Introduction to the robot operating system”, 2015, O’Reilly, Chapter 16, pp.281-296

[9] Parag H. Batavia, Illah Nourbakhsh, “Path Planning for the Cye Personal Robot”, Submitted to the IEEE International Conference on Robots and Systems, 2000

[10] Khepera Robot, LAMI laboratory of Professor Jean-Daniel Nicoud at EPFL, 1990

[11] Caprari, G., Siegwart, R., “Design and Control of the Mobile Micro Robot Alice”, 2003

[12] https://tutorials-raspberrypi.com/infrared-distance-measurement-with-the-raspberry-pi-sharp-gp2y0a02yk0f/

[13] https://ramith.fyi/setting-up-raspberry-pi-4-with-ubuntu-20-04-ros-intel-realsense/

[14] https://roboticsbackend.com/install-ubuntu-on-raspberry-pi-without-monitor/

[15] http://wiki.ros.org/noetic/Installation/Ubuntu

[16] https://github.com/EmilGus/install_raspi-config/

[17] https://github.com/OOyindamola/lsm9ds1_ros_python

[18] https://abyz.me.uk/rpi/pigpio/index.html