ΕΛΛΗΝΙΚΟ ΜΕΣΟΓΕΙΑΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ

Τμήμα Μηχανικών Πληροφορικής Τ.Ε

Πρόγραμμα Σπουδών: Μηχανικών Πληροφορικής

Πτυχιακή Εργασία

Τίτλος: ΑΝΑΠΤΥΞΗ ΑΥΤΟΝΟΜΟΥ ΡΟΜΠΟΤΙΚΟΥ ΟΧΗΜΑΤΟΣ ΕΔΑΦΟΥΣ ΕΛΕΓΧΟΜΕΝΟΥ ΜΕΣΩ ROS

Ντάγκινης Νικόλαος (ΑΜ: ΤΠ 4012)

Επιβλέπων Εκπαιδευτικός : Σπυρίδων Παναγιωτάκης.

Επιτροπή Αξιολόγησης :

Ημερομηνία Παρουσίασης:

Ευχαριστίες

Θα ήθελα να ευχαριστήσω τον καθηγητή μου, κύριο Σπυρίδων Παναγιωτάκη και τον κύριο Γεώργιο Αλεξάκη, για την απεριόριστη βοήθεια και καθοδήγηση όποτε τη χρειάστηκα.

Abstract

The purpose of this thesis is to study, design and develop an autonomous robotic ground vehicle controlled through ROS, with the lowest possible implementation cost. It was studied to what extent this implementation is reliable both in terms of functionality of the vehicle and in terms of achieving autonomous navigation in an unknown environment.

More specifically, ROS (Robotic Operating System), an open-source operating system for robots, was used, which is based on the programming languages C++, Python and Lisp and runs on Unix-type systems and mainly on Ubuntu Linux. Also, XML, WiFi technologies and Arduino IDE software were used for the system requirements.

The study evaluated the capability of the vehicle for mapping the environment, the moving functionality of the vehicle and finally its capability for autonomous navigation. Mapping was deemed unreliable due to material limitations; evaluation of basic vehicle movements is considered satisfactory while autonomous navigation based on existing limitations is deemed reliable.

Finally, it is concluded that the partial upgrade of the equipment is necessary for the full functionality of the system.

Σύνοψη

Ο Σκοπός της παρούσας πτυχιακής εργασίας είναι η μελέτη, σχεδίαση και ανάπτυξη ενός αυτόνομου ρομποτικού οχήματος εδάφους ελεγχόμενου μέσω ROS, με το χαμηλότερο δυνατό κόστος υλοποίησης. Μελετήθηκε κατά το πόσο αυτή η υλοποίηση είναι αξιόπιστη τόσο ως προς τη λειτουργικότητα του οχήματος, όσο και ως προς την επίτευξη της αυτόνομης πλοήγησης σε άγνωστο περιβάλλον.

Πιο συγκεκριμένα, χρησιμοποιήθηκε ένα ανοιχτού κώδικα λειτουργικό για ρομπότ (Robot Opetating System ή ROS), το οποίο βασίζεται στις γλώσσες προγραμματισμού C++, Python και Lisp και εκτελείται σε συστήματα τύπου Unix και κυρίως στο Ubuntu Linux. Επίσης, για τις απαιτήσεις του συστήματος χρησιμοποιήθηκαν οι τεχνολογίες XML και WiFi και το λογισμικό Arduino IDE.

Η μέλετη αξιολόγησε τη χαρτογράφηση του περιβάλλοντος, τη λειτουργικότητα του όχηματος και τέλος την αυτόνομη πλοήγηση. Η χαρτογράφηση κρίθηκε αναξιόπιστη, λόγω των περιορισμών των υλικών, η αξιολόγηση των βασικών κινήσεων του οχήματος θεωρείται ικανοποιητική ενώ η αυτόνομη πλοήγηση με βάση των περιορισμών που υπάρχουν κρίνεται ως αξιόπιστη.

Τέλος, συμπαιραίνεται, πως η μερική αναβάθμιση του εξοπλισμού είναι απαραίτητη για την πλήρη λειτουργικότητα του συστήματος.

ΠΙΝΑΚΑΣ ΠΕΡΙΕΧΟΜΕΝΩΝ

1. Εισαγωγή

1.1. Αυτόνομο ρομποτικό όχημα

Ένα αυτόνομο ρομποτικό όχημα είναι ένα ρομπότ, το οποίο έχει σχεδιαστεί και κατασκευαστεί για να εκτελεί καθήκοντα, και να λειτουργεί σε ένα περιβάλλον ανεξάρτητα, χωρίς ανθρώπινο έλεγχο ή παρέμβαση. Έτσι, στο ρομπότ μπορεί να δοθούν καθήκοντα τα οποία μπορεί να είναι βαρετά ή επικίνδυνα για έναν άνθρωπο [1].

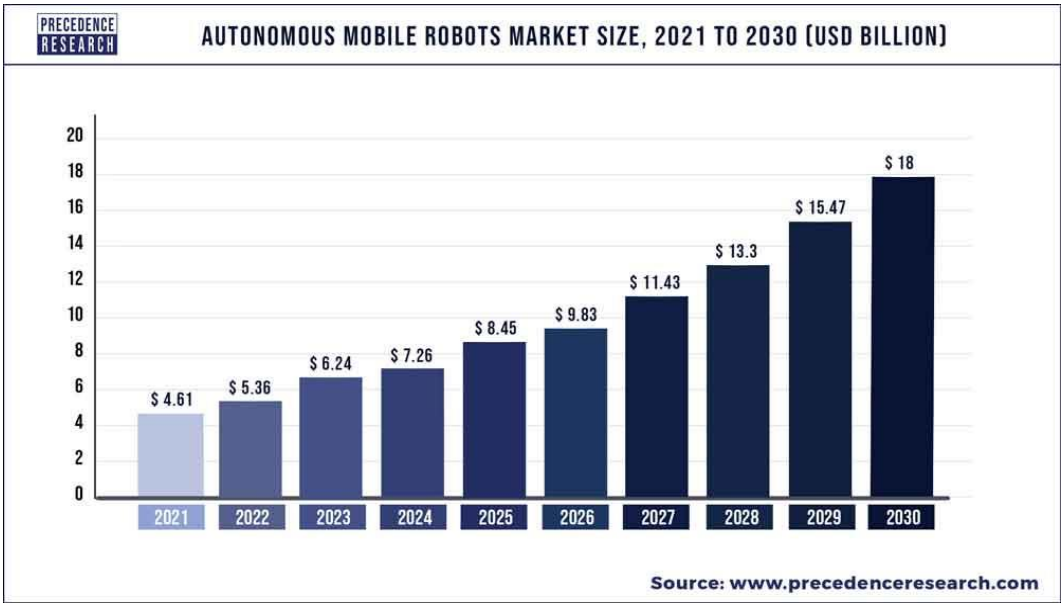
Για να θεωρηθεί ένα ρομπότ αυτόνομο χρειάζεται να ικανοποιεί κάποια κριτήρια. Το πρώτο κριτήριο για τη πλήρη αυτονομία, είναι η ικανότητα να μπορεί το ρομπότ να φροντίζει τον εαυτό του. Δηλαδή, να μπορεί να γνωρίζει πότε έχει χαμηλή μπαταρία και να ψάχνει το σταθμό φόρτισης, ή να “αισθάνεται” αν υπερθερμαίνεται.

Δεύτερο κριτήριο, είναι να μπορεί να έχει εξωαντίληψη, δηλαδή να γνωρίζει τι συμβαίνει γύρω του, στο περιβαλλον στο οποίο είναι. Τα αυτόνομα ρομπότ πρέπει να έχουν μια γκάμα από αισθητήρες, για να μπορούν να εκπληρώνουν τα καθήκοντα τους χωρίς κανένα πρόβλημα. Οι αισθητήρες θερμοκρασίας, υψόμετρου, απόστασης και άλλα είναι παραδείγματα τέτοιων αισθητήρων.

Τρίτο κριτήριο για αυτόνομη συμπεριφορά είναι, η ικανότητα να εκτελεί σωματικές εργασίες. Ένα παράδειγμα είναι οι σκούπες ρομπότ που πλέον έχουν κατακλύσει τις οικίες. Αυτά τα ρομπότ μπορούν να πλοηγηθούν σε μεγάλες περιοχές, όπως επίσης σε δύσκολες καταστάσεις μέσα στο σπίτι, χρησιμοποιώντας αισθητήρες επαφής ή μη επαφής.

Τέταρτο κριτήριο, που είναι και το θέμα της πτυχιακής, είναι η αυτόνομη πλοήγηση. Για ένα ρομπότ το να μπορεί να συσχετίσει συμπεριφορές με ένα μέρος,απαιτεί να γνωρίζει το που είναι και να μπορει να πλοηγηθεί από σημείο σε σημείο. Η αυτόνομη πλοήγηση χωρίζεται σε πλοήγηση σε εσωτερικούς χώρους και σε εξωτερικούς. Η εξωτερική αυτονομία επιτυγχάνεται πιο εύκολα στον αέρα, διότι τα εμπόδια είναι σπάνια. Οι πύραυλοι Κρουζ, είναι μεν αυτόνομα ρομπότ, δε πολύ επικίνδυνοι. Επίσης, τα μη επανδρωμένα αεροχήματα που χρησιμοποιούνται για αναγνωριστική έρευνα, είναι κάποια παραδείγματα. Η εξωτερική αυτονομία για οχήματα εδάφους είναι πολύ δύσκολη, λόγω του τρισδιάστατου εδάφους, των μεγάλων διαφορών στην επιφανειακή πυκνότητα, των απαιτήσεων του καιρού και στην αστάθεια του ασθητού περιβάλλοντος [2].

Για να ικανοποιήσει αυτά τα κριτήρια το εκάστοτε όχημα, χρείαζεται με κάποιον τρόπο να έχει την δυνατότητα να αποφασίζει μόνο του για όλα αυτά. Τη λύση σε αυτό το πρόβλημα έρχεται να τη δώσει το ROS. Το Robot Operating System, είναι ένα λογισμικό που βοηθάει το όχημα να ικανοποιήσει όλα αυτά τα κριτήρια. Διαθέτει διάφορα εργαλεία με τα οποία, μπορεί να διαβάσει και να υπολογίσει, όλες αυτές τις απαραίτητες τιμές που έρχονται από τους αισθητήρες και να πάρει τις κατάλληλες αποφάσεις για τον εαυτό του.



Εικόνα 1. Μέγεθος αγοράς αυτόνομων κινητών ρομπότ, 2021 μέχρι 2030 (Σε δισεκατομμύρια δολάρια)

Το παγκόσμιο μέγεθος αγοράς των αυτόνομων κινητών ρομπότ, αντιστοιχεί στα 5.36 δισεκατομμύρια δολάρια για το 2022, και προβλέπεται να χτυπήσει γύρω στα 18 δισεκατομμύρια δολάρια μέχρι το 2030. Αυτή η έκρηξη κερδών οφείλεται στην ταχύτατη επέκταση του ηλεκτρονικού εμπορίου τα τελευταία χρόνια. Για παράδειγμα, ένα μεγάλο Αμερικανικό κατάστημα του οποίου οι ηλεκτρονικές συναλλαγές εκτοξεύθηκαν 300% κατά τη διάρκεια της πανδημίας. Λόγω του μεγάλου φόρτου εργασίας, χρησιμοποιεί αυτόνομα κινητά ρομπότ για να παίρνουν τα εμπορεύματα και να τα ετοιμάζουν για παραλαβή [3].

1.2. Στόχος πτυχιακής εργασίας

Ο στόχος της πτυχιακής εργασίας είναι να κατασκευαστεί, ένα αυτόνομο ρομποτικό όχημα εδάφους που θα χρησιμοποιεί το Robot Operating System (ROS), και τα εργαλεία που το συνοδεύουν, για αναγνώριση και χαρτογράφηση αγνώστου περιβάλλοντος.

1.3. Δομή πτυχιακής εργασίας

Στο παρακάτω υποκεφάλαιο, θα αναφερθούμε εν συντομία στα κεφάλαια τα οποία θα αναλύσουμε στη συνέχεια.

Αρχής γενομένης απο το κεφάλαιο 2, θα μιλήσουμε για τη πλοήγηση, το ROS, την οδομετρία, την IMU και συναφείς υλοποιήσεις.

Στο κεφάλαιο 3, θα παρουσιάσουμε την ανάλυση των απαιτήσεων της πτυχιακής και πως αυτές οι απαιτήσεις με οδήγησαν στη κατάλληλη επιλογή λογισμικού και εξαρτημάτων.

Στο κεφάλαιο 4, θα δούμε αναλυτικά τα υλικά που χρησιμοποιήθηκαν για την υλοποίηση του οχήματος, όπως είναι ο αισθητήρας υπερήχων, η MPU 6050, οι Optical encoders και το λογισμικό το οποίο χρησιμοποιήθηκε και αναπτύχθηκε.

Στο κεφάλαιο 5, θα παρουσιάσουμε τη χρήση του οχήματος, το πως όλα αυτά τα εξαρτήματα συνδυάζονται με το λογισμικό.

Στο κεφάλαιο 6, θα παρουσιαστεί η χρήση του οχήματος, δηλαδή το πως όλα αυτά τα εξαρτήματα συνδυάζονται με το λογισμικό.

Στο κεφάλαιο 7, θα αξιολογήσουμε τη συμπεριφορά του οχήματος, δηλαδή το πόσο καλά αποδίδει βάσει κάποιων πειραμάτων.

Στο κεφάλαιο 8, θα μιλήσουμε για τα συμπεράσματα που προέκυψαν κατά τη διαδικασία συναρμολόγησης και δοκιμής του οχήματος, και για τις πιθανές μελλοντικές επεκτάσεις που θα μπορούσαν να βελτιώσουν το υπάρχον όχημα.

2. Σχετικές τεχνολογίες

2.1. Robot Operating System (ROS)

Το ROS είναι μια σουίτα ανοιχτού κώδικα ενδιάμεσου λογισμικού. Αν και το ROS δεν είναι λειτουργικό σύστημα, αλλά ένα σύνολο πλαισίων λογισμικού για την ανάπτυξη λογισμικού ρομπότ. Παρέχει υπηρεσίες για ένα ετερογενές σύμπλεγμα υπολογιστών, όπως η αφαίρεση υλικού, έλεγχος συσκευής χαμηλού ελέγχου, ανταλλαγή μηνυμάτων μέσω διαδικασιών, και διαχείρηση πακέτων. Εκτελώντας σύνολα διαδικασιών βασισμένα σε ROS, αναπαριστώνται σε αρχιτεκτονική γραφήματος, όπου η επεξεργασία λαμβάνει χώρα σε κόμβους οι οποίοι λαμβάνουν, στέλνουν και συνθέτουν δεδομένα αισθητήρων, ελέγχου, κατάστασης, σχεδιασμού, ενεργοποιητών και άλλων μηνυμάτων. Ανεξάρτητα απο τη σπουδαιότητα της αντιδραστικότητας και της χαμηλής καθυστέρησης μεταφοράς στον έλεγχο του ρομπότ, το ROS δεν είναι ένα πραγματικού χρόνου λειτουργικό σύστημα. Εν τούτοις, είναι πιθανόν να ενσωματώσουμε στο ROS πραγματικού χρόνου υπολογιστικό κώδικα.

Το λογισμικό στο οικοσύστημα του ROS μπορεί να διαχωριστεί σε τρείς ομάδες:

- Εργαλεία που είναι ανεξάρτητα από τη γλώσσα και τη πλατφόρμα, που χρησιμοποιούνται για κατασκευή και διανομή λογισμικού βασισμένο σε ROS.
- Υλοποιήσεις βιβλιοθηκών προγραμμάτων-πελατών σε ROS, όπως είναι η roscpp,rospy,και η roslisp.
- Πακέτα που περιέχουν εφαρμογές σχετικές με κώδικα, οι οποίες χρησιμοποιούν μια ή περισσότερες βιβλιοθήκες προγραμμάτων-πελατών σε ROS.

Το ROS χρησιμοποιεί C++, Python και Lisp για γλώσσες προγραμματισμού.

Οι κυρίως βιβλιοθήκες του ROS προσανατολίζονται προς ένα σύστημα που μοιάζει με unix, λόγω της εξάρτησής τους από μεγάλα σύνολα εξαρτήσεων που αποτελούνται από λογισμικό ανοιχτού κώδικα. Για αυτές τις βιβλιοθήκες, το Ubuntu Linux αναφέρεται ως “υποστηριζόμενο”, καθώς άλλες παραλλαγές όπως είναι το Fedora Linux, το macOS και το Microsoft Windows ορίζονται ως “πειραματικά” και υποστηρίζονται από την κοινότητα [4].

2.1.1. Σχεδιασμός Robot Operating System (ROS)

Το ROS σχεδιάστηκε για να είναι λογισμικό ανοιχτού κώδικα, σκοπεύοντας ότι οι χρήστες θα ήταν ικανοί να διαλέξουν τη διαμόρφωση εργαλίων και βιβλιοθηκών, τα οποία θα αλληλεπιδρούσαν με τον πυρήνα του ROS, έτσι ώστε οι χρήστες θα μπορούσαν να αλλάξουν τις στοίβες λογισμικού τους, για να ταιριάξουν στα ρομπότ τους και τη περιοχή εφαρμογή τους. Ως εκ τούτου, πέρα από τη γενική δομή μέσα στην οποία τα προγράμματα πρέπει να υπάρχουν και να επικοινωνούν, είναι πολύ λίγα τα οποία είναι ο πυρήνας του ROS.

Οι διαδικασίες του ROS παρουσιάζονται ως κόμβοι σε μια δομή γραφήματος, που συνδέονται από άκρες που ονομάζονται θέματα. Οι κόμβοι του ROS μπορούν να μεταφέρουν αναμεταξύ τους μηνύματα μέσω των θεμάτων, να κάνουν κλήσεις υπηρεσιών σε άλλους κόμβους, να παρέχουν μια υπηρεσία σε άλλους κόμβους, ή να λάβουν μοιρασμένα δεδομένα από μια κοινόχρηση βάση δεδομένων, η οποία ονομάζεται διακομιστής παραμέτρων. Η διαδικασία που ονομάζεται ROS Master τα κάνει όλα αυτά να είναι δυνατά με το να εγγράφει τους κόμβους στον εαυτό της, να εγκαθιστά από κόμβο σε κόμβο επικοινωνία για τα θέματα, και να ελέγχει τις ενημερώσεις για τον διακομιστή παραμέτρων. Τα μηνύματα και οι κλήσεις

υπηρεσιών δεν περνούν δια μέσου του master, απλά ο master στήνει διομότιμη επικοινωνία μεταξύ όλων των διαδικασίων κόμβου, αφού αυτοί εγγραφούν μαζί με τον master. Αυτή η αποκεντρωμένη αρχιτεκτονική είναι κατάλληλη για τα ρομπότ, τα οποία συχνά αποτελούνται από ένα υποσύνολο δικτυωμένων υλικών υπολογιστή, και μπορούν να επικοινωνούν με υπολογιστές για βαρείς υπολογισμούς ή εντολές.

Ένας κόμβος αντιπροσωπεύει μια διαδικασία η οποία τρέχει στο γράφημα ROS. Κάθε κόμβος πρέπει να έχει ένα όνομα, το οποίο καταχωρείται στο ROS master πριν κάνει οποιαδήποτε ενέργεια. Πολλαπλοί κόμβοι με διαφορετικά ονομάτα μπορούν να υπάρχουν κάτω απο διαφορετικούς χώρους ονομάτων, ή ένας κόμβος μπορεί να οριστεί ως ανώνυμος, που στη περίπτωση αυτή θα δημιουργηθεί τυχαία ένα επιπλέον αναγνωριστικό που θα προσθεθεί στο υπάρχουν όνομά του. Οι κόμβοι είναι στο κέντρο του προγραμματισμού του ROS, αφού ο περισσότερος κώδικας είναι σε μορφή κόμβου ROS οποίος αναλαμβάνει δράση βασισμένος στις πληροφοριες που λαμβάνει από τους άλλους κόμβους, στέλνει πληροφορίες σε άλλους κόμβους, ή στέλνει και λαμβάνει αιτήματα για δράσεις από και σε άλλους κόμβους.

Τα θέματα τα οποία ονομάζονται και ως δίαυλοι, μέσα από τα οποία οι κόμβοι στέλνουν και λαμβάνουν μηνύματα. Τα ονόματα των θεμάτων πρέπει να είναι μοναδικά όπως επίσης και στον χώρο ονομάτων τους. Για να σταλεί ένα μήνυμα σε ένα θέμα, ένας κόμβος πρέπει να δημοσιεύει στο εν λόγω θέμα,ενώ για να δέχεται μηνύματα πρέπει να εγγραφεί. Το μοντέλο δημοσίευσης/εγγραφής είναι ανώνυμο, κανένας κόμβος δεν ξέρει ποιος κόμβος στέλνει ή λαμβάνει σε ένα θέμα, μόνο ότι στέλνει και λαμβάνει σε αυτό το θέμα. Οι τύποι των μηνυμάτων που περνούν σε ένα θέμα ποικύλλουν ευρέως και μπορούν να οριστούν από τον χρήστη. Τα περιεχόμενα αυτών των μηνυμάτων μπορεί να είναι δεδομένα αισθητήρων, εντολές ελέγχου μοτέρ, ή στέλνει και λαμβάνει αιτήματα για δράσεις σε και από άλλους κόμβους.

Ο διακομιστής παραμετροποίησης είναι μια μοιρασμένη βάση δεδομένων μεταξύ των κόμβων που του επιτρέπει κοινόχρηση πρόσβαση σε σταθερές ή ημισταθερές πληροφορίες. Δεδομένα δηλαδή, τα οποία δεν αλλάγουν συχνά, και τα οποία θα είναι σπάνια προσβάσιμα, όπως είναι η αποστάση μεταξύ 2 σημείων σε ένα περιβάλλον, ή το βάρος του ρομπότ [4].

2.1.2. Εργαλεία του Robot Operating System (ROS)

Η λειτουργικότητα του πυρήνα του ROS είναι επαυξημένη με μια ποικιλία από εργαλεία,τα οποία επιτρέπουν στους προγραμματιστές να απεικονίσουν και να καταγράψουν δεδομένα, να πλοηγηθούν εύκολα στις δομές των πακέτων του ROS, και να δημιουργήσουν κείμενα αυτοματοποίησης σύνθετων διαμορφώσεων και διαδικασίες εγκατάστασης. Η προσθήκη αυτών των εργαλείων αυξάνει πολύ τις ικανότητες των συστημάτων που χρησιμοποιούν το ROS, με το να απλοποιούν και να προσφέρουν λύσεις σε έναν αριθμό κοινών προβλήμάτων που παρουσιάζονται κατά τη διάρκεια της ανάπτυξης ενός ρομπότ. Τα εργαλεία αυτά παρέχονται σε πακέτα όπως σε κάθε άλλο αλγόριθμο.

Το **rosserial**, είναι ένα γενικό πρωτόκολλο για την αποστολή μηνυμάτων ROS μέσω σειριακών συνδέσμων, κυρίως για την ενσωμάτωση μικροελεγκτών χαμηλού κόστους (Arduino) σε ROS.

Το **rviz** είναι ένα εργαλείο τρισδιάστατης απεικόνισης, που χρησιμοποιείται για να απεικονίσει ρομπότ, τα περιβάλλοντα στα οποία εργάζονται, και δεδομένα αισθητήρων.Είναι ένα εργαλείο υψηλής παραμετροποίησης, με πολλούν διαφορετικούς τύπους απεικονίσεων και πρόσθετων.

Το **rosbag** είναι ένα εργαλείο της γραμμής εντολών που χρησιμοποιείται για εγγραφή και αναπαραγωγή δεδομένων μηνυμάτων ROS. Το rosbag χρησιμοποιεί μια μορφή αρχείου που ονομάζεται bags, η οποία καταγράφει μηνύματα ROS με το να ακούει θέματα και να καταγράφει μηνύματα καθώς μπαίνουν. Η αναπαραγωγή των μηνυμάτων μέσα από ένα αρχείο bag είναι σε μεγάλο βαθμό το ίδιο, με το να έχουμε τους γνήσιους κόμβους, από τους οποίους παράγονται τα δεδομένα μέσα στο γράφημα επεξεργασίας του ROS, κατατάσοντας τα αρχεία bags σε ένα χρήσιμο εργαλείο για καταγραφή δεδομένων, για να χρησιμοποιηθούν σε μετεγενέστερη ανάπτυξη.

Το **catkin** είναι ένα σύστημα κατασκευής ROS, το οποίο αντικατέστησε το rosbuild, από την έκδοση ROS Groovy. Το catkin είναι βασισμένο στο CMake.

Το **roslaunch** είναι ένα εργαλείο το οποίο χρησιμοποιείται για να ξεκινήσουν πολλαπλοί κόμβοι ROS, τόσο τοπικά, όσο και εξ'αποστάσεως, όπως επίσης και να ρυθμίζει παραμέτρους στον διακομιστή παραμετροποίησης του ROS. Το roslaunch χρησιμοποιεί αρχεία παραμετροποίησης, τα οποία είναι γραμμένα σε XML, μπορούν εύκολα να αυτοματοποιήσουν μια περίπλοκη διαδικασία εκκίνησης και διαμόρφωσης σε μια εντολή. Τα κείμενα roslaunch μπορούν να συμπεριλάβουν άλλα κείμενα roslaunch, κόμβους εκκίνησης σε συγκεκριμένα μηχανήματα, ακόμα και να επανεκκινήσουν διαδικασίες οι οποίες τερματίστηκαν κατά τη διάρκεια της εκτέλεσης [4].

2.1.3. Πακέτα του Robot Operating System (ROS)

Το λογισμικό στο ROS είναι οργανωμένο σε πακέτα. Ένα πακέτο μπορεί να περιέχει κόμβους ROS, μια ανεξάρτητη βιβλιοθήκη ROS, ένα σύνολο δεδομένων, αρχεία ρυθμίσεων, ένα τρίτο κομμάτι λογισμικού, ή οτιδήποτε άλλο που λογικά συνιστά μια χρήσιμη μονάδα. Ο στόχος αυτών των πακέτων είναι να προμηθεύσει αυτή τη χρήσιμη λειτουργικότητα, σε έναν εύχρηστο τρόπο έτσι ώστε το λογισμικό να μπορεί εύκολα να επαναχρησιμοποιηθεί. Σε γενικές γραμμές, τα πακέτα του ROS ακολοθούν την αρχή "Goldilocks": αρκετή λειτουργικότητα για να είναι χρήσιμα, αλλά όχι τόσο πολύ που τα πακέτα να είναι βαριά και δύσκολα να χρησιμοποιηθούν από άλλα λογισμικα [5]. Αξιοσημείωτα πακέτα είναι:

- **Slam toolbox**: παρέχει πλήρη ταυτόχρονη δισδιάστατη χαρτογράφηση και εντοπισμό.
- **Gmapping**: παρέχει ένα "περιτύλιγμα" για τον αλγόριθμο OpenSlam's Gmapping για ταυτόχρονη χαρτογράφηση και εντοπισμό.
- **Amcl**: παρέχει μια υλοποίηση από τον προσαρμοστικό εντοπισμό Μόντε Κάρλο.
- **Navigation**: παρέχει την ικανότητα για πλοήγηση ενός κινούμενου ρομπότ σε ένα επίπεδο περιβάλλον.
- **Tf**: παρέχει ένα σύστημα για την αναπαράσταση, μεταμόρφωση και εντοπισμού πλαισίων συντεταγμένων [4].

2.2. Οδομετρία

Είναι σημαντικό για ένα ρομπότ να ξέρει το που βρίσκεται. Οι άνθρωποι μπορούν να ξέρουν το που βρίσκονται, χρησιμοποιώντας την όραση τους, μακροχρόνιες μνήμες, και την επίγνωση του περιβάλλοντος. Όμως, τα ρομπότ δεν διαθέτουν αυτές τις εξελιγμένες ικανότητες, έτσι, η οδομετρία είναι μια τεχνική που χρησιμοποιούν τα ρομπότ για τοποθετούν τον εαυτό τους στο περιβάλλον στο οποίο βρίσκονται.

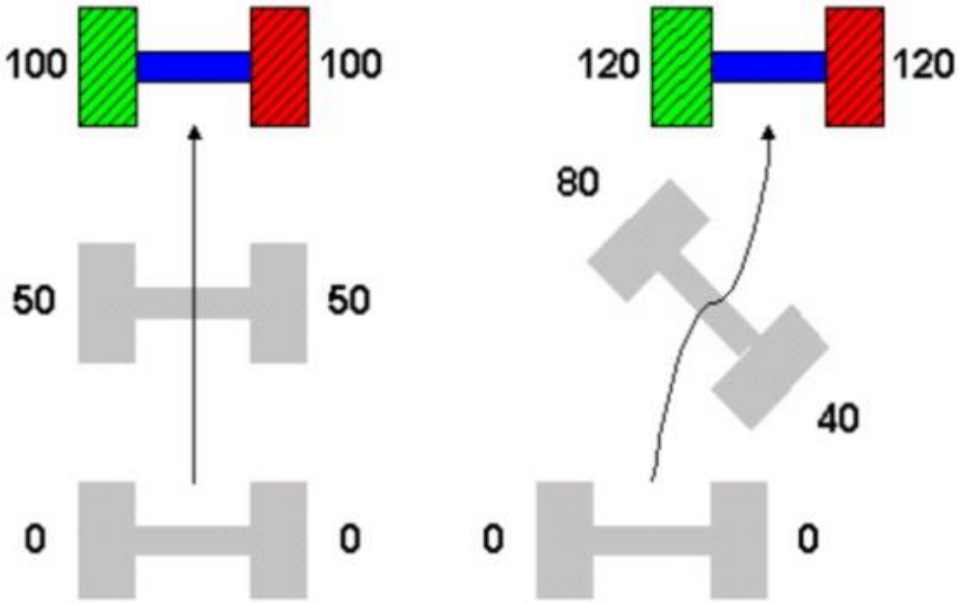
Η μέτρηση της αλλαγής θέσης ενός ρομπότ σε μια δεδομένη θέση χρησιμοποιώντας αισθητήρες κίνησης, είναι γνωστή ως οδομετρία. Για παράδειγμα, αν ένα ρομπότ κινείται ευθεία μπροστά και ξέρει τη διάμετρο των τροχών του, τότε μπορεί να καθορίσει το πόσο μακριά έχει πάει, με το να μετράει τον αριθμό των περιστροφών των τροχών του. Οι κινητήριοι τροχοί του ρομπότ είναι συνήθως εξοπλισμένοι με κωδικοποιητές άξονα, οι οποίοι παράγουν έναν προκαθορισμένο αριθμό παλμών για κάθε περιστροφή. Αυτοί οι παλμοί μετρούνται από τη κεντρική μονάδα επεξεργασίας, η οποία μπορεί να μετρήσει την απόσταση που διήνυσε.

Η οδομετρία έχει ορισμένα μειονεκτήματα παρά το γεγονός ότι είναι ένας αισθητήρας θέσης που χρησιμοποιείται συχνά από κινούμενα ρομπότ. Οποιαδήποτε ανακρίβεια ανίχνευσης, θα γίνεται όλο και χειρότερη με το χρόνο επειδή είναι αθροιστική μέτρηση. Για την αποφυγή της συσσώρευσης των λαθών, τα ρομπότ μπορούν περιστασιακά να χρειαστούν να απασχολήσουν, παραπάνω αισθητήρες για να προσδιορίσουν την ακριβή θέση τους.

Ένα απο τα βασικά προβλήματα σε εφαρμογές για κινούμενα ρομπότ είναι ο ακριβής εντοπισμός του οχήματος. Για να είναι δυνατή η αυτόνομη πλοήγηση, ένα ρομπότ πρέπει να ξέρει συνεχώς τη θέση του. Ως αποτέλεσμα, οι ερευνητές και οι μηχανικοί έχουν δημιουργήσει μια σειρά από διάφορες μεθόδους και συστήματα για τη τοποθεσία των κινούμενων ρομπότ, συμπεριλαμβανομένων της οδομετρίας τροχών, οδομετρίας λέιζερ,το GPS και άλλα[6].

Η ανακρίβεια της οδομετρίας μπορεί να προκληθεί από:

- Τη λανθασμένη μέτρηση της διαμέτρου του τροχού.
- Συστήματα πολλαπλών τροχών με ποικίλες διαμέτρους τροχών.
- Προβλήματα στη μέτρηση των παλμών κατά τη διάρκεια χρήσης των κωδικοποιητών άξονα.
- Η επεξεργασία της οδομετρίας είναι αργή.



Εικόνα 2. Παράδειγμα οδομετρίας.

Οι δύο πιθανές διαδρομές κίνησης ενός ρομπότ διαφορικής κίνησης παρουσιάζονται στη παραπάνω εικόνα. Οι τιμές των κωδικοποιητών βρίσκονται δίπλα σε κάθε τροχό. Στο αριστερό διάγραμμα, το ρομπότ κινείται σε ένα ευθύ μονοπάτι, εξασφαλίζοντας ότι οι τιμές του κωδικοποιητή είναι πανομοιότυποι. Το ρομπότ στα δεξιά όμως, ακολουθεί μια φιδίσια πορεία. Το ρομπότ κάνει μια δεξιά στροφή όταν ο πράσινος κωδικοποιητής δείχνει ογδόντα και ο κόκκινος σαράντα στα μισά του δρόμου. Οι κωδικοποιητές στη τελευταία θέση του ρομπότ, όμως, διαβάζουν την ίδια τιμή ακόμα και αν είναι μεγαλύτερη από τις τιμές στα αριστερά, από τη στιγμή που το δεξιά ρομπότ έκανε μεγαλύτερη απόσταση ακολουθώντας μια S-καμπύλη. Αυτό συμβαίνει επειδή το ρομπότ στη συνέχεια γύρισε πίσω προς τα αριστερά. Το πρόγραμμα ελέγχου θα νομίζει ότι το ρομπότ κινήθηκε σε ευθύ μονοπάτι, αν λάβει υπόψιν του τους κωδικοποιητές στη τελευταία θέση. Αυτός είναι ο λόγος γιατί τα λάθη μπορούν να προέλθουν από την αργή επεξεργασία του κωδικοποιητή. Η πληροφορία της οδομετρίας πρέπει να επεξεργάζεται όσο πιο γρήγορα γίνεται, εκτός αν το ρομπότ είναι μηχανικά αποδεδειγμένο ότι θα κινηθεί σε ευθύ μονοπάτι.

Εν κατακλείδι, η οδομετρία είναι η διαδικασία παρακολούθησης της απόλυτης θέσης ενός ρομπότ. Αλγόριθμοι κίνησης μπορούν να χρησιμοποιούν αυτή τη πληροφορία για να οδηγούν σε τοποθεσίες ή για να στρίβουν σε απόλυτες γωνίες [6].

2.3. Αυτόνομη πλοήγηση

Το αυτόνομο σύστημα πλοήγησης είναι ένα σύστημα, το οποίο μπορεί να σχεδιάσει τη πορεία του και να εκτελέσει το σχέδιο του, χωρίς την ανθρώπινη παρέμβαση. Είναι ένας συνδυασμός πολύπλοκων συστημάτων που βοηθά στη λήψη αποφάσεων με βάση τις περιβάλλουσες καταστάσεις. Εγκαθιστάται σε οχήματα όπως drones,ρομπότ, αυτοκίνητα, βάρκες και άλλα. Η τεχνολογία του συστήματος πλοήγησης εκτελείται με χρήση πλοηγμένων

τεχνολογιών, όπως το σύστημα αδρανειακής πλοήγησης, το σύστημα δορυφορικής πλοήγησης, με ραντάρ, με κάμερες, και με πλοηγήση υπερήχων, μαζί με αυτόνομους αλγόριθμους πλοήγησης για ακριβή και ασφαλή πλοήγηση οχημάτων [7].

Η κατανόηση των απαραίτητων βημάτων που επιτρέπουν στα ρομπότ και στις αυτοματοποιημένες συσκευές να αναλαμβάνουν πολλές βαρέτες, δύσκολες και επικίνδυνες εργασίες μπορεί να ρίξει λίγο φως στις προκλήσεις που υπάρχουν στην ανάπτυξη αυτόνομων συστημάτων πλοήγησης ρομπότ. Τρία κύρια στοίχεια αφορούν την ανάπτυξη της αυτόνομης ρομποτικής πλοήγησης:

- Καθοδήγηση.
- Πλοήγηση με χρήση GPS και άλλων συστηματων.
- Έλεγχος.

Αυτά τα στοιχεία δουλεύουν μεταξύ τους, για να επιτρέψουν τα αυτόνομα ρομποτικά συστήματα να αποδώσουν σωστά σε πραγματικές συνθήκες. Η καθοδήγηση καθορίζει την επιθυμητή τροχιά για το όχημα ή το ρομπότ. Αυτό περιλαμβάνει τη στόχευση, τις προσαρμογές στην ταχύτητα και την επιτάχυνση,και το στρίψιμο όπως απαιτείται για την επιτεύξη ενός καθορισμένου στόχου. Κατά τη διάρκεια της κίνησης του ρομπότ ή της αυτόνομης συσκευής, η πλοήγηση παρέχει πληροφορίες για την ακριβή ταχύτητα και την ταχύτητα του οχήματος προς μια κατεύθυνση. Οι μονάδες ελέγχου και το λογισμικό διαχειρίζονται τις δυνάμεις που απαιτούνται για την προώθηση των ρομπότ, και για τη διασφάλιση του μέγιστου βαθμού σταθερότητας όταν αυτά διασχίζουν διάφορους τύπους εδάφους.

Κάθε ένα από αυτά τα τρία στοιχεία είναι απαραίτητα για να επιτευχθούν τα επιθυμητά αποτελέσματα για την αυτονομία πλοήγησης σε πρακτικές εφαρμογές.

Ενώ πολλές πρακτικές λύσεις υλικού είναι διαθέσιμες για αυτόνομη πλοήγηση ρομπότ και για τις απαιτήσεις πλοήγησης οχημάτων, δεν υπάρχει ακόμη κάποιο πρότυπο για συστήματα λογισμικού. Αυτό έχει δημιουργήσει κάποια θέματα στο πεδίο της αυτόνομης πλοήγησης:

- Το φιλτράρισμα των δεδομένων είναι απαραίτητο για τον διαχωρισμό των σχετικών πληροφοριών από τα μεγάλα σύνολα δεδομένων, που συλλέγονται από το υλικό του αισθητήρα. Το φιλτράρισμα Κάλμαν, η οποία επίσης ονομάζεται γραμμική τετραγωνική εκτίμηση, είναι μια λύση για αυτό το πρόβλημα. Μερικές αστοχίες σε αυτή τη διαδικασία φιλτραρισμού δεδομένων υπάρχουν όταν χρησιμοποιείται αυτή η μέθοδος.
- Επί του παρόντος, δεν υπάρχει τυποποιημένη λύση plug and play που να καλύπτει τις βιομηχανίες. Αυτό σημαίνει ότι η αυτόνομη πλοήγηση για ιπτάμενα ρομπότ και διαστημόπλοια, χρησιμοποιεί διαφορετικά πρότυπα λογισμικού από την αυτόνομη πλοήγηση οχημάτων εδάφους, κάνοντας το δύσκολο να ενσωματωθούν οι συσκευές και τα ρομπότ αυτόνομης πλοήγησης σε ένα σύστημα.
- Ο οπτικός ταυτόχρονος εντοπισμός και η χαρτογράφηση είναι ακόμα στα πρώτα στάδια ανάπτυξης. Για να το θέσουμε απλά,είναι η διαδικασία η οποία επιτρέπει στα συστήματα αυτόνομης πλοήγησης να καθορίσουν τη στάση και τη θέση ενός αισθητήρα μέσα στο πλαίσιο του άμεσου περιβάλλοντός του,με το να χαρτογραφούν

το περιβάλλον γύρω από τον αισθητήρα. Όταν τα συστήματα GPS πλαστογραφούνται σκόπιμα από χάκερ ή δέχονται επίθεση DoS, αυτά τα συστήματα εντοπισμού θέσης μπορούν να παρέχουν ανακριβή δεδομένα σε αυτόνομες εφαρμογές ρομποτικής. Ο οπτικός ταυτόχρονος εντοπισμός και η χαρτογράφηση, μπορεί να παρέχει μια σταθερή θέση επιστροφής. Σε κάποια περιβάλλοντα, μπορεί να υπηρετήσει ως η βασική μέθοδος για την αυτόνομη πλοήγηση οχημάτων.

Η σωστή λύση για το λογισμικό αυτόνομου ρομπότ θα αντιμετωπίσει το φιλτράρισμα δεδομένων, την τυποποίηση και τις αρνήσεις GPS με πρακτικό τρόπο [8].

2.3.1 Αυτόνομη πλοήγηση και Robot Operating System (ROS)

Για να φτιάξουμε ένα ρομπότ αυτόνομης πλοήγησης στο ROS, πρέπει να κατανοήσουμε τα βασικά στοιχεία των διάφορων πτυχών. Το ρομπότ πρέπει να μοντελοποιηθεί με τέτοιο τρόπο,ώστε να μπορούν να επιτευχθούν οι μετασχηματισμού της στάσης του. Επίσης, το ρομπότ πρέπει να ακούει και να αντιδρά σε δεδομένα περιστροφής. Και τελικά, πρέπει συνεχώς να δημοσιεύει τα δεδομένα του αισθητήρα θέσης σε μια μορφή μηνυμάτων οδομετρίας.

Σύμφωνα με το εγχειρίδιο του ROS,οι ακόλουθες υποθέσεις και απαιτήσεις πρέπει να πληρούνται από το ρομπότ:

- Η βάση του έχει σχήμα περίπου τετράγωνο.
- Χρησιμοποιέι διαφορική κίνηση ή είναι ολονομικό τροχοφόρο.
- Ακούει στις εντολές μηνυμάτων περιστροφής και μεταφράζει τη **X** ταχύτητα, τη **Y** ταχύτητα και **Theta** ταχύτητα, σε συγκεκριμένες και σωστές κινήσεις.
- Λαμβάνει τα δεδομένα των αισθητήρων για το εξωτερικό του περιβάλλον σε μορφή LaserScan ή Pointcloud μηνυμάτων.

Λαμβάνοντας υπόψη αυτά τα σημεία, το τελευταίο σημείο είναι ένας μάλλον δραστικός περιορισμός: Άλλοι αισθητήρες, όπως στερεοσκοπικές κάμερες, σόναρ και υπέρυθρες δεν έχουν άμεση χρησιμότητα. Προτιμότερο, είναι η εγγενής μορφή δεδομένων τους να μεταφράζεται στα αντίστοιχα θέματα.

2.3.1.1. Απαιτήσεις δεδομένων

Δεδομένα οδομετρίας. Τα δεδομένα μετασχηματισμού αντιπροσωπεύουν τη στατική σχέση μεταξύ δύο αναφορών συντεταγμένων. Όταν το ρομπότ κινέιται, τα δεδομένα οδομετρίας, αντιπροσωπεύουν την δυναμική αλλαγή της ταχύτητας του. Στο ROS, το ρομπότ χρειάζεται να δημοσιέυει τόσο τα δεδομένα μετασχηματισμού όσο και τα δεδομένα της οδομετρίας. Αφηρημένα μιλώντας, τα δεδομένα οδομετρίας φιλτράρονται μέσω των δεδομένων μετασχηματισμού. Τα δεδομένα οδομετρίας δημοσιεύονται ως **Odometry.**

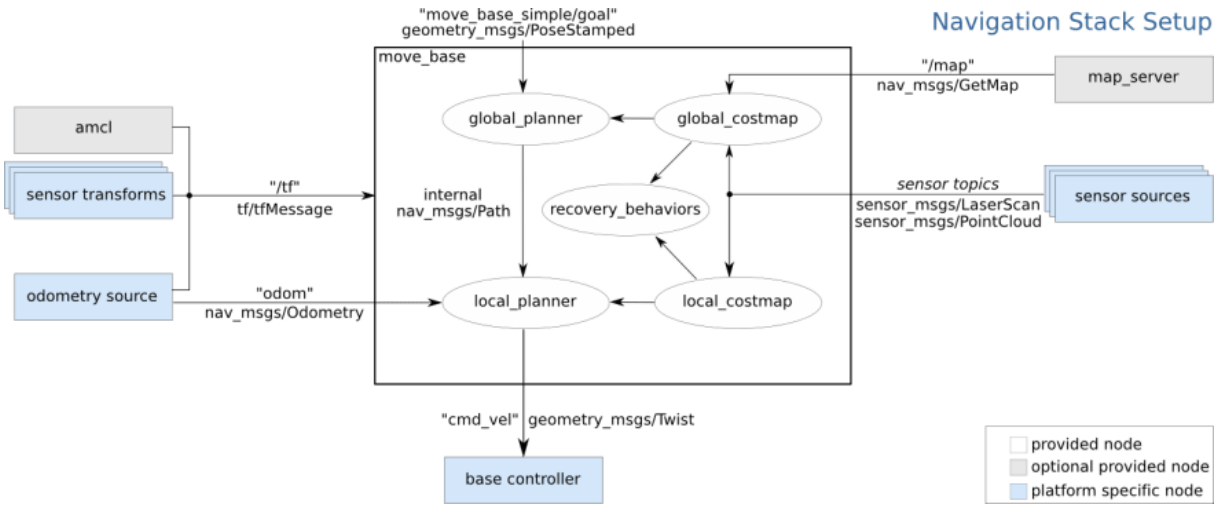
Δεδομένα αισθητήρα. Ο τελευταίος τύπος δεδομένων που απαιτείται για τη πλοήγηση, είναι τα δεδομένα αισθητήρα σε μορφή **Laserscan** ή **PointCloud.** Αυτά τα δεδομένα χρησιμοποιούνται για να μετρήσουμε τον περιβάλλοντα χώρο του ρομπότ. Μια σάρωση με λέιζερ αναπαριστά δισδιάστατα δεδομένα που έχουν ληφθεί σε οριζόντιο επίπεδο. Τα σημεία δεδομένων λαμβάνονται σε μια συγκεκριμένη γωνία, σε μια συγκεκριμένη απόσταση μεταξύ

μεμονωμένων σημείων και σε μια συγκεκριμένη χρονική σήμανση. Όλα αυτά περιέχονται σε ένα μήνυμα τύπου laserscan. Ένα pointcloud είναι μια τρισδιάστατη αναπαράσταση του περιβάλλοντός του. Αποτελείται από επιμέρους σημεία με **X,Y,Z** δεδομένα. Ο συνδυασμός όλων αυτών των σημείων, δίνει μια τρισδιάστατη εικόνα βάθους του περιβάλλοντος.

Δεδομένα μετασχηματισμού. Τα δεδομένα που εκφράζουν χωρικές σχέσεις σχετικά με το ρομπότ ή τον αισθητήρα, πρέπει να μεταφράζονται για να παρουσιαστούν τα πραγματικά δεδομένα. Σε ένα αφηρημένο επίπεδο, ένα κινητό ρομπότ αποτελείται από διαφορετικά δομικά στοιχεία, π.χ. το κέντρο του, τους τροχούς του, τους αισθητήρες που είναι τοποθετημένοι πάνω του και πολλά άλλα. Κάθε ένα από αυτά τα στοιχεία έχει ένα σημείο βάσης, μια απόλυτη αναφορά συντεταγμένων με τη μορφή **X**, **Y**, **Z**. Αν υποθέσουμε ότι ένας αισθητήρας είναι τοποθετημένος στη κορυφή του ρομπότ. Αυτός ο αισθητήρας εκτελεί σάρωση με λέιζερ του περιβάλλοντος χώρου του. Αυτές οι μετρήσεις γίνονται από το σημείο βάσης του αισθητήρα. Αλλά για να ενεργήσουμε σε αυτά, πρέπει να μετατρέψουμε τα δεδομένα του αισθητήρα ώστε να ταιριάζουν με τα σημεία βάσης του κέντρου των ρομπότ. Για αυτό, το ROS χρησιμοποιεί ένα δέντρο μετασχηματισμού, μια αναπαράσταση που καθορίζει τη μετατόπιση από τα κεντρικά σημεία βάσης του ρομπότ, στο σημείο βάσης άλλων τμημάτων του. Για παράδειγμα, η μετάφραση **x**=1,0, **y**=0,0, **z**=5,0 σημαίνει ότι ο αισθητήρας είναι τοποθετημένος 1,0 εκατοστό πιο μπροστά και 5,0 εκατοστά πάνω από το κεντρικό σημείο βάσης. Εάν έχουμε ένα πλήρες μοντέλο του ρομπότ, τότε αυτές οι σχέσεις υπολογίζονται αυτόματα από το ROS [9].

2.3.1.2 Στοιχεία στοίβας πλοήγησης

Στην στοίβα πλοήγησης του ROS, ο μετασχηματισμός, η οδομετρία και τα δεδομένα του αισθητήρα χρησιμοποιούνται για να προσδιοριστεί που βρίσκεται το ρομπότ, και δεδομένου ενός στόχου, που να μετακινηθεί. Για να επιτευχθεί αυτό, εμπλέκονται διάφορα στοιχεία λογισμικού της στοίβας πλοήγησης του ROS.



Εικόνα 3. Επιμέρους στοιχεία στοίβας πλοήγησης.

Σχεδιαστής πλοήγησης. κόμβοι σχεδιασμού είναι υπεύθυνοι να υπολογίζουν όλες τις πιθανές επιλογές διέλευσης για έναν δεδομένο στόχο. Θα δημιουργήσουν χάρτες

κόστους(**costmaps**) για την αναπαράσταση της προσπάθειας να ακολουθήσει το ρομπότ μια συγκεκριμένη διαδρομή ή μια ακολουθία βημάτων. Υπάρχουν δύο σχεδιαστές. Ο παγκόσμιος σχεδιαστής(**Global planner**) είναι στοιχεία υψηλού επιπέδου που κάνει τον μακροπρόθεσμο σχεδιασμό. Οι επιλογές διαμόρφωσής του περιλαμβάνουν, μέσα σε ποια απόσταση θα αποθηκευτούν τα εμπόδια που έχουν ανιχνευτεί στο χάρτη κόστους, εάν και πόσος χώρος μεταξύ ρομπότ και εμποδίου πρέπει να διατηρείται ελεύθερος και το βασικό αποτύπωμα ή ακτίνα του ρομπότ. Ο τοπικός σχεδιαστής αξιολογεί το επόμενο βήμα που πρέπει να κάνει το ρομπότ και πρέπει να διαμορφωθεί με τις βασικές κινητικές δυνατότητες του ρομπότ, την ελάχιστη/μέγιστη ταχύτητα και την επιτάχυνση του.

Ελεγκτής βάσης. Όταν ο σχεδιαστής αποφασίσει προς ποια κατεύθυνση θα ακολουθήσει, θα στείλει μηνύματα περιστροφής στο θέμα **cmd_vel.** Στη συνέχεια, ένας ελεγκτής βάσης του ρομπότ πρέπει να ακούσει αυτά τα μηνύματα και να εφαρμόσει τις εντολές κίνησης.

Διακομιστής χαρτών. Ο διακομιστής χαρτών είναι ένα προαιρετικό στοιχείο με το οποίο δημιουργείται μια αναπαράσταση του περιβάλλοντος. Ένας χάρτης μπορεί να καταγραφεί και να αποθηκευτεί και στη συνέχεια να χρησιμοποιηθεί με την επόμενη εκτέλεση του ρομπότ ή ακόμη και να εξαχθεί σε άλλο ρομπότ.

Αλγόριθμοι πλοήγησης. Η πλοήγηση του ROS βασίζεται στο SLAM, μια μέθοδο για τη παροχή ταυτόχρονου τοπικού εντοπισμού και χαρτογράφησης. Αυτό σημαίνει ότι το ρομπότ μπορεί να φτιάξει έναν χάρτη ενώ κινείται, και να χρησιμοποιεί αυτόν τον χάρτη για τον εντοπισμό της θέσης του ταυτόχρονα. Η στοίβα πλοήγησης απαιτεί τη διαθεσιμότητα δεδομένων σάρωσης λέιζερ ή pointcloud. Επομένως, οι υποστηριζόμενοι αλγόριθμοι πλοήγησης βασίζονται σε αυτές τις μορφές [9].

3. Σχεδιασμός Συστήματος

3.1. Σκοπός του συστήματος

Ο σκοπός του συστήματος που κατασκευάσαμε είναι η επιτυχής λειτουργία αυτόνομης πλοήγησης ενός ρομποτικού οχήματος εδάφους με τη βοήθεια του ROS. Το σύστημα θα αποτελείται τόσο από λογισμικό, στην προκειμένη περίπτωση το ROS, όσο και από υλικό, αισθητήρες, μικροελεγκτές και άλλα. Η υλοποίηση μας θα μπορεί να εκτελεί τις βασικές λειτουργίες της αυτόνομης πλοήγησης, και όσο μπορεί καλύτερα, βάση υλικών, τη χαρτογράφηση.

Όπως και κάθε άλλη παρόμοια υλοποίηση, έτσι και η δική μας χρειάζεται να μπορεί να παρακολουθεί τις διάφορες τιμές των αισθητήρων, να τις στέλνει στον υπολογιστή μέσω του ROS και αντιστρόφως, να δέχεται δεδομένα και εντολές, με σκοπό τη σωστή λειτουργία του συστήματος. Όμως η υλοποίηση μας αφόρα χαμηλού κόστους υλικά, και έτσι έχει ως αποτέλεσμα, κάποιους περιορισμούς σε διάφορες λειτουργίες.

Το σύστημα μας θα αποτελείται από ένα ESP32, ένα λαπτοπ όπου θα είναι εγκατεστημένο το ROS, κωδικοποιητές ταχύτητας,διάφορων ειδών αισθητήρες, και ένα σετ με ρόδες, σασί και ενεργοποιητές DC.

3.2. Τι προσπαθούμε να πετύχουμε

Οι συναφείς υλοποιήσεις των αυτόνομων ρομποτικών οχημάτων εδάφους συνήθως περιλαμβάνουν υψηλού κόστους αισθητήρες όπως Lidar, ή Imu, και έχουν εγκατεστημένο πάνω στο όχημα ένα υψηλού κόστους μικρουπολογιστή τύπου Nvidia Jetson, για να λειτουργήσει το ROS, οποίος συνδέεται σειριακά με ένα υψηλού κόστους Raspberry Pi.

Στην δική μας υλοποίηση όμως αντί για Lidar, θα χρησιμοποιήσουμε αισθητήρες απόστασης υπερήχων, αντί για το Nvdia Jetson, θα χρησιμοποιήσουμε ένα απλό λάπτοπ, το οποίο θα περιέχει το ROS και αντί για το Raspberry Pi, θα χρησιμοποιήσουμε ένα ESP32 το οποίο θα επικοινωνεί μέσω WIFI με το ROS.

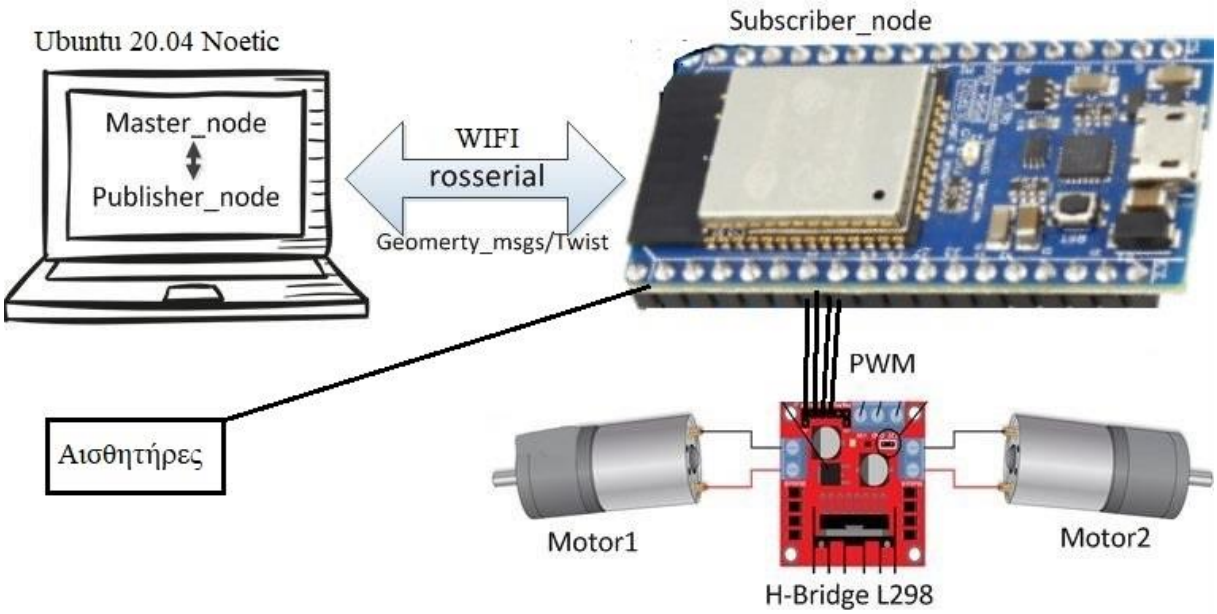
Για να μπορέσουμε να επιτύχουμε το στόχο μας με αυτά τα υλικά, θα πρέπει για αρχή να επικοινωνήσει το ESP32 με το λάπτοπ για μπορούν να ανταλλάξουν δεδομένα. Σε αυτό θα μας βοηθήσει, το τσίπακι που έχουν τα ESP32 και τους επιτρέπει τη χρήση ασύρματου ίντερνετ. Έπειτα μέσω κώδικα, θα επιτρέψουμε στο ESP32 να λαμβάνει δεδομένα από τους αισθητήρες, τους κωδικοποιητές και το γυροσκόπιο, να στέλνει αυτά τα δεδομένα στο λάπτοπ, όπου με τη βοήθεια του ROS, θα γίνει η κατάλληλη επεξεργασία των δεδομένων αυτών και στη συνέχεια, θα στέλνει τις πρέπουσες εντολές στο ESP32 για το ποιες κινήσεις πρέπει να κάνει το όχημά μας.

Μετά που θα έχουμε επιτύχει να δημιουργήσουμε όλες αυτές τις ανταλλαγές δεδομένων, μένει ο τελικός σκοπός του συστήματος, οποίος είναι να πλοηγείται χωρίς ανθρώπινη παρέμβαση, μέσα σε ένα περιβάλλον που θα δημιουργήσουμε με εμπόδια, έτσι

ώστε να μπορεί να τα αποφύγει και συνάμα να καταφέρει να επιτύχει όσο το δυνατόν καλύτερη χαρτογράφηση του περιβάλλοντος.

3.3. Αρχιτεκτονική συστήματος

Το σύστημα μας όπως είδαμε επικοινωνεί με το ESP32 μέσω Wifi, και με τη βοήθεια του Rosserial, που είναι μια έκδοση ROS σημείου σε σημείου για επικοινωνία μέσω σειριακής, κυρίως για την ενσωμάτωση μικροελεγκτών χαμηλού κόστους σε ROS. Η σειριακή του ROS αποτελείται από ένα γενικό πρωτόκολλο ομότιμου δικτύου, και βιβλιοθήκες για χρήση με το ESP32 [10]. Μέσω αυτού γίνεται η ανταλλαγή δεδομένων και εντολών απο το ROS προς το ESP32. Το σύστημα μας θα έχει τη μορφή όπως παρουσιάζεται στη παρακάτω εικόνα:



Εικόνα 4. Παράδειγμα τελικής μορφής του αυτόνομου ρομποτικού συστήματος πλοήγησης.

3.4 Δυνατότητες συστήματος

Με την υλοποίηση του συστήματός μας, έχουμε ως στόχο να μπορούμε να παρέχουμε τις παρακάτω δυνατότητες στον χρήστη:

- **Παρακολούθηση σε πραγματικό χρόνο:**
  Το σύστημα μας, μέσω του εργαλείου Rviz του ROS, θα μπορεί ο χρήστης να παρακολουθεί τη πορεία του οχήματος, ακόμα και αν δεν έχει άμεση ορατότητα σε αυτό, όπως και παρακολούθηση των τιμών όλων των αισθητήρων.

- **Έλεγχος του οχήματος σε πραγματικό χρόνο:**
  Αν για κάποιο λόγο υπάρξει πρόβλημα με την αυτόνομη πλοήγηση του οχήματος, ο χρήστης μέσω χειριστηρίου που παρέχεται από το ROS, θα μπορεί να πλοήγησει το όχημα.

- **Αποφυγή εμποδίων και υπολογισμός συντομότερης διαδρομής:**
  Με τη βοήθεια του ROS, το όχημα μας, θα μπορεί να αποφύγει εμπόδια και να ακολουθήσει το συντομότερο μονοπάτι για τον στόχο που έχουμε θέσει.
- **Χαρτογράφηση σε πραγματικό χρόνο:**
  Καθώς το όχημα θα προχωράει προς την επίτευξη του στόχου που του έχουμε θέσει, με τη βοήθεια των αισθητήρων απόστασης, θα μπορεί να δημιουργήσει χάρτη, για μελλοντική πλοήγηση για αυτό ή για άλλα οχήματα.

4. Επιμέρους Υποσυστήματα

4.1. Υλικό

4.1.1. Μικροελεγκτές

4.1.1.1. ESP32

Το ESP32 είναι μια σειρά από συστήματα χαμηλού κόστους και χαμηλής κατανάλωσης σε μικροελεγκτές τσιπ με ενσωματωμένο Wi-Fi και Bluetooth διπλής λειτουργίας. Η σειρά ESP32 χρησιμοποιεί είτε έναν μικροεπεξεργαστή Xtensa LX6 της Tensilica σε παραλλαγές διπλού ή μονού πυρήνα,είτε μικροεπεξεργαστή διπλού πυρήνα Xtensa LX7 ή μικροεπεξεργαστή RISC-V ενός πυρήνα, που περιλαμβάνει ενσωματωμένους διακόπτες κεραίας, RF-balum,ενισχυτή ισχύος,ενισχυτή λήψης χαμηλού θορύβου, φίλτρα και μονάδες διαχείρισης ενέργειας [11].

Εικόνα 5. ESP32

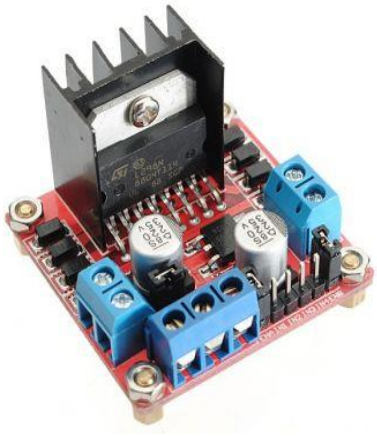
Κύρια χαρακτηριστικά ενός ESP32:

- Κρυπτογράφηση: AES,WPA,WPA2-PSK,WPS
- Υποστηριζόμενα πρωτόκολλα: 802.11 b/g/n
- Συχνότητα: 2.4 GHz με 2.5 GHz
- Τάση τροφοδοσίας λειτουργίας: 5V
- Τύπος διεπαφής: USB
- Διεπαφή: GPIO,I2C x2,I2S x2,SDIO,SPI x3,UART x3,USB

4.1.2. Ενεργοποιητές

4.1.2.1. H-Bridge

Η L298N γέφυρα είναι ένας οδηγός διπλής πλήρους γέφυρας υψηλής τάσης,υψηλού ρεύματος που έχει σχεδιαστεί για να δέχεται τυπικά λογικά επίπεδα TTL και να οδηγεί επαγωγικά φορτία όπως ρελέ, ηλεκτρομαγνητικές βαλβίδες, DC και βηματικούς κινητήρες [12].



Εικόνα 6. L298N H-Bridge

Η L298N παρέχει τέσσερις ακίδες για τον έλεγχο των κινητήρων.τέσσερις εξόδους για τη τροφοδοσία των κινητήρων,δύο ακίδες για χρήση PWM, για έλεγχο της ταχύτητας των κινητήρων, δύο εισόδους τροφοδοσίας δώδεκα Volt και πέντε Volt και μία για τη γείωση.

4.1.2.2. Κινητήρας DC

Ο κινητήρας DC είναι ένας κινητήρας μονού άξονα, με τάση λειτουργίας 3V DC και RPM 125R/ λεπτό. Επιτρέπει την εναλλαγή της ταχύτητας μέσω PWM, χρησιμοποιείται συνήθως σε εφαρμογές ρομποτικής ή και άλλα.

Εικόνα 7. Κινητήρας DC

4.1.3. Αισθητήρες

4.1.3.1. HC-SR04

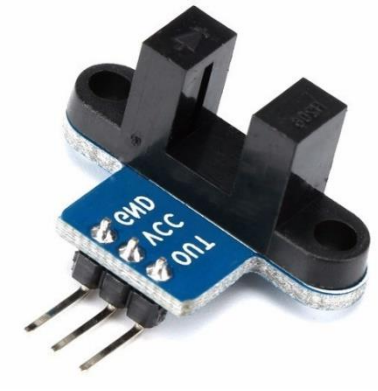
Ο HC-SR04 είναι ένας αισθητήρας απόστασης. Είναι ένας οικονομικός αισθητήρας, παρέχει λειτουργία μέτρησης χωρίς επαφή, από δύο εκατοστά έως τετρακόσια εκατοστά και με ακρίβεια που μπορεί να φτάσει τα τρία χιλιοστά. Κάθε μονάδα HC-SR04 περιλαμβάνει έναν πομπό υπερήχων, έναν δέκτη και ένα κύκλωμα ελέγχου.

Εικόνα 8. Αισθητήρας απόστασης HC-SR04

Ο αισθητήρας έχει τέσσερις ακίδες, μία για τη τροφοδοσία, μία για τη γείωση, μία για τον πομπό και μια για τον δέκτη.

4.1.3.2. Αισθητήρας μέτρησης ταχύτητας

Αυτή η μονάδα είναι μια έξυπνη μονάδα μέτρησης της ταχύτητας. Η μία πλευρά εκπέμπει υπέρυθρο φως ενώ η άλλη έχει δέκτη φωτός. Οταν περιστρέφεται ένας δίσκος κωδικοποιητή, οι υποδοχές επιτρέπουν στο δέκτη να ενεργοποιείται και να απενεργοποιείται. Αυτό είναι ανάλογο της ταχύτητας.

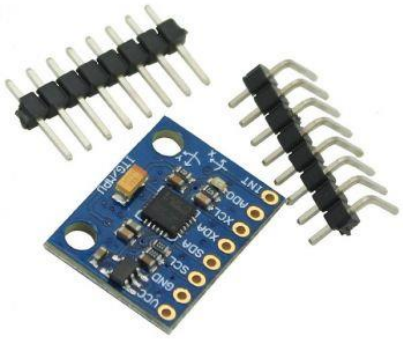


Εικόνα 9.Αισθητήρας μέτρησης ταχύτητας.

Το σήμα καθαρίζεται μέσω του ενσωματωμένου μετατροπέα Schmidt και στην συνέχεια αποστέλλεται στο μικροελεγκτή για επεξεργασία. Ο αισθητήρας έχει τρείς ακίδες, μία για τη τροφοδοσία, μια για τη γείωση και μία έξοδο για τον μικροελεγκτή.

4.1.3.3. IMU-MPU6050

Η μονάδα αδρανειακής μέτρησης είναι μια ηλεκτρονική συσκευή που μετρά και αναφέρει τη συγκεκριμένη δύναμη, τον γωνιακό ρυθμό και μερικές φορές τον προσανατολισμό του σώματος, χρησιμοποιώντας έναν συνδυασμό επιταχυνσιόμετρων, γυροσκόπιων και μερικές φορές μαγνητόμετρων [13].



Εικόνα 10. MPU6050-IMU

Εμείς χρησιμοποιούμε τη μονάδα MPU-6050, που περιέχει ένα επιταχυνσιόμετρο και ένα γυροσκόπιο σε ένα μόνο τσιπ. Είναι πολύ ακριβής, καθώς περιέχει υλικό μετατροπής αναλογικού σε ψηφιακό, 16 bits για κάθε κανάλι. Επομένως, μπορεί να λαμβάνει τα κανάλια X, Y, Z ταυτόχρονα. Ο αισθητήρας χρησιμοποιεί το δίαυτλο I2C για διασύνδεση με το ESP32.

4.1.4. Powerbank

Για πηγή τροφοδοσίας του ESP-32 χρησιμοποιούμε ένα powerbank της Trust. Το powerbank είναι ένας φορητός φορρτιστής, ο οποίος διαφέρει σε μεγέθη και έτσι είναι πολύ βολικό για χρήση στη πτυχιακή.

Εικόνα 11. Powerbank της Trust

4.2. Λογισμικό

4.2.1. ROS Noetic Ninjemys

Το ROS Noetic Ninjemys είναι μια διανομή ROS, δηλαδή είναι ένα εκδομένο σύνολο πακέτων ROS. Ο σκοπός των διανομών ROS είναι να αφήσουν τους προγραμματιστές να εργαστούν ενάντια σε μια σχετικά σταθερή βάση κώδικα. Επομένως, μόλις κυκλοφορήσει μια διανομή, προσπαθούν να περιορίσουν τις αλλαγές σε επιδιορθώσεις σφαλμάτων και βελτιώσεις χωρίς σοβαρά σφάλματα σε πακέτα του πυρήνα. Και γενικά αυτό ισχύει για όλη την κοινότητα, αλλά για πακέτα "υψηλότερου"επιπέδου, οι κανόνες είναι λιγότερο αυστηροί,

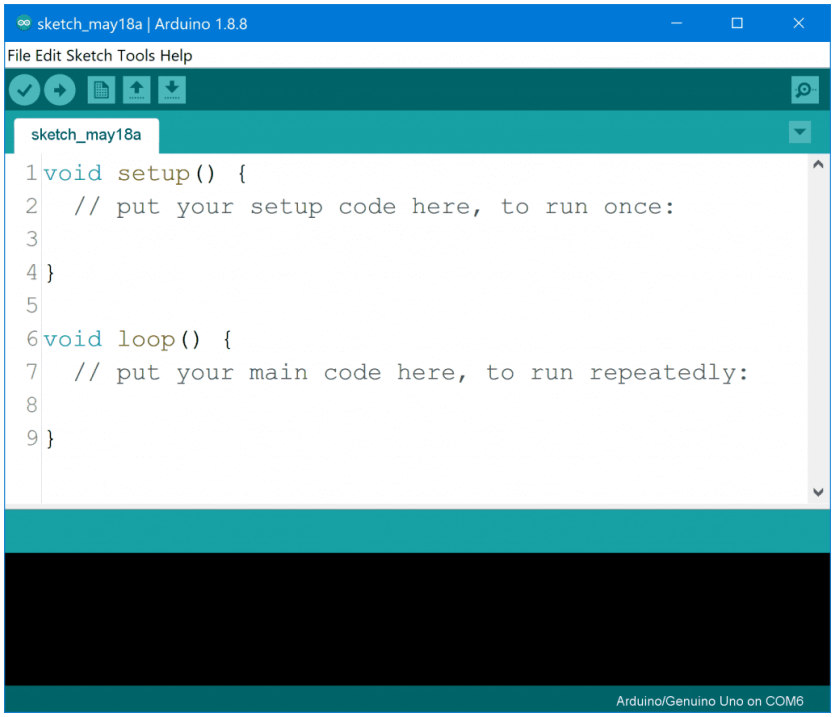
και έτσι εναπόκειται στους συντηρητές ενός συγκεκριμένου πακέτου, να αποφύγουν τις παραβιάσεις των αλλαγών [14].

Εικόνα 12. ROS Noetic Ninjemys

4.2.2. Arduino Integrated Development Environment (IDE)

Το ολοκληρωμένο περιβάλλον ανάπτυξης Arduino, περιέχει ένα πρόγραμμα επεξεργασίας κειμένου για τη σύνταξη κώδικα, μια περιοχή, μια κονσόλα κειμένου, μια γραμμή εργαλείων με κουμπιά για κοινές λειτουργίες και μια σειρά μενού. Συνδέεται με το υλικό Arduino ή στη δική μας περίπτωση ESP32, για να ανεβάζει προγράμματα και να επικοινωνεί μαζί τους.

Τα προγράμματα που γράφτηκαν με χρήση λογισμικού Arduino, ονομάζονται σκίτσα. Αυτά τα σκίτσα γράφονται στο πρόγραμμα επεξεργασίας κειμένου και αποθηκεύονται με την επέκταση αρχείου .ino. Το πρόγραμμα επεξεργασίας έχει δυνατότητες για αποκοπή/επικόλληση και για αναζήτηση/αντικατάσταση κειμένου. Η περιοχή μηνυμάτων παρέχει σχόλια κατά την αποθήκευση και την εξαγωγή και εμφανίζει επίσης σφάλματα. Η κονσόλα εμφανίζει την έξοδο κειμένου από το λογισμικό Arduino, συμπεριλαμβανομένων πλήρων μηνυμάτων σφάλματος και άλλων πληροφοριών. Η κάτω δεξιά γωνία του παραθύρου εμφανίζει τη διαμορφωμένη πλακέτα και τη σειριακή θύρα. Τα κουμπιά της γραμμής εργαλείων, επιτρέπουν, την επαλήθευση και το ανέβασμα των προγραμμάτων, το άνοιγμα και η αποθήκευση σκίτσων και το άνοιγμα της σειριακής οθόνης [15].



Εικόνα 13. Περιβάλλον Arduino IDE

4.2.3 Rosserial Arduino

Το rosserial Arduino είναι ένα πακέτο με το οποίο μπορούμε να χρησιμοποιήσουμε το ROS απευθείας με το Arduino IDE. Το rosserial παρέχει ένα πρωτόκολλο επικοινωνίας ROS, που λειτουργεί πάνω από το UART του Arduino. Επιτρέπει στο Arduino, να είναι ένας πλήρης κόμβος ROS που μπορεί να δημοσιεύει απευθείας και να εγγράφεται σε μηνύματα ROS, να δημοσιεύει μετασχηματισμούς TF και να παίρνει το χρόνο συστήματος ROS [16].

5. Ανάπτυξη συστήματος

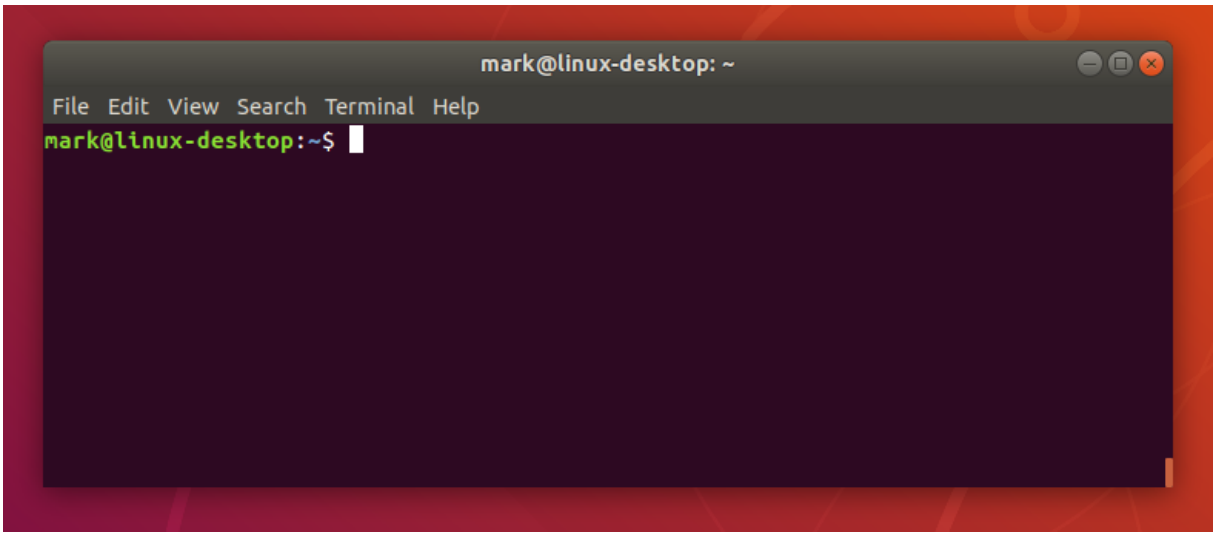
5.1. Εγκατάσταση λογισμικού

5.1.1. Εγκατάσταση ROS

Σε αυτό το σημείο της πτυχιακής θα αναφερθούμε στην υλοποίηση της πτυχιακής μου, αναφέροντας τα βήματα που ακολουθήσαμε για την επίτευξή της.

Ξεκινώντας, θα αναφερθούμε στην εγκατάσταση του λογισμικού. Η οποία θα επιτρέψει την επικοινωνία μεταξύ ROS και οχήματος, αλλά και τη φόρτωση του κατάλληλου κώδικα στο ESP32 για τη σωστή λειτουργία του οχήματος. Αναλυτικότερα, θα εγκατασταθούν, το ROS, το Arduino IDE, και το Rosserial.

Για αρχή, θα ανοίξουμε τη γραμμή εντολών του Ubuntu όπου εκεί θα εισάγουμε τις παρακάτω εντολές.



Εικόνα. Γραμμή εντολών Ubuntu

Αρχικά, θα πρέπει να προσαρμόσουμε τον υπολογιστή μας, να δεχτεί λογισμικό απο το packages.ros.org. Με την παρακάτω εντολή επιτυγχάνεται αυτό.

```
sudo sh -c 'echo "deb http://packages.ros.org/ros/ubuntu $(lsb_release -sc) main" > /etc/apt/sources.list.d/ros-latest.list'
```

Εικόνα 14. Κώδικας για αποδοχή λήψης λογισμικού απο packages.ros.org

Έπειτα θα πρέπει να εγκαταστήσουμε τα κλειδιά και να κάνουμε εγκατάσταση την εντολή curl, αν δεν την έχουμε κάνει ήδη.

```
sudo apt install curl # if you haven't already installed curl
curl -s https://raw.githubusercontent.com/ros/rosdistro/master/ros.asc | sudo apt-key add -
```

Εικόνα 15. Κώδικας για εγκατάσταση κλειδιού.

Τέλος, πριν ξεκινήσουμε την εγκατάσταση του ROS, πρέπει να βεβαιωθούμε ότι το ευρετήριο του πακέτου Debian είναι ενημερωμένο.

```
sudo apt update
```

Εικόνα 16. Ενημέρωση πακέτου Debian

Είμαστε έτοιμοι να ξεκινήσουμε την εγκατάσταση. Θα επιλέξουμε την πλήρη εγκατάσταση του ROS, η οποία είναι και η προτεινόμενη.

```
sudo apt install ros-noetic-desktop-full
```

Εικόνα 17. Εγκατάσταση του ROS

Μόλις γράψουμε την εντολή αυτή, στη γραμμή εντολών, θα μας εμφανίζει τα πακέτα που θα εγκατασταθούν, το χώρο που θα καταλάβουν και την επικύρωση της ενέργειας από εμάς.

```
  ros-noetic-rqt-common-plugins ros-noetic-rqt-console
  ros-noetic-rqt-dep ros-noetic-rqt-graph ros-noetic-rqt-gui
  ros-noetic-rqt-gui-cpp ros-noetic-rqt-gui-py
  ros-noetic-rqt-image-view ros-noetic-rqt-launch
  ros-noetic-rqt-logger-level ros-noetic-rqt-moveit
  ros-noetic-rqt-msg ros-noetic-rqt-nav-view ros-noetic-rqt-plot
  ros-noetic-rqt-pose-view ros-noetic-rqt-publisher
  ros-noetic-rqt-py-common ros-noetic-rqt-py-console
  ros-noetic-rqt-reconfigure ros-noetic-rqt-robot-dashboard
  ros-noetic-rqt-robot-monitor ros-noetic-rqt-robot-plugins
  ros-noetic-rqt-robot-steering ros-noetic-rqt-runtime-monitor
  ros-noetic-rqt-rviz ros-noetic-rqt-service-caller
  ros-noetic-rqt-shell ros-noetic-rqt-srv ros-noetic-rqt-tf-tree
  ros-noetic-rqt-top ros-noetic-rqt-topic ros-noetic-rqt-web
  ros-noetic-rviz ros-noetic-rviz-plugin-tutorials
  ros-noetic-rviz-python-tutorial ros-noetic-self-test
  ros-noetic-sensor-msgs ros-noetic-shape-msgs ros-noetic-smach
  ros-noetic-smach-msgs ros-noetic-smach-ros ros-noetic-smclib
  ros-noetic-std-msgs ros-noetic-std-srvs ros-noetic-stereo-msgs
  ros-noetic-tf ros-noetic-tf-conversions ros-noetic-tf2
  ros-noetic-tf2-geometry-msgs ros-noetic-tf2-kdl
  ros-noetic-tf2-msgs ros-noetic-tf2-py ros-noetic-tf2-ros
  ros-noetic-topic-tools ros-noetic-trajectory-msgs
  ros-noetic-turtle-actionlib ros-noetic-turtle-tf
  ros-noetic-turtle-tf2 ros-noetic-turtlesim ros-noetic-urdf
  ros-noetic-urdf-parser-plugin ros-noetic-urdf-tutorial
  ros-noetic-visualization-marker-tutorials
  ros-noetic-visualization-msgs ros-noetic-visualization-tutorials
  ros-noetic-viz ros-noetic-webkit-dependency ros-noetic-xacro
  ros-noetic-xmlrpcpp sbcl shiboken2 sip-dev tango-icon-theme
  uuid-dev va-driver-all vdpau-driver-all x11proto-core-dev
  x11proto-dev xorg-sgml-doctools xtrans-dev zlib1g-dev
0 upgraded, 649 newly installed, 0 to remove and 0 not upgraded.
Need to get 295 MB of archives.
After this operation, 1 615 MB of additional disk space will be used.
Do you want to continue? [Y/n]
```

Εικόνα 18.Εγκατάσταση ROS

Μετά την εγκατάσταση του ROS, κάθε φορά που ανοίγουμε καινούργια γραμμή εντολών θα πρέπει να τρέχουμε τη παρακάτων εντολή.

```
source /opt/ros/noetic/setup.bash
```

Εικόνα 19. Εντολή Περιβάλλοντος ROS

Η επόμενη εντολή είναι ένα εργαλείο αλλά και κάποιες εξαρτήσεις για να μπορούμε να δημιουργούμε τα δικά μας πακέτα.

```
sudo apt install python3-rosdep python3-rosinstall python3-rosinstall-generator python3-wstool bu
ild-essential
```

Εικόνα 20. Εντολή για εργαλείο δημιουργίας πακέτων από τον χρήστη.

Για να ολοκληρώσουμε την εγκατάσταση του ROS, θα πρέπει να αρχικοποιήσουμε το rosdep. Αυτό μας επιτρέπει την εύκολη εγκατάσταση εξαρτήσεων του συστήματος από τη πηγή που θέλουμε να μεταγλωττίσουμε και είναι απαραίτητο για τρέξουμε κάποια εξαρτήματα του πυρήνα του ROS.

```
sudo apt install python3-rosdep
```

Εικόνα 21.Εγκατάσταση rosdep

```
sudo rosdep init
rosdep update
```

Εικόνα 22.Αρχικοποίηση του rosdep.

Στη συνέχεια θα φτιάξουμε το χώρο εργασίας του ROS.

```
$ mkdir -p ~/catkin_ws/src
$ cd ~/catkin_ws/
$ catkin_make
```

Με την εντολή mkdir, θα δημιουργήσουμε το φάκελο catkin_ws και το src. Η εντολή catkin_make είναι ένα εργαλείο για δουλεύουμε με χώρους εργασίας catkin. Τρέχοντας το για πρώτη φορά, θα δημιουργηθεί το αρχείο CMakeLists.txt μέσα στον φάκελο src.

Τέλος θα τρέξουμε την εντολή source,για να επικαλύψουμε τον χώρο εργασίας που θέλουμε πάνω απο το περιβάλλον μας.

```
$ source devel/setup.bash
```

Εικόνα 23. Εντολή source

5.1.2. Εγκατάσταση Arduino IDE

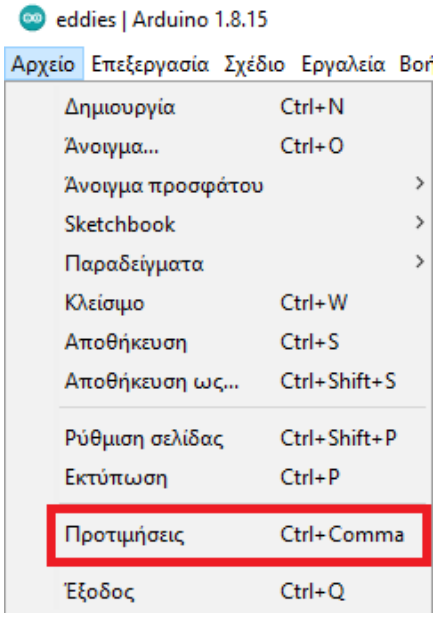
Έπειτα θα περάσουμε στην εγκατάσταση του Arduino IDE. Μετά που θα τελειώσει η λήψη του αρχείου, θα κάνουμε εξαγωγή του αρχείου σε όποιο φάκελο θέλουμε. Τέλος, ανοίγοντας τη γραμμή εντολών, θα πλοηγηθούμε στο φάκελο που έχουμε εξάγει και θα πληκτρολογήσουμε τη παρακάτω εντολή.

```
./install.sh
```

Εικόνα 24. Εγκατάσταση Arduino IDE

Μόλις πατήσουμε enter, θα εκτελεστεί η εγκατάσταση του αρχείου. Μόλις ολοκληρωθεί, ένα εικονίδιο του Arduino IDE θα εμφανιστεί στην επιφάνεια εργασίας.

Έπειτα, θα ανοίξουμε το Arduino IDE για να εγκαταστήσουμε το πακέτο που θα υποστηρίζει το ESP32. Θα πατήσουμε το **Αρχείο➡ Προτιμήσεις**.

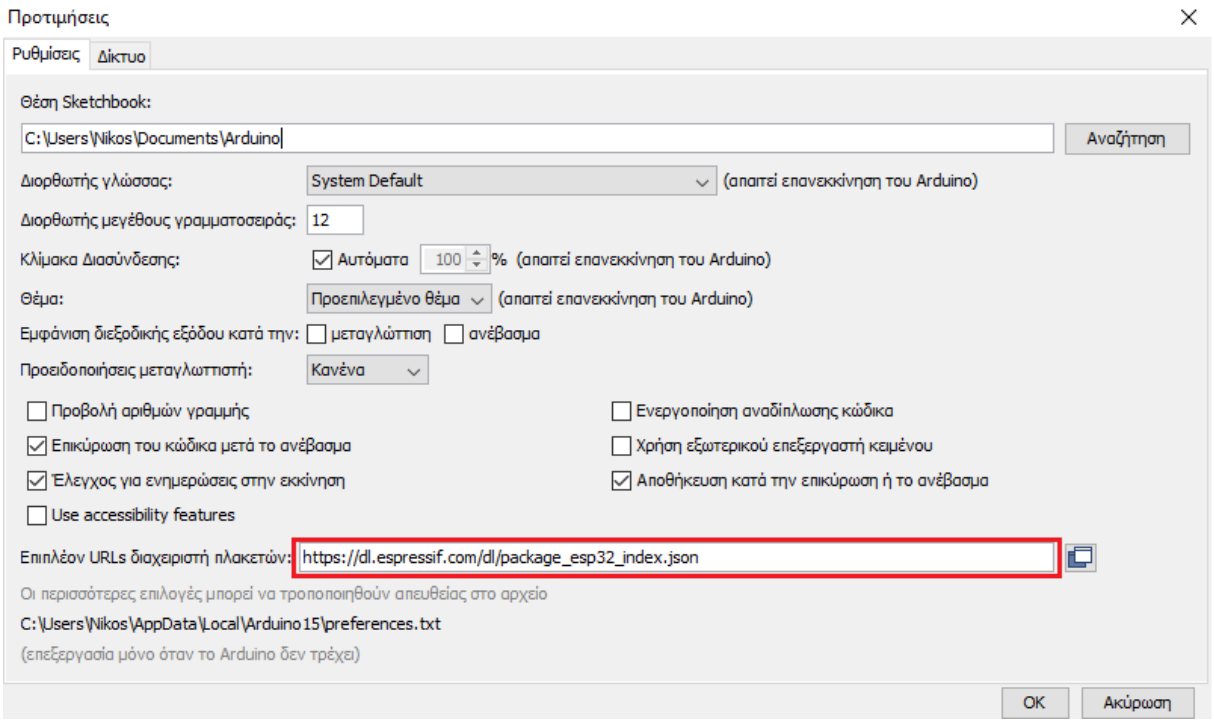


Εικόνα 25. Μενού Arduino IDE

Στο πεδίο "Επιπλέον URLs διαχειριστή πλακετών", βάζουμε το ακόλουθο:

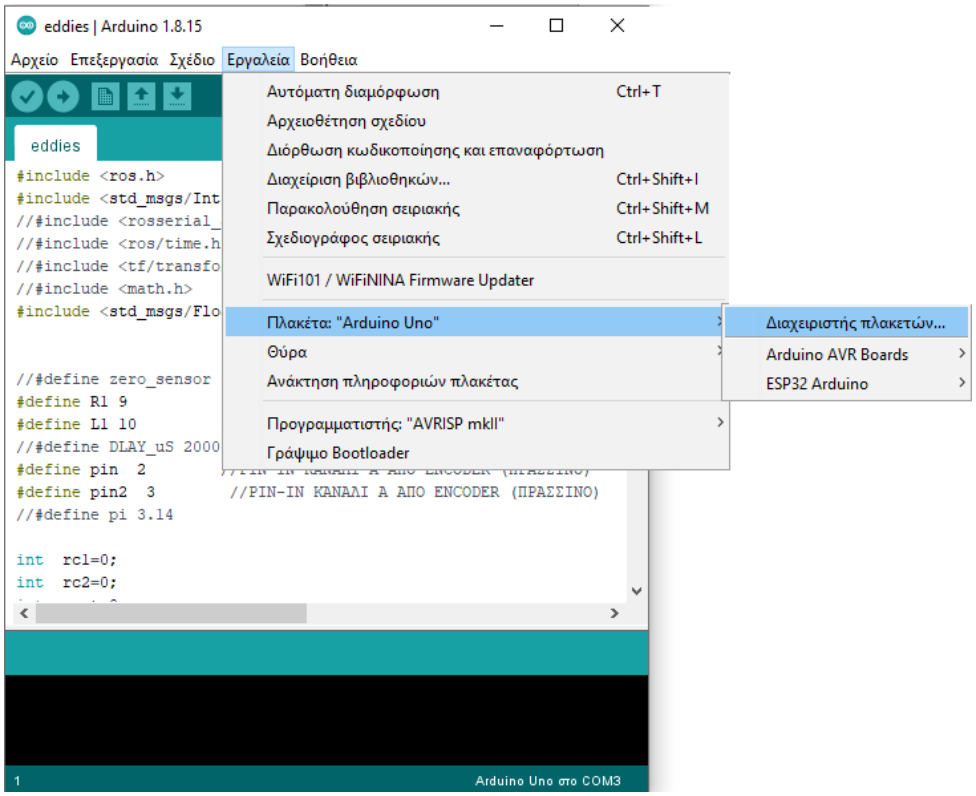
```
https://raw.githubusercontent.com/espressif/arduino-esp32/gh-pages/package_esp32_index.json
```

Και πατάμε το OK.



Εικόνα 26.Μενού προτιμήσεων Arduino IDE

Μετά, θα πλοηγηθούμε **Εργαλεία ➡ Πλακέτα ➡ Διαχειριστής πλακετών**



Εικόνα 27. Πλοήγηση στα εργαλεία.

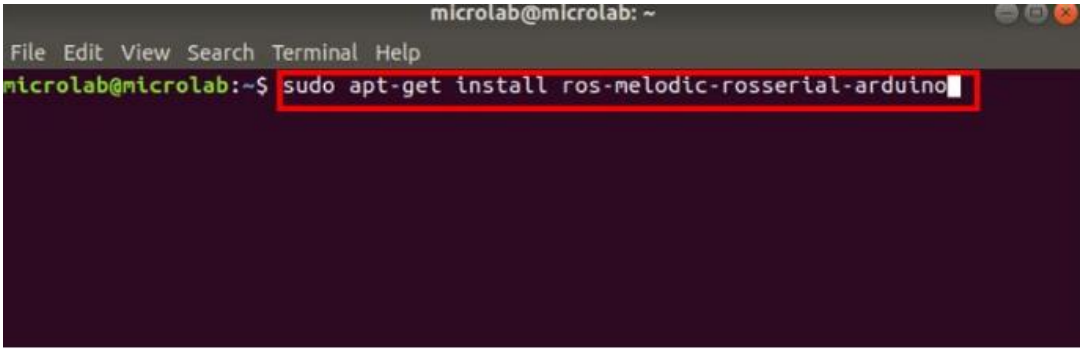
Εικόνα 28. Διαχειριστής πλακετών

5.1.3. Εγκατάσταση Rosserial

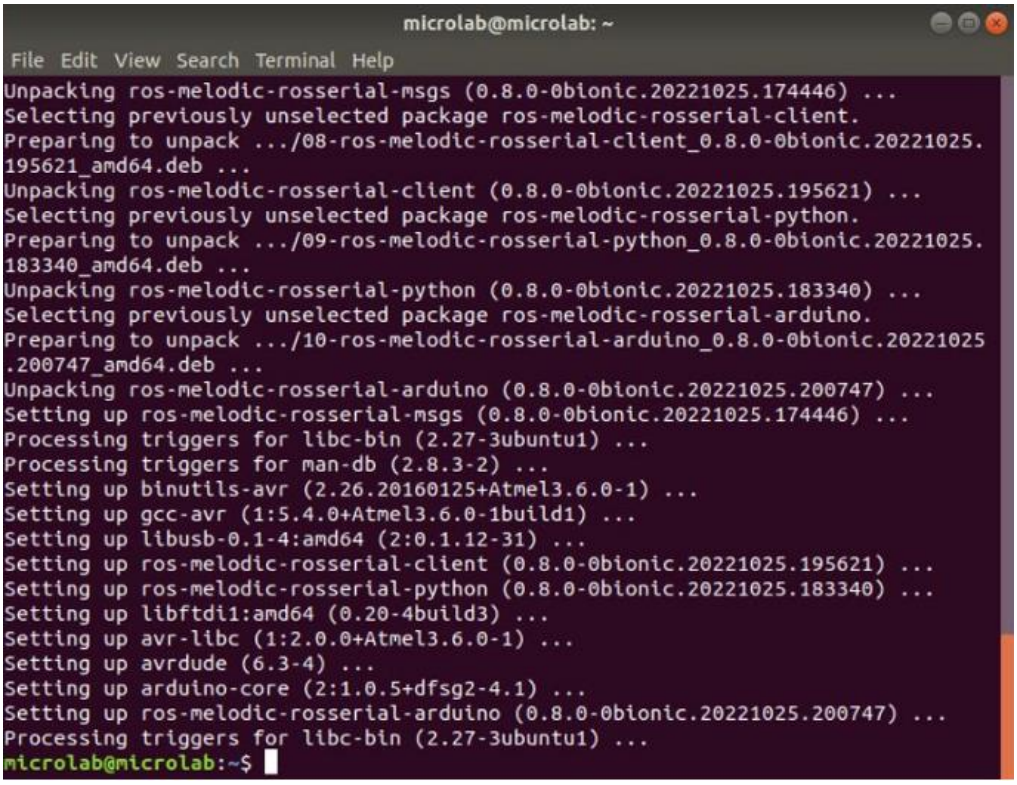
Το επόμενο και τελευταίο βήμα είναι να εγκαταστήσουμε το πακέτο Rosserial στο ROS και στο Arduino IDE. Ανοίγουμε τη γραμμή εντολών και βάζουμε τη παρακάτω εντολή.

```
sudo apt-get install ros-<distro>-rosserial
sudo apt-get install ros-<distro>-rosserial-arduino
```

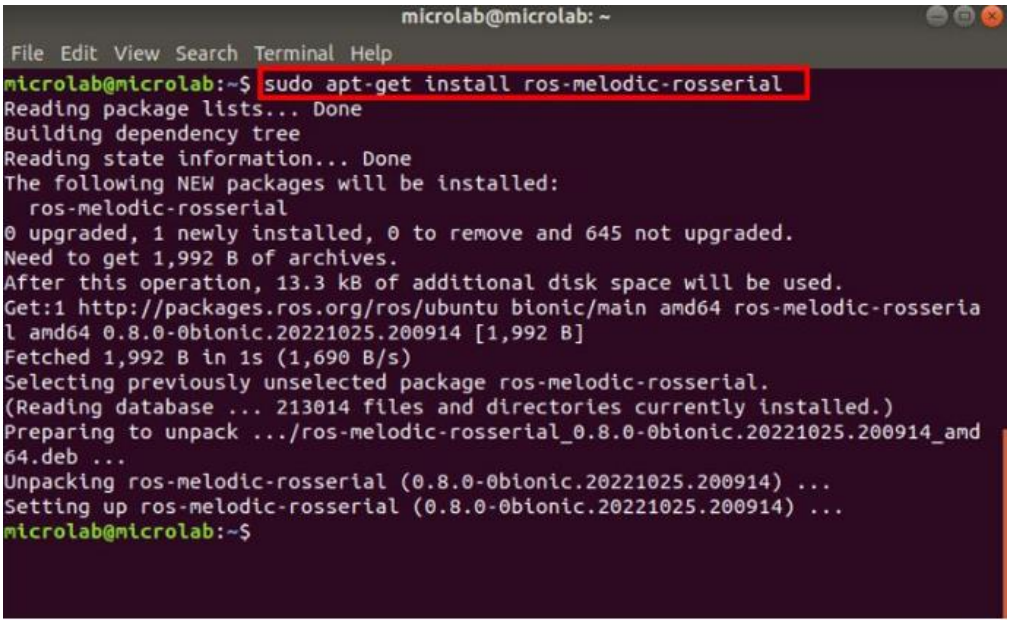
Βάζουμε την έκδοση που έχουμε, εμείς χρησιμοποιούμε την έκδοση Noetic.



Εικόνα 29. Εγκατάσταση Rosserial

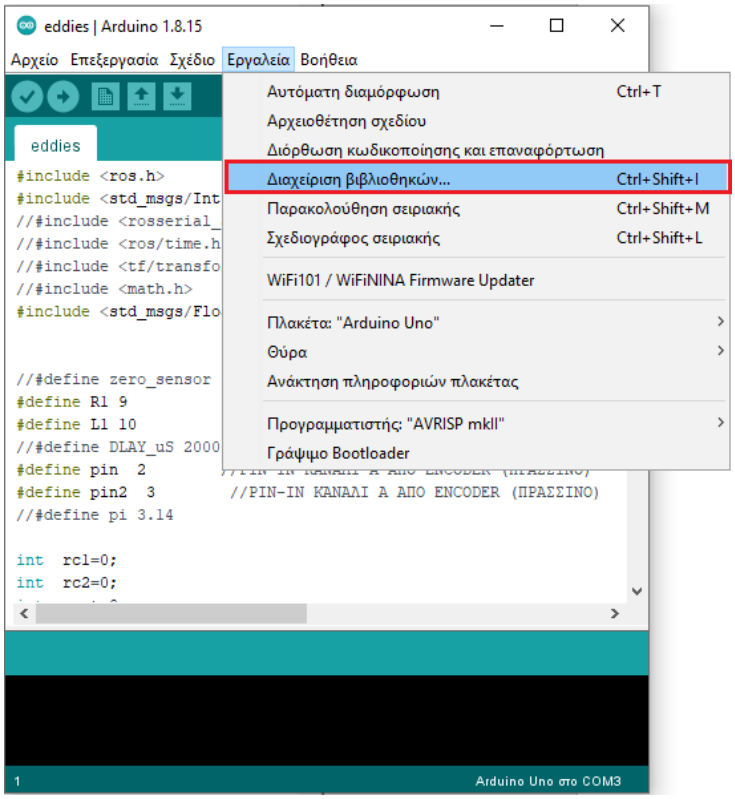


Εικόνα 30. Εγκατάσταση Rosserial



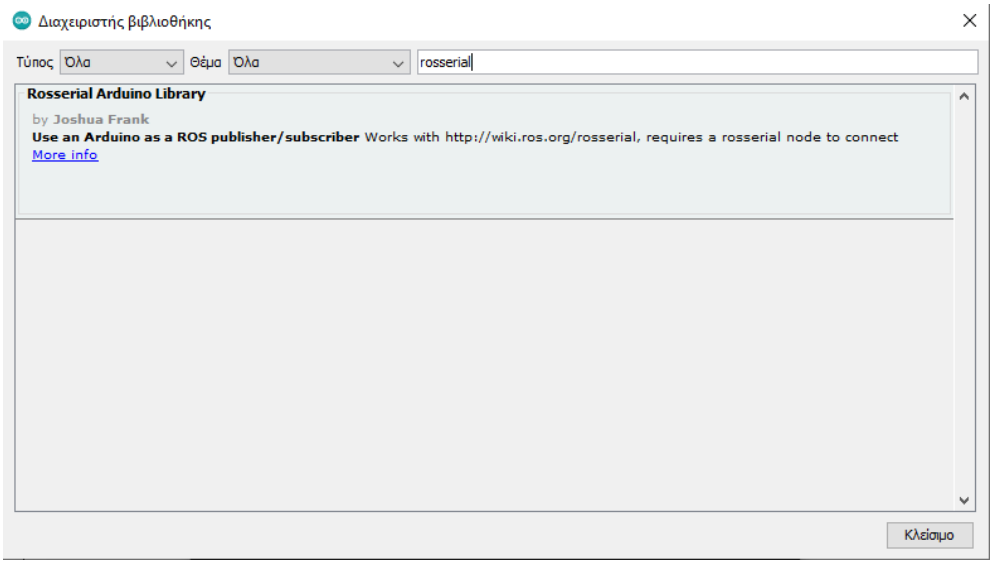
Εικόνα 31. Εγκατάσταση Rosserial

Μετά ανοίγουμε πάλι το Arduino IDE για να εγκαταστήσουμε και εκεί το πακέτο rosserial. Πάμε **εργαλεία➡ Διαχείριση βιβλιοθηκών**.



Εικόνα 32.Εγκατάσταση rosserial στο arduino IDE

Κάνουμε αναζήτηση το Rosserial, και κάνουμε εγκατάσταση το Rosserial Arduino Library.

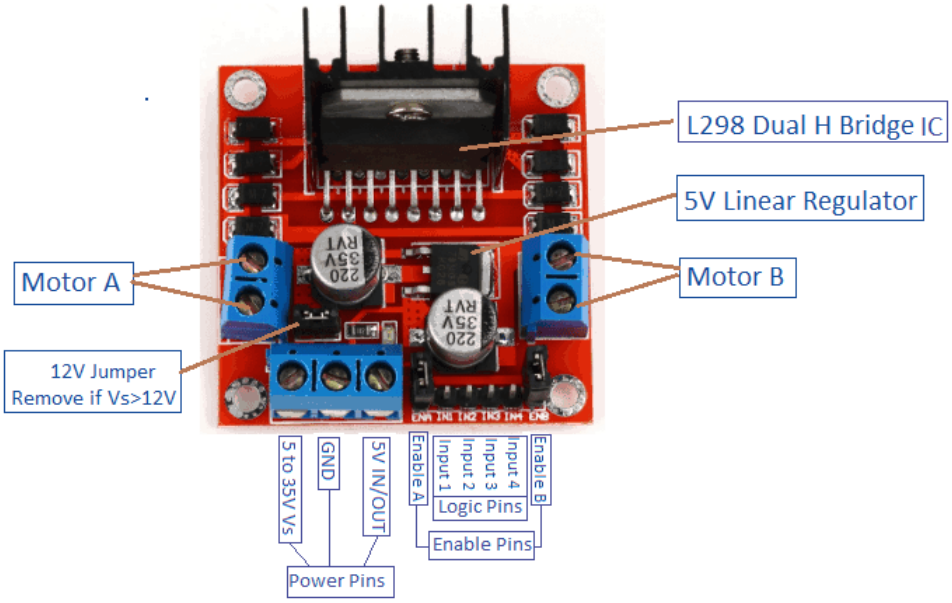


Εικόνα 33.Διαχειριστής Βιβλιοθήκης

5.2. Συνδεσμολογία Οχήματος

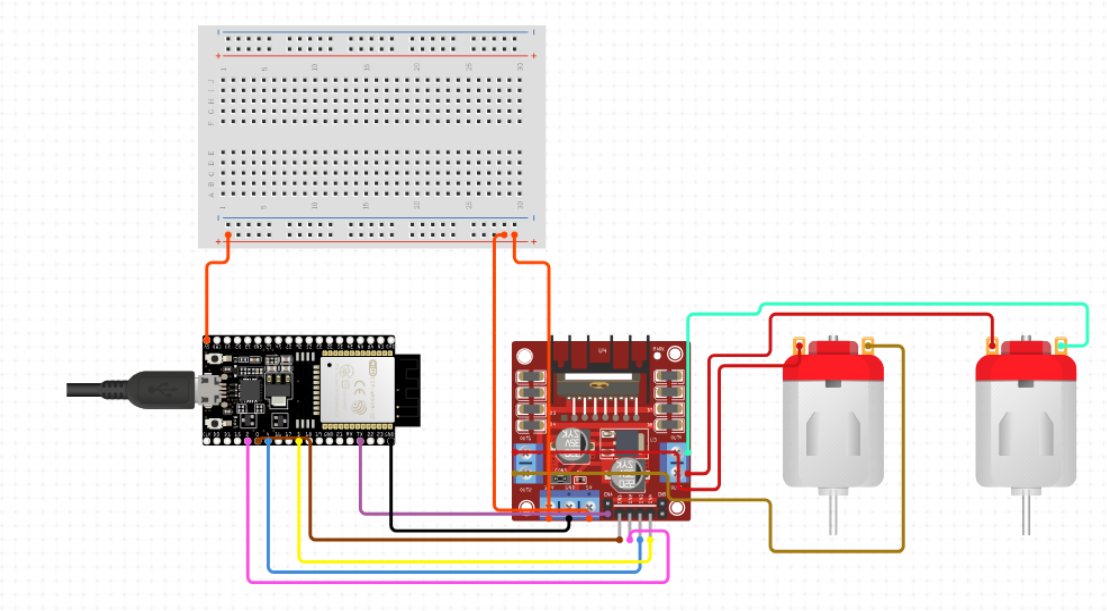
5.2.1. Συνδεσμολογία LN298N-DC κινητήρες

Σε αυτή τη παράγραφο θα περιγράψω τη συνδεσμολογία του οχήματος. Ξεκινώντας θα περιγράψουμε τη συνδεσμολογία μεταξύ ESP32, LN298N και των δύο DC κινητήρων. Αρχικά έχουμε τοποθετήσει τη θήκη με τις τέσσερις AA μπαταρίες στη κάτω πλευρά του οχήματος. Η θήκη έχει δύο καλώδια, ένα κόκκινο που είναι το + και ένα μαύρο καλώδιο που είναι το -. Το κόκκινο καλώδιο ενώνεται με το κόκκινο καλώδιο που έχει το κουμπί για ON/OFF. Το μαύρο καλώδιο πάει στη θέση GND που έχει η LN298N. Το μάυρο καλώδιο του κουμπιού, πάει στη θέση 12V είσοδο της LN298N. Ο κινητήρας έχει 2 καλώδια που πάνε στο OUT1 και OUT2 της LN298N, και για τον άλλο κινητήρα στη θέση OUT3 και OUT4 αντίστοιχα.



Εικόνα 34. LN298N

Η LN298N, έχει έξι θέσεις για τον έλεγχο των κινητήρων. Οι δύο θέσεις ENA και ENB είναι για τη χρήση PWM και τον έλεγχο της ταχύτητας του οχήματος. Έπειτα υπάρχουν οι θέσεις IN1, IN2, IN3, IN4. Οι έξι αυτές θέσεις συνδέονται στο ESP32.



Εικόνα 35.Συνδεσμολογία ESP32,LN298N και DC κινητήρες

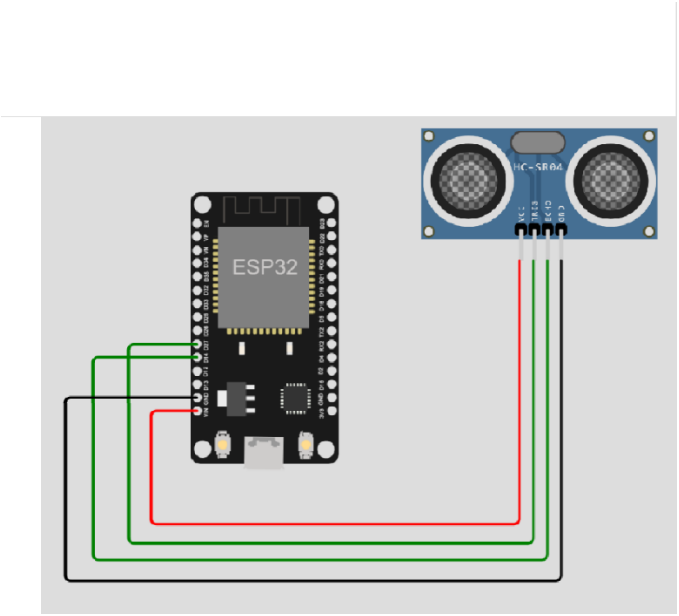
- Κώδικας Arduino:

```
//LN298N
const uint8_t ENA = 23;
const uint8_t R_FORW = 5;
const uint8_t R_BACK = 15;
const uint8_t ENB = 17;
const uint8_t L_FORW = 32;
const uint8_t L_BACK = 13;
```

Εικόνα 36.Κώδικας LN298N

5.2.2. Συνδεσμολογία HC-SR04

Στο όχημα έχουμε 5 αισθητήρες υπερήχων, με γωνία 45^{o} μεταξύ τους. Οι αισθητήρες είναι τοποθετημένοι να μετρούν την απόσταση στο μπροστά μέρος του οχήματος, έχουν εύρος 180 μοιρών.



Εικόνα 37. Συνδεσμολογία HC-SR04

- Κώδικας Arduino:

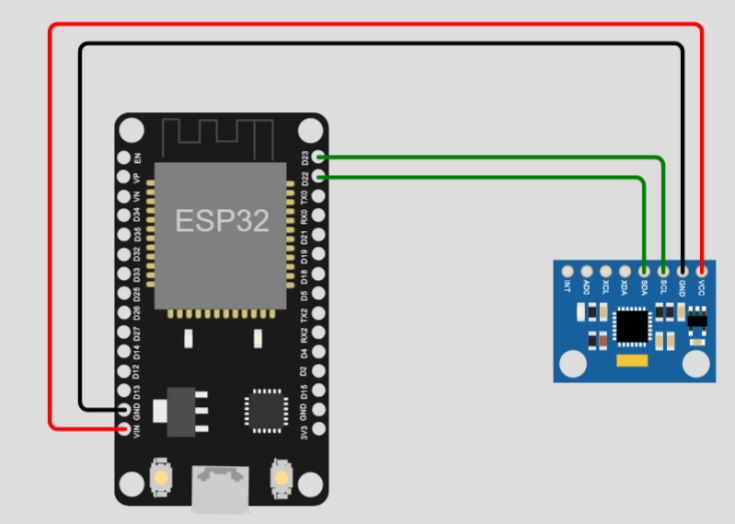
```
//HC-SR04
float readSonar(const int trig, const int echo) {

  RunningMedian samples= RunningMedian(15);

  pinMode(trig, OUTPUT);
  pinMode(echo, INPUT);

  digitalWrite(trig, LOW);
  delayMicroseconds(2);
  digitalWrite(trig, HIGH);
  delayMicroseconds(10);
  digitalWrite(trig, LOW);

  long duration = pulseIn(echo, HIGH);
  long distance= (duration/2)*0.034;

  samples.add(distance);

  long m=samples.getMedian();
  return(m/100.0);

}
```

Εικόνα 38.Κώδικας HC-SR04

Η μεταβλητή **duration** αποθηκεύει το χρόνο ταξιδιού των κυμάτων υπερήχων. Η μεταβλητή **trig** όταν γίνεται **HIGH** παράγει τον υπερήχο. Η συνάρτηση **PulseIn** λαμβάνει το χρόνο ταξιδιού του ηχητού κύματος. Με το RunningMedian, περιορίζουμε τα spikes του αισθητήρα.

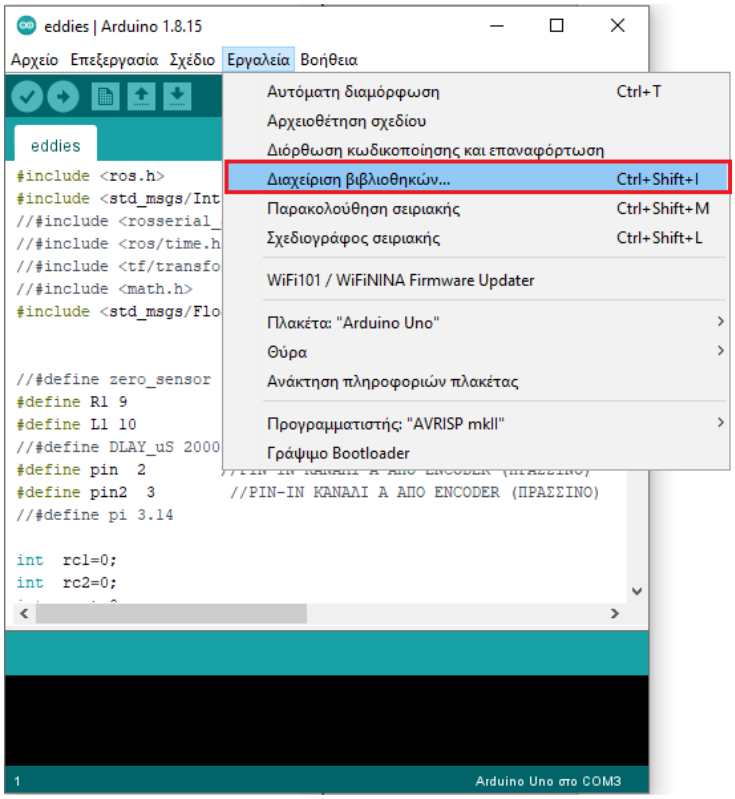
5.2.3 Συνδεσμολογία MPU-6050

Στον συγκεκριμένο αισθητήρα χρησιμοποιώ τις θέσεις SCL και SDA για επικοινωνία I2C με το ESP32.

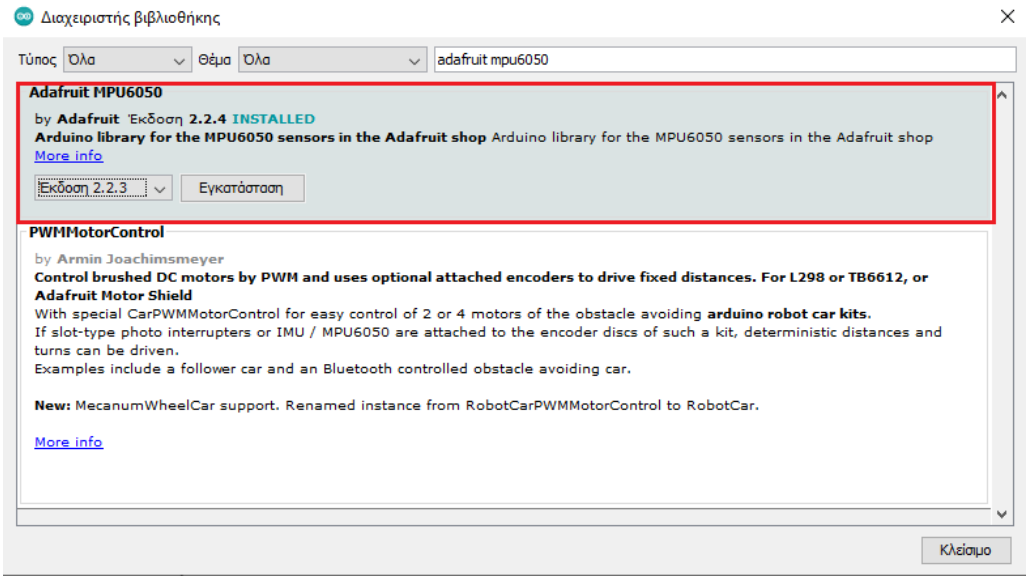


Εικόνα 39. Συνδεσμολογία MPU-6050

Χρειάζεται να εγκαταστήσουμε κάποιες βιβλιοθήκες στο Arduino IDE για να λειτουργήσει το MPU-6050 με το ESP32. Ξεκινώντας πάμε Εργαλεία ➡ Διαχείριση βιβλιοθηκών.

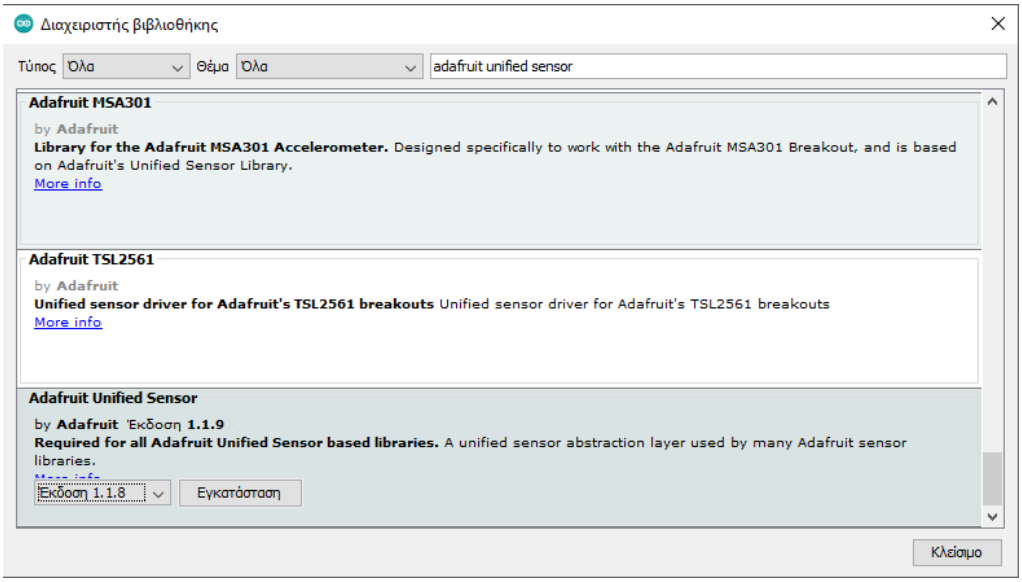


Κάνουμε αναζήτηση “Adafruit MPU6050” και κάνουμε εγκατάσταση.



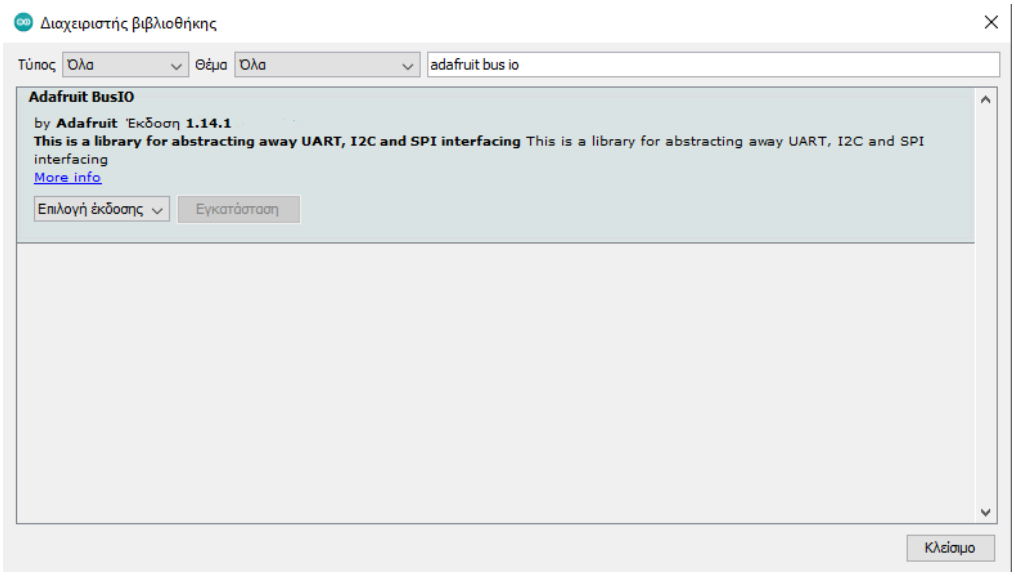
Εικόνα 40. βιβλιοθήκη Adafruit MPU-6050

Έπειτα κάνουμε αναζήτηση “Adafruit Unified Sensor” και το εγκαθιστούμε.



Εικόνα 41.Βιβλιοθήκη Adafruit unified sensor

Τέλος, κάνουμε αναζήτηση "Adafruit Bus IO"και πατάμε εγκατάσταση.



Εικόνα 42. Βιβλιοθήκα Adafruit BusIO

- Κώδικας Arduino:

```
Adafruit_MPU6050 mpu;

void setup() {

 if (!mpu.begin()) {
    Serial.println("Failed to find MPU6050 chip");
    while (1) {
      delay(10);
    }
  }
  Serial.println("MPU6050 found!");

 mpu.setAccelerometerRange(MPU6050_RANGE_8_G);
 mpu.setGyroRange(MPU6050_RANGE_500_DEG);
 mpu.setFilterBandwidth(MPU6050_BAND_5_HZ);
}

 void loop() {
   sensors_event_t a, g, temp;
   mpu.getEvent(&a, &g, &temp);

   double Ax, Ay, Az, T, Gx, Gy, Gz;

   Ax = a.acceleration.x - 0.67;
   Ay = a.acceleration.y + 0.52;
   Az = a.acceleration.z + 1;
   T = temp.temperature / 340 + 36.53;
   Gx = g.gyro.x + 0.01;
   Gy = g.gyro.y - 0.02;
   Gz = g.gyro.z;
```

Η εντολή **Adafruit_MPU6050 mpu**, είναι ένα αντικείμενο το οποίο καλείται για τον χειρισμό του αισθητήρα. Μετά κάνουμε αρχικοποίηση του αισθητήρα και έπειτα ορίζουμε το εύρος μέτρησης του επιταχυνσιόμετρου,το εύρος μέτρησης γυροσκοπίου και το εύρος ζώνης του φίλτρου. Έπειτα, πρέπει να λάβουμε τα νέα συμβάντα του αισθητήρα με τις τρέχουσες μετρήσεις. Τέλος, λαμβάνουμε τις μετρήσεις του αισθητήρα.

5.3. Παραμετροποίηση Συστήματος

5.3.1 Εγκατάσταση πακέτων

Ξεκινώντας, πρέπει να γίνει η εγκατάσταση κάποιων πακέτων ROS, που θα βοηθήσουν στην λειτουργία του συστήματος. Θα φτιάξουμε το δικό μας πακέτο, έτσι ώστε να έχουμε σε ένα χώρο όλα τα απαίτητα αρχεία που χρειαζόμαστε. Θα ανοίξουμε τη γραμμή εντολών και θα γράψουμε την ακόλουθη εντολή:

```
nikos@nikospc:~$ cd catkin_ws/src
nikos@nikospc:~/catkin_ws/src$
```

Εικόνα 43. Εντολή cd (Change Directory)

Με αυτή την εντολή αλλάζουμε directory σε χώρο εργασίας catkin.Μετά χρησιμοποιούμε την εντολή catkin_create_pkg, για να δημιουργήσουμε το πακέτο my_robot.

```
nikos@nikospc:~/catkin_ws/src$ catkin_create_pkg my_robot std_msgs rospy roscpp
```

Τέλος, αλλάζουμε πάλι directory από το src στο catkin_ws και τρέχουμε την εντολή catkin_make για να κάνουμε debug και να χτίσουμε το πακέτο.

```
nikos@nikospc:~$ cd catkin_ws/
nikos@nikospc:~/catkin_ws$
```

Εικόνα 44. Αλλαγή directory

```
nikos@nikospc:~/catkin_ws$ catkin_make
Base path: /home/nikos/catkin_ws
Source space: /home/nikos/catkin_ws/src
Build space: /home/nikos/catkin_ws/build
Devel space: /home/nikos/catkin_ws/devel
Install space: /home/nikos/catkin_ws/install
####
#### Running command: "make cmake_check_build_system" in "/home/nikos/catkin_ws/
build"
####
####
#### Running command: "make -j4 -l4" in "/home/nikos/catkin_ws/build"
####
[ 50%] Built target ekf_odom_pub
[ 50%] Built target imu_publisher
[ 75%] Built target imu_test
[100%] Built target rviz_click_to_2d
nikos@nikospc:~/catkin_ws$
```

Εικόνα 45. Εντολή catkin_make

Έπειτα, θα περιγράψουμε την εγκατάσταση όλων το πακέτων που θα χρησιμοποιήσουμε. Με την εντολή sudo apt install, κάνουμε εγκατάσταση στα πακέτα.

- **Navigation**

```
nikos@nikospc:~$ sudo apt install ros-noetic-navigation
[sudo] password for nikos:
Reading package lists... Done
Building dependency tree
Reading state information... Done
ros-noetic-navigation is already the newest version (1.17.3-1focal.20230216.0215
27).
The following packages were automatically installed and are no longer required:
  chromium-codecs-ffmpeg-extra gir1.2-goa-1.0 gstreamer1.0-vaapi
  libfwupdplugin1 libgstreamer-plugins-bad1.0-0 libopts25 libva-wayland2
  libxmlb1 sntp
Use 'sudo apt autoremove' to remove them.
0 upgraded, 0 newly installed, 0 to remove and 3 not upgraded.
```

Εικόνα 46. Εγκατάσταση πακέτου Navigation.

- **DWA Local Planner**

```
nikos@nikospc:~$ sudo apt install ros-noetic-dwa-local-planner
Reading package lists... Done
Building dependency tree
Reading state information... Done
ros-noetic-dwa-local-planner is already the newest version (1.17.3-1focal.202302
16.015538).
The following packages were automatically installed and are no longer required:
  chromium-codecs-ffmpeg-extra gir1.2-goa-1.0 gstreamer1.0-vaapi
  libfwupdplugin1 libgstreamer-plugins-bad1.0-0 libopts25 libva-wayland2
  libxmlb1 sntp
Use 'sudo apt autoremove' to remove them.
0 upgraded, 0 newly installed, 0 to remove and 3 not upgraded.
```

Εικόνα 47.Εγκατάσταση DWA Local Planner

- **GMapping**

```
nikos@nikospc:~$ sudo apt install ros-noetic-gmapping
Reading package lists... Done
Building dependency tree
Reading state information... Done
ros-noetic-gmapping is already the newest version (1.4.2-1focal.20230215.232030)
.
The following packages were automatically installed and are no longer required:
  chromium-codecs-ffmpeg-extra gir1.2-goa-1.0 gstreamer1.0-vaapi
  libfwupdplugin1 libgstreamer-plugins-bad1.0-0 libopts25 libva-wayland2
  libxmlb1 sntp
Use 'sudo apt autoremove' to remove them.
0 upgraded, 0 newly installed, 0 to remove and 3 not upgraded.
```

Εικόνα 48.Εγκατάσταση GMapping

- **IMU Complementary Filter**

```
nikos@nikospc:~$ sudo apt install ros-noetic-imu-complementary-filter
Reading package lists... Done
Building dependency tree
Reading state information... Done
ros-noetic-imu-complementary-filter is already the newest version (1.2.5-1focal.
20230215.232108).
The following packages were automatically installed and are no longer required:
  chromium-codecs-ffmpeg-extra gir1.2-goa-1.0 gstreamer1.0-vaapi
  libfwupdplugin1 libgstreamer-plugins-bad1.0-0 libopts25 libva-wayland2
  libxmlb1 sntp
Use 'sudo apt autoremove' to remove them.
0 upgraded, 0 newly installed, 0 to remove and 3 not upgraded.
```

Εικόνα 49.Εγκατάσταση IMU Complementary Filter

- **IMU Madgwick Filter**

```
nikos@nikospc:~$ sudo apt install ros-noetic-imu-filter-madgwick
Reading package lists... Done
Building dependency tree
Reading state information... Done
ros-noetic-imu-filter-madgwick is already the newest version (1.2.5-1focal.20230
215.231615).
The following packages were automatically installed and are no longer required:
  chromium-codecs-ffmpeg-extra gir1.2-goa-1.0 gstreamer1.0-vaapi
  libfwupdplugin1 libgstreamer-plugins-bad1.0-0 libopts25 libva-wayland2
  libxmlb1 sntp
Use 'sudo apt autoremove' to remove them.
0 upgraded, 0 newly installed, 0 to remove and 3 not upgraded.
```

Εικόνα 50. Εγκατάσταση IMU Madgwick filter

- **IMU tools**

```
nikos@nikospc:~$ sudo apt install ros-noetic-imu-tools
Reading package lists... Done
Building dependency tree
Reading state information... Done
ros-noetic-imu-tools is already the newest version (1.2.5-1focal.20230330.132550
).
The following packages were automatically installed and are no longer required:
  chromium-codecs-ffmpeg-extra gir1.2-goa-1.0 gstreamer1.0-vaapi
  libfwupdplugin1 libgstreamer-plugins-bad1.0-0 libopts25 libva-wayland2
  libxmlb1 sntp
Use 'sudo apt autoremove' to remove them.
0 upgraded, 0 newly installed, 0 to remove and 3 not upgraded.
```

Εικόνα 51.Εγκατάσταση IMU tools

- **Teleop Twist Keyboard**

```
nikos@nikospc:~$ sudo apt install ros-noetic-teleop-twist-keyboard
Reading package lists... Done
Building dependency tree
Reading state information... Done
ros-noetic-teleop-twist-keyboard is already the newest version (1.0.0-1focal.202
30215.213345).
The following packages were automatically installed and are no longer required:
  chromium-codecs-ffmpeg-extra gir1.2-goa-1.0 gstreamer1.0-vaapi
  libfwupdplugin1 libgstreamer-plugins-bad1.0-0 libopts25 libva-wayland2
  libxmlb1 sntp
Use 'sudo apt autoremove' to remove them.
0 upgraded, 0 newly installed, 0 to remove and 3 not upgraded.
```

Εικόνα 52.Εγκατάσταση Teleop Twist Keyboard

- **Robot Pose EKF**

```
nikos@nikospc:~$ sudo apt install ros-noetic-robot-pose-ekf
Reading package lists... Done
Building dependency tree
Reading state information... Done
ros-noetic-robot-pose-ekf is already the newest version (1.15.0-2focal.20230215.
232423).
The following packages were automatically installed and are no longer required:
  chromium-codecs-ffmpeg-extra gir1.2-goa-1.0 gstreamer1.0-vaapi
  libfwupdplugin1 libgstreamer-plugins-bad1.0-0 libopts25 libva-wayland2
  libxmlb1 sntp
Use 'sudo apt autoremove' to remove them.
0 upgraded, 0 newly installed, 0 to remove and 3 not upgraded.
```

Εικόνα 53. Εγκατάσταση Robot Pose EKF

5.3.2 Κώδικας Arduino IDE

- **Σύνδεση ESP32 με το ROS**

Μετά την εγκατάσταση των απαιτούμενων πακέτων, θα ανέβασουμε στο ESP32, τον κώδικα για το Rosserial για να επιτύχουμε τη σύνδεση με τον διακομιστή, δηλαδή τον υπολογιστή μας που έχει το λογισμικό ROS.

```
#define ROSSERIAL_ARDUINO_TCP
#include <WiFi.h>
#include <ros.h>

//Router Settings
const char* ssid = "Zyxel_F065B1";
const char* password = "Olympiakos.7";

//Server Settings
IPAddress server(192,168,1,201);
const uint16_t serverPort = 11411;

void setup() {
node.getHardware()->setConnection(server, serverPort);

}

void loop() {
  // put your main code here, to run repeatedly:

}


void setupWifi() {
  Serial.println();
  Serial.print("Connecting to ");
  Serial.println(ssid);

  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  Serial.println("");
  Serial.println("WiFi connected");
  Serial.println("IP address: ");
  Serial.println(WiFi.localIP());
}
```

Εικόνα 54. Κώδικας του Rosserial Arduino

Αρχικά φορτώνουμε τις βιβλιοθήκες, έπειτα βάζουμε τις ρυθμίσεις του Router και του Server, η διευθυνση του Server, είναι η ίδια με τη διεύθυνση του υπολογιστή που έχουμε το ROS. Στη συνάρτηση setupWifi() το ESP32 συνδέεται με το ίντερνετ. Με την παρακάτω εντολή επιτυγχάνεται η επικοινωνία με το ROS και τα μηνύματα του ESP32 προωθούνται στο ROS.

```
nikos@nikospc:~$ rosrun rosserial_python serial_node.py tcp
```

- **Οδομετρία**

Μετά την επίτευξη της επικοινωνίας μεταξύ ESP32 και ROS, θα γράψω τον απαραίτητο κώδικα για τη λειτουργία του οχήματος, ξεκινώντας με την οδομετρία.

```
#include <ros.h>

//Θέσεις στο ESP32
int r_enc = 19;
int l_enc = 18;

//Μετρητής διακοπών
int r_interruptCounter = 0;
int l_interruptCounter = 0;

//Δημιουργία Publisher για να σταλούν τα μηνύματα στο ROS
ros:: NodeHandle node;
std_msgs::Int16 lwheel_msg;
std_msgs::Int16 rwheel_msg;
ros::Publisher lwheel_pub("lwheel", &lwheel_msg);
ros::Publisher rwheel_pub("rwheel", &rwheel_msg);

void setup(){

node.initNode();
node.advertise(lwheel_pub);
node.advertise(rwheel_pub);
}

void setupPins() {

//Δημιουργία διακοπών
  pinMode(r_enc, INPUT_PULLUP);
  attachInterrupt(digitalPinToInterrupt(r_enc), r_handleInterrupt, RISING);
  pinMode(l_enc, INPUT_PULLUP);
  attachInterrupt(digitalPinToInterrupt(l_enc), l_handleInterrupt, RISING);
}

void loop(){
 //Οι μετρήσεις στέλνονται στο ROS
 lwheel_msg.data = l_interruptCounter;
 rwheel_msg.data = r_interruptCounter;
 lwheel_pub.publish(&lwheel_msg);
 rwheel_pub.publish(&rwheel_msg);
  node.spinOnce();
  delay(20);
}

//Μετρητές διακοπών/ticks
void r_handleInterrupt() {
  r_interruptCounter++;
}

void l_handleInterrupt() {
  l_interruptCounter++;
}
```

Εικόνα 55 Κώδικας μετρητή ταχύτητας

Με τον παραπάνω κώδικα θέτουμε σε λειτουργία τη μέτρηση της ταχύτητας. Κάθε φορά που ενεργοποιείται ο αισθητήρας ταχύτητας, το πρόγραμμα θέτει σε λειτουργία τη διακοπή, με τη σειρά της, αυξάνει το μετρήτη που έχουμε ορίσει κατά ένα και κάθε φορά οι μετρήσεις στέλνονται με τους Publishers που έχουμε δημιουργήσει στο ROS, το οποίο με τη σειρά του μετατρέπει τις τιμές αυτές σε οδομετρία, δηλάδη ορίζει αν το όχημα κινείται μπροστά, πίσω, αριστερά ή δεξιά.

- **IMU-MPU 6050**

Παίρνοντας τις τιμές απο την MPU 6050, κατασκεύασα το μήνυμα για να σταλούν οι τιμές στο ROS.

```
#include <ros.h>
#include <sensor_msgs/Imu.h>

//Δημιουργία Publisher για την IMU
sensor_msgs::Imu msg;
ros::Publisher IMU("imu/data_raw", &msg);

void setup() {
  Serial.begin(115200);
  node.initNode();
  node.advertise(IMU);
}


void loop() {

  //Αποστολή τιμών γωνιακής ταχύτητας στο ROS
  msg.angular_velocity.x = Gx;
  msg.angular_velocity.y = Gy;
  msg.angular_velocity.z = Gz;

 //Αποστολή γραμμικής επιτάχυνσης
  msg.linear_acceleration.x = Ax;
  msg.linear_acceleration.y = Ay;
  msg.linear_acceleration.z = Az;

  msg.header.frame_id = "imu";
  msg.header.stamp=node.now();
  IMU.publish(&msg);
  node.spinOnce();
  delay(20);
  }
```

Εικόνα 56. Κώδικας IMU

Για αρχή, δημιουργούμε τον Publisher της IMU ο οποίος είναι υπεύθυνος για να σταλούν οι τιμές στο ROS.Μετά τα μηνύματα της γωνιακής ταχύτητας και της γραμμικής επιτάχυνσης ισούνται με τις κατάλληλες τιμές της MPU6050 και στέλνονται στο ROS.

- **Αισθητήρες υπερήχων**

Στη συνέχεια δημιουργώ το μήνυμα για να στείλουμε τις τιμές των αισθητήρων υπερήχων, για να γνωρίζει το όχημα κάθε στιγμή την απόσταση του από τα εμπόδια.

```
#include <ros.h>
#include <sensor_msgs/LaserScan.h>



float distance0, distance1,distance2,distance3,distance4;

//Δημιουργία Publisher για τους αισθητήρες υπερήχων
sensor_msgs::LaserScan laser_msg;
ros::Publisher laser("scan", &laser_msg);


ros:: NodeHandle node;

void setup() {
  Serial.begin(115200);
  node.initNode();
  node.advertise(laser);
}


void loop() {

//Διάβασμα Αισθητήρων
  distance0 = readSonar(4, 16);
  distance1 = readSonar(27, 14);
  distance2 = readSonar(25, 33);
  distance3 = readSonar(12,34);
  distance4 = readSonar(26,35);

  // Δημιουργία μηνύματος
  laser_msg.header.stamp=node.now();
  laser_msg.header.frame_id = "sonar_link";
  //Ορίζουμε τη γωνία σάρωσης
  laser_msg.angle_min = -1.57;
  laser_msg.angle_max = 1.57;
  //Γωνία μεταξύ των αισθητήρων
  laser_msg.angle_increment = 0.785398;
  laser_msg.scan_time= 0.2;
  //Ελάχιστη τιμή μέτρησης αισθητήρα
  laser_msg.range_min = 0.02;
  //Μέγιστη τιμή μέτρησης αισθητήρα
  laser_msg.range_max = 3.00;
  //Δεδομένα μέτρησης
  float ranges[] = {distance0,distance4,distance1,distance3,distance2};
  laser_msg.ranges = ranges;
  laser_msg.ranges_length = 5;
  float intensities[] = {100.0,100.0,100.0,100.0,100,0};
  laser_msg.intensities = intensities;
  laser_msg.intensities_length = 5;
  laser_msg.time_increment = 0.0;

  // Publish message
  laser.publish(&laser_msg);

  node.spinOnce();
  delay(20);
  }
```

Εικόνα 57. Κώδικας για Μήνυμα Laserscan

Στη συνάρτηση loop, γίνεται το διάβασμα των πέντε αισθητήρων, μετά γίνεται η δημιουργία του μηνύματος laserscan, με το οποίο στέλνονται οι τιμές στο ROS.

- **Μήνυμα ROS για κίνηση τροχών**

Το τελευταίο κομμάτι του κώδικα του Arduino IDE, αφορά τη κίνηση των τροχών, τη δημιουργία ενός μηνύματος Twist και ενος Subscriber που θα δέχεται το μήνυμα αυτό ώστε να κινούνται οι τροχοί στη κατεύθυνση που θέλουμε.

```
#include <math.h>
#include <ros.h>
#include <geometry_msgs/Twist.h>

void setupPins();
void setupWifi();
void onTwist(const geometry_msgs::Twist &msg);

float mapPwm(const float x,const float in_min,const float in_max,const float out_min,const float out_max);

ros:: NodeHandle node;
//Δημιουργία Subcriber για να λάβει μηνύματα Twist
ros::Subscriber<geometry_msgs::Twist>sub("/cmd_vel", &onTwist);

void setup() {

  setupPins();
  Serial.begin(115200);
  node.initNode();
}

void setupPins() {
  pinMode(L_PWM, OUTPUT);
  pinMode(L_FORW, OUTPUT);
  pinMode(L_BACK, OUTPUT);
  pinMode(R_PWM, OUTPUT);
  pinMode(R_FORW, OUTPUT);
  pinMode(R_BACK, OUTPUT);
  stop();
}
```

Το μήνυμα Twist αποτελείται από δύο διανύσματα, το ένα ονομάζεται γραμμικό και το άλλο γωνιακό. Και τα δύο περιέχουν X, Y, Z μεταβλητές. Για την κίνηση του οχήματος χρησιμοποιείται η X μεταβλητή του γραμμικού διανύσματος και η Z μεταβλητή του γωνιακού διανύσματος. Για παράδειμα, αν δώσουμε θετική τιμή στη μεταβλητή X του γραμμικού διανύσματος, τότε το όχημα θα πάει μπροστά, αν δώσουμε αρνητική θα πάει πίσω. Όσο αφορά το γωνιακό διάνυσμα, αν το Z λάβει θετική τιμή τότε το όχημα θα στρίψει αριστερά, ενώ αν λάβει αρνητική θα στρίψει δεξιά.Αυτό συμβαίνει γιατί εφαρμόζεται ο κανόνας του δεξιού χεριού.

```
void onTwist(const geometry_msgs::Twist &msg) {
//Μέγιστες τιμές -1 και 1
  float x = max(min(msg.linear.x, 1.0f), -1.0f);
  float z = max(min(msg.angular.z, 1.0f), -1.0f);

//Εξίσωση κινηματικής για διαφορική κίνηση
  float l = (msg.linear.x - msg.angular.z) / 2;
  float r = (msg.linear.x + msg.angular.z) / 2;

//Μετατροπή σε PWM για ορισμό ταχύτητας
  uint16_t lPwm = mapPwm(fabs(l),0,1,105,255);
  uint16_t rPwm = mapPwm(fabs(r),0,1,105,255);

  Serial.print("Pwms : ");
  Serial.print(lPwm);
  Serial.println(rPwm);

   // stop
  if (l == 0 && r == 0) {
    digitalWrite(L_FORW, l = 0);
    digitalWrite(L_BACK, l = 0);
    digitalWrite(R_FORW, r = 0);
    digitalWrite(R_BACK, r = 0);
    analogWrite(L_PWM, lPwm);
    analogWrite(R_PWM, rPwm);
  }

   // κίνηση οχήματος μπροστά
  if (l > 0 && r > 0) {
    digitalWrite(L_FORW, l > 0);
    digitalWrite(L_BACK, l < 0);
    digitalWrite(R_FORW, r > 0);
    digitalWrite(R_BACK, r < 0);
    analogWrite(L_PWM, lPwm);
    analogWrite(R_PWM, rPwm);
  }

    // Κίνηση οχήματος αριστερά
  if (l < 0 && r > 0) {
    digitalWrite(L_FORW, l < 0);
    digitalWrite(L_BACK, l < 0);
    digitalWrite(R_FORW, r > 0);
    digitalWrite(R_BACK, r < 0);
    analogWrite(L_PWM, lPwm);
    analogWrite(R_PWM, rPwm);
  }

   //Κίνηση οχήματος δεξιά
  if (l > 0 && r < 0) {
    digitalWrite(L_FORW, l > 0);
    digitalWrite(L_BACK, l < 0);
    digitalWrite(R_FORW, r < 0);
    digitalWrite(R_BACK, r < 0);
    analogWrite(L_PWM, lPwm);
    analogWrite(R_PWM, rPwm);
  }
}

void loop() {

  node.spinOnce();
  delay(20);
}

float mapPwm(const float x,const float in_min,const float in_max, const float out_min, const float out_max) {
  return (x-in_min) * (out_max - out_min) / (in_max-in_min)+out_min;
}
```

5.3.3 URDF και αρχείο Launch

- **URDF**

Το Unified Robotics Description Format ή στα Ελληνικά, Ενοποιημένη Μορφή Περιγραφής Ρομποτικής, είναι ένα αρχείο σε μορφή XML που περιλαμβάνει τη φυσική περιγραφή ενός ρομπότ. Είναι τρισδιάστατο μοντέλο που περιέχει πληροφορίες για τις αρθρώσεις, για τον κινητήρα, μάζα και άλλα. Το URDF χρησιμοποιείται για την πραγματικού χρόνου παρακολούθηση, για να γνωρίζει ο χειριστής τι κάνει το ρομπότ.

```
1 <?xml version="1.0"?>
2 <robot xmlns:xacro="http://www.ros.org/wiki/xacro" name="kbot">
3
4   <xacro:property name="base_width" value="0.11"/>
5   <xacro:property name="base_len" value="0.21"/>
6   <xacro:property name="wheel_radius" value="0.035"/>
7   <xacro:property name="base_wheel_gap" value="0.007"/>
8   <xacro:property name="wheel_separation" value="0.15"/>
9   <xacro:property name="wheel_joint_offset" value="0.02"/>
10
11  <xacro:property name="caster_wheel_radius" value="${wheel_radius/2}"/>
12  <xacro:property name="caster_wheel_mass" value="0.001"/>
13  <xacro:property name="caster_wheel_joint_offset" value="-0.052"/>
14
15  <!--Color Properties-->
16  <material name="blue">
17    <color rgba="0 0 0.8 1"/>
18  </material>
19  <material name="black">
20    <color rgba="0 0 0 1"/>
21  </material>
22  <material name="white">
23    <color rgba="1 1 1 1"/>
24  </material>
25  <material name="red">
26    <color rgba="0.8 0.0 0.0 1.0"/>
27  </material>
28
```

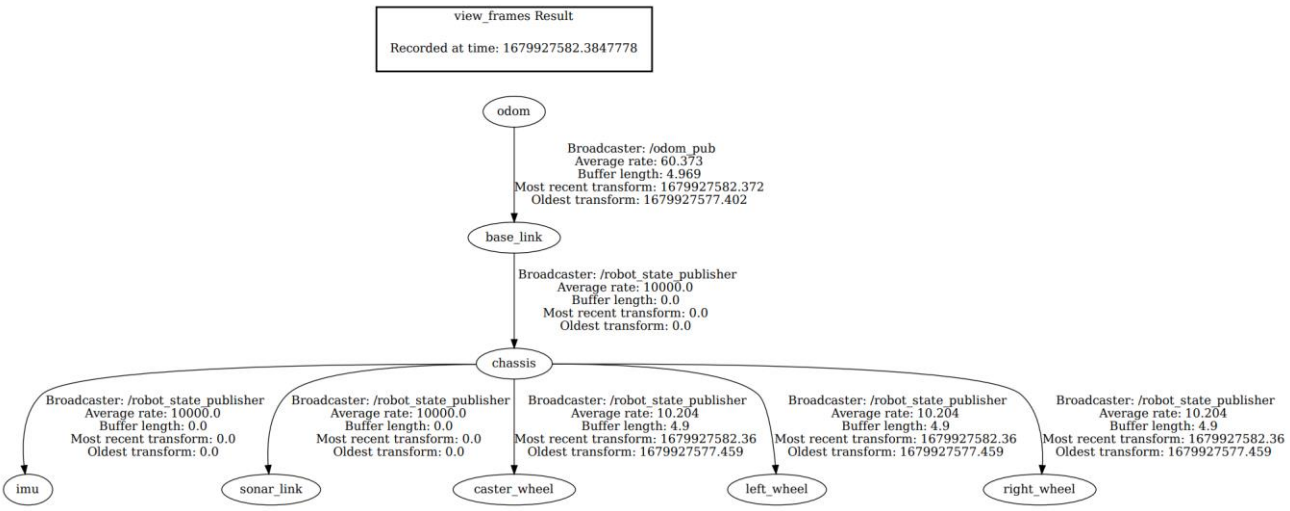
Εικόνα 58. Κώδικας URDF του οχήματος

Στην εικόνα 58, περιέχει τις βασικές πληροφορίες του οχήματος, όπως επίσης και τι χρώματα θα χρησιμοποιήσουμε για να αναπαραστήσουμε τα διάφορα μέρη του οχήματος. Στην εικόνα 59, περιγράφουμε τι μέγεθος θα έχει το σασί στην αναπαράσταση μας, τι γεωμετρικό σχήμα θα έχει και τι θέση θα έχει στο χώρο. Με αυτόν τον τρόπο περιγράφονται όλα τα μέρη του οχήματος.

```
30  <!--Base Link-->
31  <link name="base_link">
32    <visual>
33        <origin xyz="0 0 0" rpy="0 0 0" />
34        <geometry>
35            <box size="0.001 0.001 0.001" />
36        </geometry>
37    </visual>
38  </link>
39
40  <!--chassis-->
41  <link name="chassis">
42    <visual>
43      <geometry>
44        <box size="${base_len} ${base_width} 0.01"/>
45      </geometry>
46      <material name="black"/>
47    </visual>
48    <collision>
49      <geometry>
50        <box size="${base_len} ${base_width} 0.01"/>
51      </geometry>
52    </collision>
53  </link>
54
55  <!--base_link to base_footprint Joint-->
56  <joint name="base_link_joint" type="fixed">
57    <origin xyz="0 0 ${wheel_radius + 0.005}" rpy="0 0 0" />
58    <parent link="base_link"/>
59    <child link="chassis" />
60  </joint>
```

Εικόνα 59. Κώδικας URDF του οχήματος

Στην εικόνα 60, βλέπουμε το διάγραμμα που έχει δημιουργήθει από τη περιγραφή των μερών του οχήματος. Βλέπουμε το σασί και πως οι τροχοί, οι αισθητήρες και η IMU είναι τοποθετημένοι πάνω του.



Εικόνα 60. Διάγραμμα URDF του οχήματος

- **Αρχείο Launch**

Ένα αρχείο launch, εκτελείται από το ROS με την εντολή Roslaunch, το αρχείο αυτό χρησιμοποιείται διότι μπορούμε να ξεκινήσουμε πολλαπλούς κόμβους μαζί, τον κόμβο master, όπως επίσης να θέσουμε παραμέτρους σε έναν κόμβο.

```
<?xml version="1.0"?>
<launch>

 <!--Έναρξη Master node -->
 <master auto="start"/>

 <!--Έναρξη κόμβου για φόρτωση μοντέλου URDF -->
<arg name="model" default="$(find my_robot)/urdf/new_robot.urdf"/>
<param name="robot_description" command="$(find xacro)/xacro --inorder $(arg model)"/>


<node name="joint_state_publisher" pkg="joint_state_publisher" type="joint_state_publisher">
 <param name="use_gui" value="False"/>
</node>

<node name="robot_state_publisher" pkg="robot_state_publisher" type="robot_state_publisher"/>

<!--Έναρξη κόμβου για επικοινωνία ESP32 με το ROS -->
<node pkg="rosserial_python" name="wifi_ep32_communication_node" type="serial_node.py" args="tcp" output="screen"/>

 <!--Έναρξη κόμβου για την οδομετρία -->
<node pkg="my_robot" type="diff_tf_odom.py" name="odom_pub">
  </node>

<!--Έναρξη κόμβου για το φίλτρο Madgwick για την IMU -->
 <node pkg="imu_filter_madgwick" name="filter_madgwick" type="imu_filter_node">
      <param name="use_mag" value="false"/>
      <param name="publish_tf" value="false"/>
</node>

<!--Έναρξη κόμβου για το RVIZ -->
<node pkg="rviz" type="rviz" name="rviz">
 </node>

 <!--Έναρξη κόμβου για καταχώριση συντεταγμένων μέσω του RVIZ -->
 <node pkg="navstack_pub" type="rviz_click_to_2d" name="rviz_click_to_2d">
 </node>

 <arg name="scan_topic" default="/scan" />

<!--Έναρξη κόμβου Move Base, ο κόμβος που ευθύνεται για τη πλοήγηση του οχήματος -->
<node pkg="move_base" type="move_base" respawn="false" name="move_base" output="screen">
   <rosparam file="$(find navstack_pub)/param/spatia_costmap_common_params.yaml" command="load" ns="global_costmap"/>
   <rosparam file="$(find navstack_pub)/param/spatia_costmap_common_params.yaml" command="load" ns="local_costmap"/>
   <rosparam file="$(find navstack_pub)/param/odom_params/local_costmap_params.yaml" command="load" />
   <rosparam file="$(find navstack_pub)/param/odom_params/global_costmap_params.yaml" command="load"  />
   <rosparam file="$(find navstack_pub)/param/dwa_local_planner.yaml" command="load" />
   <rosparam file="$(find navstack_pub)/param/global_planner_params.yaml" command="load" />
   <rosparam file="$(find navstack_pub)/param/navfn_global_planner_params.yaml" command="load" />
   <rosparam file="$(find navstack_pub)/param/move_base_params.yaml" command="load" />
  </node>

</launch>
```

Εικόνα 61.Κώδικας αρχείου launch

Η μεταβλητή **pkg** αφορά το πακέτο που σχετίζεται με τον κόμβο που θα ξεκινήσει, η μεταβλητή **type** αφορά στο όνομα του εκτελέσιμου αρχείου του κόμβου. Η μεταβλητή **respawn** είναι προαιρετική, είναι σύνηθες να την έχουμε. Αν το **respawn=true,** τότε ο κόμβος θα επανεκκινηθεί αν για κάποιο λόγο έκλεισε. Η μεταβλητή **ns** ή namespace χρησιμοποιείται όταν έχουμε πολλαπλές περιπτώσεις του ίδιου κόμβου [17].

5.4 Κόμβος Move Base

Ο κόμβος Move base είναι εκείνος που κυρίως ευθύνεται για τη πλοήγηση του οχήματος.

```
<!--Έναρξη κόμβου Move Base, ο κόμβος που ευθύνεται για τη πλοήγηση του οχήματος -->
<node pkg="move_base" type="move_base" respawn="false" name="move_base" output="screen">
    <rosparam file="$(find navstack_pub)/param/spatia_costmap_common_params.yaml" command="load" ns="global_costmap"/>
    <rosparam file="$(find navstack_pub)/param/spatia_costmap_common_params.yaml" command="load" ns="local_costmap"/>
    <rosparam file="$(find navstack_pub)/param/odom_params/local_costmap_params.yaml" command="load" />
    <rosparam file="$(find navstack_pub)/param/odom_params/global_costmap_params.yaml" command="load"  />
    <rosparam file="$(find navstack_pub)/param/dwa_local_planner.yaml" command="load" />
    <rosparam file="$(find navstack_pub)/param/global_planner_params.yaml" command="load" />
    <rosparam file="$(find navstack_pub)/param/navfn_global_planner_params.yaml" command="load" />
    <rosparam file="$(find navstack_pub)/param/move_base_params.yaml" command="load" />
  </node>
```

Εικόνα 62.Κόμβος Move Base

- **Spatia Costmap Common Params**

Η στοίβα της πλοήγησης χρησιμοποιεί δύο χάρτες κόστους για να αποθηκεύσει πληροφορίες για τα εμπόδια στο κόσμο. Ο ένας χάρτης κόστους χρησιμοποιείται για παγκόσμιο σχεδιασμό, δηλαδή δημιουργεί μακροπρόθεσμα σχέδια για ολόκληρο το περιβάλλον, και το άλλο χρησιμοποιείται για τοπικό σχεδιασμό και αποφυγή εμποδίων. Ωστόσο, κάποιες επιλογές διαμόρφωσης μπορεί να είναι κοινές και για τους δύο χάρτες κόστους, έτσι ουσιαστικά δημιουργούνται τρία αρχεία: global costmap, local costmap, common costmap [18].

```
#Obstacle marking parameters
obstacle_range: 1.5
raytrace_range: 2.0

#The footprint of the robot
footprint: [[-0.13,-0.10],[-0.13,0.10],[0.13,0.10],[0.13,-0.10]]
footprint_padding: 0.05

robot_base_frame: base_link
robot_radius: 0.0

meter_scoring: true

plugins:
    - {name: rgbd_obstacle_layer, type: "spatio_temporal_voxel_layer/SpatioTemporalVoxelLayer"}
    - {name: inflater_layer, type: "costmap_2d::InflationLayer"}



inflater_layer:
 inflation_radius: 1.75
 cost_scaling_factor: 2.58


rgbd_obstacle_layer:
  enabled:                true
  voxel_decay:            0.55     #seconds if linear, e^n if exponential
  decay_model:            0       #0=linear, 1=exponential, -1=persistent
  voxel_size:             0.02   #meters
  track_unknown_space:    true   #default space is unknown
  observation_persistence: 0.0  #seconds
  max_obstacle_height:    1.0    #meters
  mark_threshold:         0      #voxel height
  update_footprint_enabled: true
  combination_method:     1      #1=max, 0=override
  obstacle_range:         1.5    #meters
  origin_z:               0.0    #meters
  publish_voxel_map:      true   # default off
  transform_tolerance:    0.8    # seconds
  mapping_mode:           false  # default off, saves map not for navigation
  map_save_duration:      60     #default 60s, how often to autosave
  observation_sources:    rgbd1_mark
  rgbd1_mark:
    data_type: LaserScan
    topic: /scan
    marking: true
    clearing: false
    #min_obstacle_height: 0.3     #default 0, meters
    #max_obstacle_height: 2.0     #defaule 3, meters
    expected_update_rate: 0.0    #default 0, if not updating at this rate at least, remove from buffer
    observation_persistence: 0.0 #default 0, use all measurements taken during now-value, 0=latest
    inf_is_valid: false          #default false, for laser scans
    clear_after_reading: true    #default false, clear the buffer after the layer gets readings from it
    filter: "voxel"              #default passthrough, apply "voxel", "passthrough", or no filter to sensor data
    voxel_min_points: 0          #default 0, minimum points per voxel for voxel filter
```

Εικόνα 63. Κώδικας Spatia Costmap Common Params

Obstacle range: Η μέγιστη απόσταση από το ρομπότ, στην οποία θα εισαχθεί ένα εμπόδιο στον χάρτη κόστους σε μέτρα.

Raytrace range: Καθορίζει το εύρος στο οποίο θα ανιχνεύσουμε τον ελεύθερο χώρο, δεδομένης της ένδειξης του αισθητήρα.

Footprint: Προδιαγραφές για το αποτύπωμα του ρομπότ.

Footprint padding: Επιπλέον "μαξιλαράκι" μεταξύ του footprint και των εμποδίων

Robot base frame: Το όνομα του πλαισίου για τον βασικό σύνδεσμο του ρομπότ.

Robot radius: Προδιαγραφή του ρομπότ αν είναι κυκλικό.

Meter scoring: Εάν οι παράμετροι goal_distance_bias και path_distance_bias θα πρέπει να υποθέσουν ότι το goal_distance και το path_distance εκφράζονται σε μονάδες μέτρων ή κελιων.

Inflation radius: Ελέγχει πόσο μακριά είναι το σημείο μηδενικού κόστους από το εμπόδιο.

Cost scaling factor: Ο συντελεστής κλιμάκωσης που εφαρμόζεται στις τιμές κόστους κατά τη διάρκεια του πληθωρισμού.

Decay model: Μοντέλο φθοράς.

Voxel size: Μέγεθος ογκοστοιχείου.

Track unknown space: Εάν είναι false, κάθε pixel έχει μία από τις δύο καταστάσεις: θανατηφόρο εμπόδιο ή ελεύθερο. Εάν είναι true, κάθε pixel έχει μια από τις τρείς καταστάσεις: θανατηφόρο εμπόδιο, ελεύθερο ή άγνωστο.

Max obstacle height: Το μέγιστο ύψος οποιουδήποτε εμποδίου που πρέπει να εισαχθεί στον χάρτη κόστους σε μέτρα.

Origin z: Η Z προέλευση του χάρτη σε μέτρα.

Publish voxel map: Αν θα δημοσιευτεί ή όχι το υποκείμενο πλέγμα ογκοστοιχείων για σκοπό οπτικοποίησης.

Transform tolerance: Ορίζει τη μέγιστη επιτρεπόμενη καθυστέρηση μεταξύ αυτών των μετασχηματισμών.

Data type: Τύπος θέματος, laserscan, Pointcloud.

Topic: Το θέμα στο οποίο έρχονται δεδομένα αισθητήρα για αυτήν τη πηγή.

Marking: Εάν αυτή η παρατήρηση πρέπει να χρησιμοποιείται για να επισημένονται εμπόδια.

Clearing: Εάν αυτή η παρατήρηση πρέπει να χρησιμοποιείται για την εκκαθάριση του ελεύθερου χώρου.

Inf is valid: Επιτρέπει τιμές inf σε μηνύματα παρατήρησης Laserscan.Οι τιμές inf μετατρέπονται στο μέγιστο εύρος λέιζερ.

Clear after reading: Καθαρίζει τον buffer αφού το layer λάβει μετρήσεις από αυτό.

Filter: Φίλτρο, Voxel, passthrough ή κανένα φίλτρο.

- Global Costmap

```
global_costmap:
  global_frame: odom
  robot_base_frame: base_link
  update_frequency: 10.0
  publish_frequency: 2.0
  width: 50.0
  height: 50.0
  resolution: 0.05
  origin_x: -25.0
  origin_y: -25.0
  rolling_window: true
  static_map: false
```

Εικόνα 64. Κώδικας Global Costmap

Global Frame: Το παγκόσμιο πλαίσιο για τη λειτουργία του χάρτη κόστους.

Update Frequency: Η συχνότητα σε Hz για την ενημέρωση του χάρτη.

Publish Frequency: Η συχνότητα σε Hz για τον χάρτη που πρόκειται να εμφανίζει πληροφορίες.

Width: Το πλάτος του χάρτη σε μέτρα.

Height: Το ύψος του χάρτη σε μέτρα.

Resolution: Η ανάλυση του χάρτη σε μέτρα/κελί.

Origin X: Η προέλευση X του χάρτη στο παγκόσμιο πλαίσιο σε μέτρα.

Origin Y: Η προέλευση Y του χάρτη στο παγκόσμιο πλαίσιο σε μέτρα.

Rolling window: Αν θα χρησιμοποιηθεί ή όχι μια έκδοση κυλιόμενου παραθύρου του χάρτη κόστους.

Static map: Ενσωματώνει ως επί το πλείστον αμετάβλητα δεδομένα από μια εξωτερική πηγή.

```
local_costmap:
  global_frame: odom
  robot_base_frame: base_link
  update_frequency: 10.0
  publish_frequency: 5.0
  rolling_window: true
  width: 3.0
  height: 3.0
  resolution: 0.05
  static_map: false
```

Εικόνα 65. Κώδικας Local Costmap

- **Move Base params**

Ο κόμβος move base παρέχει μια διεπαφή ROS για διαμόρφωση, εκτέλεση και αλληλεπίδραση με τη στοίβα πλοήγησης σε ένα ρομπότ.

```
shutdown_costmaps: false

controller_frequency: 10.0
controller_patience: 15.0

planner_frequency: 10.0
planner_patience: 15.0

conservative_reset_dist: 3.0
oscillation_timeout: 10.0
oscillation_distance: 0.2

recovery_behavior_enabled: true
clearing_rotation_allowed: false

base_local_planner: "dwa_local_planner/DWAPlannerROS"
base_global_planner: "navfn/NavfnROS"
```

Εικόνα 66. Κώδικας Move Base

Shutdown costmaps: Καθορίζει εάν θα τερματιστούν ή όχι οι χάρτες κόστους του κόμβου όταν το move base είναι σε ανενεργή κατάσταση.

Controller frequency: Ο ρυθμός σε Hz με τον οποίο εκτελείται ο βρόχος ελέγχου και στέλνονται οι εντολές ταχύτητας στη βάση.

Controller patience: Πόσο θα περιμένει ο ελεγκτής σε δευτερόλεπτα χωρίς να λάβει έγκυρο έλεγχο,προτού εκτελεστούν οι λειτουργίες εκκαθάρισης χώρου.

Planner frequency: Ο ρυθμός σε Hz με τον οποίο θα εκτελεστεί ο παγκόσμιος βρόχος προγραμματισμού. Εάν η συχνότητα έχει οριστεί σε μηδέν, ο παγκόσμιος σχεδιαστής θα εκτελεστεί μόνο όταν ληφθεί ένας νέος στόχος ή ο τοπικός σχεδιαστής αναφέρει ότι η διαδρομή του έχει αποκλειστεί.

Planner patience: Πόσο θα περιμένει ο σχεδιαστής σε δευτερόλεπτα,σε μια προσπάθεια να βρει ένα έγκυρο σχέδιο πριν εκτελεστούν οι εργασίες εκκαθάρισης χώρου.

Conservative reset dist: Η απόσταση από το ρομπότ σε μέτρα,πέρα από την οποία θα αφαιρεθούν τα εμπόδια από τον χάρτη κόστους κατά την προσπάθεια εκκαθάρισης χώρου στον χάρτη.

Oscillation timeout: Πόσος χρόνος σε δευτερόλεπτα για να επιτραπεί η ταλάντωση πριν από την εκτέλεση συμπεριφορών ανάκτησης.

Oscillation distance: Πόσο μακριά σε μέτρα πρέπει να κινηθεί το ρομπότ για να θεωρηθεί ότι δεν ταλαντώνεται.

Recovery behavior enabled: Εάν ενεργοποιηθούν ή όχι οι συμπεριφορές ανάκτησης move base για να προσπαθήσουν να εκκαθαρίσουν τον χώρο.

Clearing rotation allowed: Καθορίζει εάν το ρομπότ θα επιχειρήσει μια περιστροφή στη θέση του, κατά την προσπάθεια εκκαθάρισης του χώρου.

Base local planner: Το όνομα του πρόσθετου του τοπικού σχεδιαστή για χρήση με το move base.

Base global planner: : Το όνομα του πρόσθετου του παγκόσμιου σχεδιαστή για χρήση με το move base.

- **Navfn Global Planner Params**

Το navfn παρέχει μια λειτουργία γρήγορης παρεμβολής πλοήγησης που μπορεί να χρησιμοποιηθεί για τη δημιουργία σχεδίων για μια φορητή βάση.

```
NavfnROS:
  visualize_potential: false
  allow_unknown: false
  planner_window_x: 0.0
  planner_window_y: 0.0
  default_to_tolerance: 0.0
```

Εικόνα 67. Κώδικας Navfn Global Planner

Visualize potential: Καθορίζει εάν θα απεικονιστεί ή όχι πιθανή περιοχή που υπολογίζεται από το navfn μέσω ενός PointCloud2.

Allow unknown: Καθορίζει εάν θα επιτρέπεται ή όχι στο navfn να δημιουργεί σχέδια που διασχίζουν άγνωστο χώρο.

Planner window X: Καθορίζει το μέγεθος X ενός προαιρετικού παραθύρου στο οποίο περιορίζεται ο σχεδιαστής.

Planner window Y: Καθορίζει το μέγεθος Y ενός προαιρετικού παραθύρου στο οποίο περιορίζεται ο σχεδιαστής.

Default to tolerance: Μια ανοχή στο σημείου στόχου για τον σχεδιαστή. Το navfn θα προσπαθήσει να δημιουργήσει ένα σχέδιο που να είναι όσο το δυνατόν πιο κοντά στον καθορισμένο στόχο, αλλά όχι πιο μακριά από τη μεταβλητή default tolerance.

- Global Planner params

Αυτό το πακέτο παρέχει μια εφαρμογή ενός γρήγορου, παρεμβαλλόμενου παγκόσμιου σχεδιαστή για πλοήγηση.

```
GlobalPlanner:
  old_navfn_behavior: false
  use_quadratic: true
  use_dijkstra: true
  use_grid_path: false
  allow_unknown: true
  planner_window_x: 0.0
  planner_window_y: 0.0
  default_tolerance: 0.0
  publish_scale: 100
  planner_costmap_publish_frequency: 0.0
  lethal_cost: 253
  neutral_cost: 66
  cost_factor: 0.55
  publish_potential: true
```

Εικόνα 68. Κώδικας Global Planner

Old navfn behavior: Εάν θέλουμε τον παγκόσμιο σχεδιαστή να αντικατοπτρίζει ακριβώς τη συμπεριφορά του nafn.

Use quadratic: Εάν είναι true, χρησιμοποιεί την τετραγωνική προσέγγιση του δυναμικού.

Use Dijkstra: Εάν είναι true, χρησιμοποιεί τον αλγόριθμου του Dijkstra.

Use grid path: Εάν είναι true, δημιουργεί μια διαδρομή που ακολουθεί τα όρια του πλέγματος. Διαφορετικά, χρησιμοποιεί μια μέθοδο gradient descent.

Lethal cost: Θανατηφόρο κόστος.

Neutral cost: Ουδέτερο κόστος.

Cost factor: Συντελεστής για να πολλαπλασιαστεί κάθε κόστος από τον χάρτη κόστους επί.

Publish potential: Δημοσίευση δυνητικού χάρτη κόστους.

- **DWA Local Planner**

Το πακέτο παρέχει μια εφαρμογή της δυναμικής προσέγγισης παραθύρου στην τοπική πλοήγηση του ρομπότ σε μια επίπεδη επιφάνεια. Δεδομένου ενός παγκόσμιου σχεδίου που πρέπει να ακολουθηθεί και ενός χάρτη κόστους, ο τοπικός σχεδιαστής παράγει εντολές ταχύτητας για αποστολή σε μια κινητή βάση.

```
DWAPlannerROS:
  # Robot configuration parameters
  acc_lim_x: 10.0
  acc_lim_th: 10.0

  max_vel_x: 1.0
  min_vel_x: 0.0

  max_vel_trans: 1.0
  min_vel_trans: 0.2

  max_vel_theta: 0.3
  min_vel_theta: 0.0
  min_in_place_vel_theta: 0.314
  holonomic_robot: false
  escape_vel: -0.5

  # Goal Tolerance Parameters
  yaw_goal_tolerance: 0.157
  xy_goal_tolerance: 0.25
  latch_xy_goal_tolerance: true

  # # Forward Simulation Parameters
  sim_time: 0.8
  sim_granularity: 0.04
  vx_samples: 6
  vtheta_samples: 20
  controller_frequency: 10.0

  # # Trajectory scoring parameters
  path_distance_bias: 64.0 # The weighting for how much the controller should stay close to the path it was given
  goal_distance_bias: 20.0 # The weighting for how much the controller should attempt to reach its local goal, also controls speed
  occdist_scale: 0.02 # The weighting for how much the controller should attempt to avoid obstacles
  forward_point_distance: 0.325
  stop_time_buffer: 0.2
  scalling_speed: 0.25
  max_scalling_factor: 0.2

  oscillation_reset_dist: 0.05

  publish_traj_pc: true
  publish_cost_grid_pc: true
```

Εικόνα 69. Κώδικας DWA Local PLanner

Acc lim X: Το όριο επιτάχυνσης X του ρομπότ σε μέτρα/δευτ^2

Acc lim th: Το όριο περιστροφικής επιτύαχυνσης του ρομπότ σε ακτίνια/δευτ^2

Max vel X: Η μέγιστη ταχύτητα X για το ρομπότ σε μέτρα/δευτ

Min vel X: Η ελάχιστη ταχύτητα X για το ρομπότ σε μέτρα/δευτ, αρνητική για κίνηση προς τα πίσω

Max vel trans: Η απόλυτη τιμή της μέγιστης μεταφορικής ταχύτητας για το ρομπότ σε μέτρα/δευτ

Min vel trans: Η απόλυτη τιμή της ελάχιστης μεταφορικής ταχύτητας για το ρομπότ σε μέτρα/δευτ

Max vel theta: Η μέγιστη επιτρεπόμενη ταχύτητα περιστροφής για τη βάση σε ακτίνια/ δευτ

Min vel theta: Η ελάχιστη επιτρεπόμενη ταχύτητα περιστροφής για τη βάση σε ακτίνια/ δευτ

Min in place vel theta: Η ελάχιστη ταχύτητα περιστροφής που επιτρέπεται για τη βάση ενώ εκτελούνται επιτόπιες περιστροφές σε ακτίνια/δευτ

Holonomic robot: Καθορίζει εάν δημιουργούνται εντολές ταχύτητας για ένα ολονομικό ή μη ολονομικό ρομπότ

Escape vel: Ταχύτητα που χρησιμοποιείται για οδήγηση κατά τις αποδράσεις σε μέτρα/ δευτ.

Yaw goal tolerance: Η ανοχή σε ακτίνια για τον ελεγκτή σε εκτροπή/περιστροφή κατά την επίτευξη του στόχου του.

Latch XY goal tolerance: Εάν η ανοχή στόχου είναι κλειδωμένη, εάν το ρομπότ φτάσει ποτέ στη θέση στόχο XY, απλώς θα περιστραφεί στη θέση του, ακόμα κι αν καταλήξει εκτός της ανοχής στόχου ενώ το κάνει.

Sim time: Ο χρόνος για τη προσομοίωση τροχιών προς τα εμπρός σε δευτερόλεπτα.

Sim granularity: Το μέγεθος του βήματος, σε μέτρα, που πρέπει να γίνει μεταξύ σημείων σε μια δεδομένη τροχιά.

VX samples: Ο αριθμός των δειγμάτων που θα χρησιμοποιηθούν κατά την εξερεύνηση του χώρου ταχύτητας X.

Vtheta samples: Ο αριθμός των δειγμάτων που θα χρησιμοποιηθούν κατά την εξερεύνηση του χώρου ταχύτητας θήτα.

Controller frequency: Η συχνότητα με την οποία θα καλείται ο ελεγκτής σε Hz.

Path distance bias: Η στάθμιση για το πόσο ο ελεγκτής πρέπει να παραμείνει κόντα στη διαδρομή που του δόθηκε.

Goal distance bias: Η στάθμιση για το πόσο πρέπει να προσπαθήσει ο ελεγκτής για να πετύχει τον τοπικό του στόχο, ελέγχει επίσης την ταχύτητα.

Occdist scale: Η στάθμιση για το πόσο πρέπει να προσπαθήσει ο ελεγκτής για να αποφύγει τα εμπόδια.

Forward point distance: Η απόσταση από το κεντρικό σημείο του ρομπότ, για να τοποθετηθεί ένα επιπλέον σημείο βαθμολόγησης σε μέτρα.

Stop time buffer: Το χρονικό διάστημα που πρέπει να σταματήσει το ρομπότ πριν από μια σύγκρουση προκειμένου μια τροχιά να θεωρήθει έγκυρη σε δευτερόλεπτα.

Scalling speed: Η απόλυτη τιμή της ταχύτητας με την οποία ξεκινά η κλιμάκωση του αποτυπώματος του ρομπότ σε μέτρα/ δευτ.

Max scalling factor: Ο μέγιστος παράγοντας για την κλίμακα του αποτυπώματος του ρομπότ.

Oscillation reset dist: Πόσο μακριά πρέπει να διανύσει το ρομπότ σε μέτρα πριν από την επαναφορά των σημαιών ταλάντωσης.

Publish traj pc: Εργαλείο εντοπισμού σφαλμάτων

Publish cost grid pc: Εργαλείο εντοπισμού σφαλμάτων

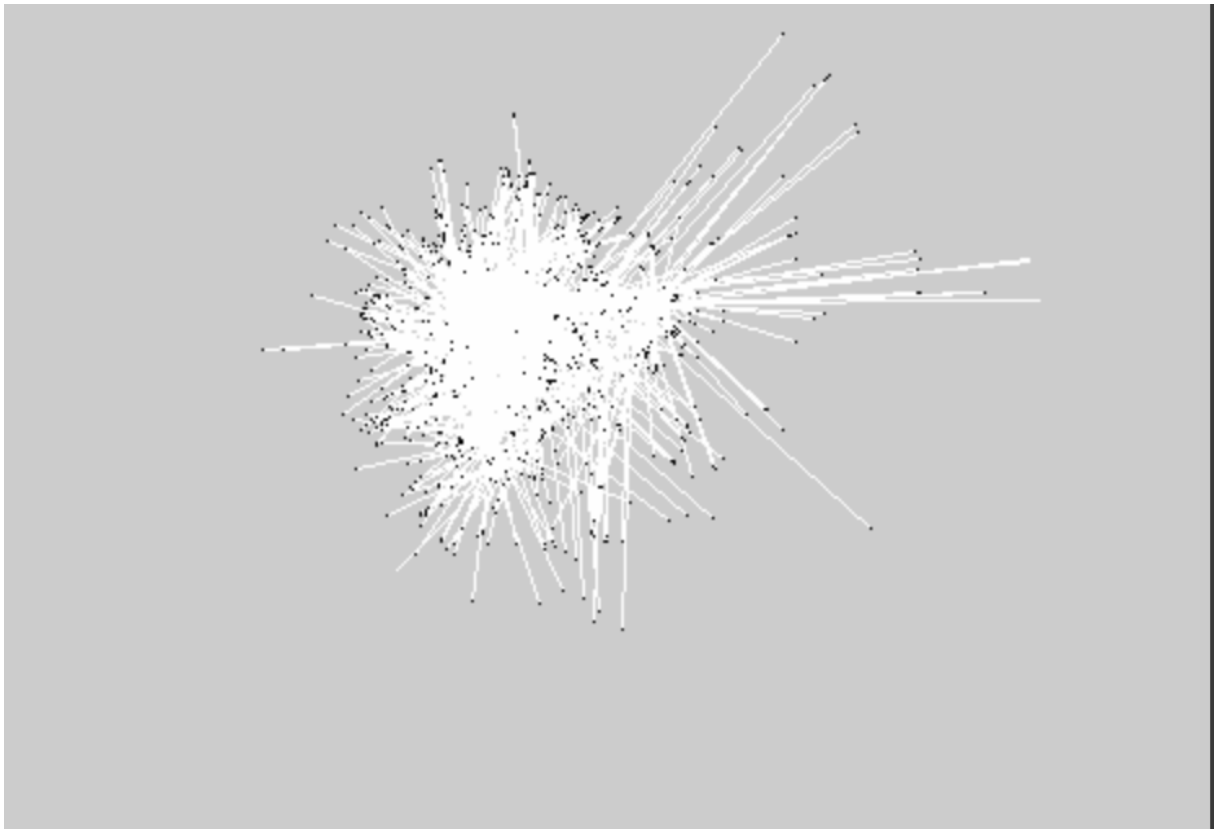
6. Σενάρια Λειτουργίας

Μετά τη περιγραφή της υλοποίησης, ήρθε η ώρα για να τη χρησιμοποιήσουμε. Το βασικό σενάριο του οχήματος είναι η αυτόνομη πλοήγηση. Τα σενάρια λειτουργίας που θα περιγράψω, θα τα κατέτασσα σαν υποσενάρια της αυτονόμης πλοήγησης απλα την εκτελούν με διαφορετικές συνθήκες.

Τα υποσέναρια λοιπόν είναι τρία. Το ένα είναι η χαρτογράφηση, το δεύτερο είναι η πλοήγηση μέσω χάρτη και η τρίτη είναι η πλοήγηση μέσω της οδομετρίας.

- **Πλοήγηση και Χαρτογράφηση**

Στο συγκεκριμένο σενάριο θα πλοηγήσω το όχημα σε έναν κλειστό χώρο.Χρησιμοποιώντας τον κόμβο gmapping, θα χαρτογραφήσουμε όσο το δυνατόν καλύτερα με τα μέσα που διαθέτουμε το περιβάλλοντα χώρο. Αρχικά θα τρέξουμε την εντολή: **roslaunch navstack_pub esp32-gmapping.launch.** Κάνοντας βόλτες στο χώρο, γίνεται παράλληλα και η χαρτογράφηση. Μόλις ολοκληρωθεί η χαρτογράφηση θα χρειαστεί να αποθηκεύσουμε το χάρτη για να μπορούμε να αξιοποιήσουμε τη πλοήγηση με χάρτη. Με την εντολή: **rosrun map_server map_saver -f map** θα αποθηκεύσουμε το χάρτη που δημιουργήσαμε.



Εικόνα 70. Χαρτογράφηση περιβάλλοντος

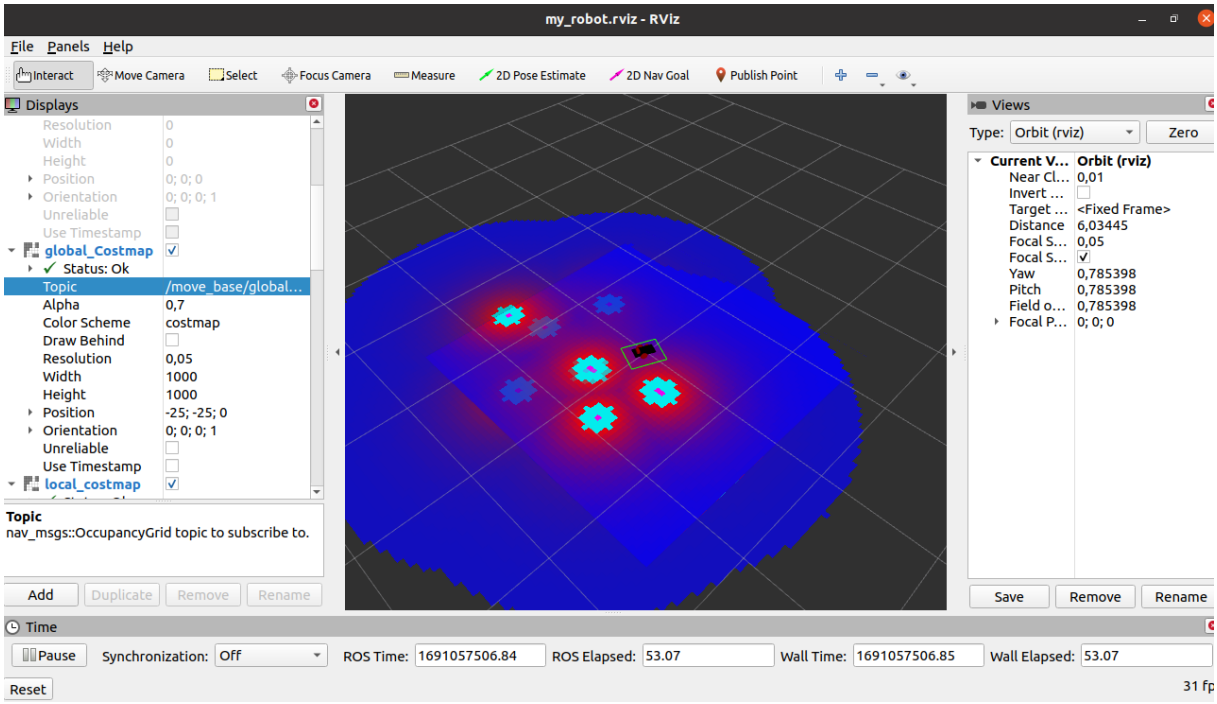
- **Αυτόνομη Πλοήγηση με οδομετρία**

Στο δεύτερο σενάριο, θα πλοηγήσουμε το όχημα με βάση την οδομετρία. Θα λαμβάνει τα δεδομένα από τους αισθητήρες για την αποφυγή εμποδίων και δίνοντάς του κάποιες συντεταγμένες στο χώρο, θα προσπαθήσει να πετύχει το στόχο του. Με την εντολή: **roslaunch navstack_pub esp32-odom.launch** θα εκκινήσουμε τους πολλαπλούς κόμβους για να ξεκινήσει η αυτόνομη πλοήγηση. Όπως βλέπουμε και στην εικόνα παρακάτω, το ROS μας δείχνει τις μεταβλητές που φορτώνονται ανάλογα τα πακέτα που χρησιμοποιούμε.

```
SUMMARY
========

PARAMETERS
 * /filter_madgwick/publish_tf: False
 * /filter_madgwick/use_mag: False
 * /joint_state_publisher/use_gui: False
 * /move_base/DWAPlannerROS/acc_lim_th: 10.0
 * /move_base/DWAPlannerROS/acc_lim_x: 10.0
 * /move_base/DWAPlannerROS/controller_frequency: 10.0
 * /move_base/DWAPlannerROS/escape_vel: -0.5
 * /move_base/DWAPlannerROS/forward_point_distance: 0.325
 * /move_base/DWAPlannerROS/goal_distance_bias: 20.0
 * /move_base/DWAPlannerROS/holonomic_robot: False
 * /move_base/DWAPlannerROS/latch_xy_goal_tolerance: True
 * /move_base/DWAPlannerROS/max_scalling_factor: 0.2
 * /move_base/DWAPlannerROS/max_vel_theta: 0.3
 * /move_base/DWAPlannerROS/max_vel_trans: 1.0
 * /move_base/DWAPlannerROS/max_vel_x: 1.0
 * /move_base/DWAPlannerROS/min_in_place_vel_theta: 0.314
 * /move_base/DWAPlannerROS/min_vel_theta: 0.0
 * /move_base/DWAPlannerROS/min_vel_trans: 0.2
 * /move_base/DWAPlannerROS/min_vel_x: 0.0
```

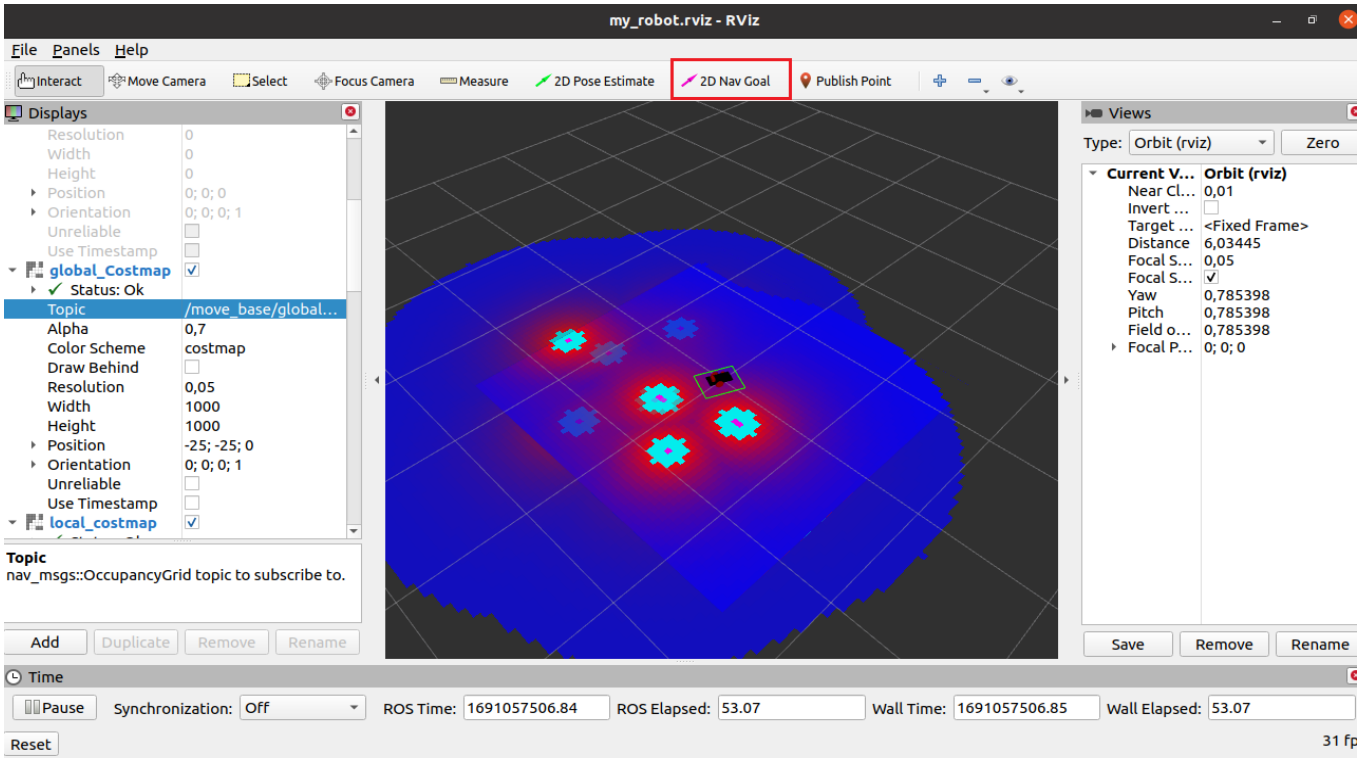
Εικόνα 71. Παράμετροι ESP32 Odom launch

Αυτόματα θα μας ανοίξει και το RVIZ το πρόγραμμα οπτικοποίησης του ROS για να μπορούμε να παρακολουθούμε το τι συμβαίνει με το όχημά μας. Στην παρακάτων εικόνα βλέπουμε το μοντέλο του οχήματος μας, που έφτιαξα με το URDF και τα εμπόδια τα οποία βλέπουν οι αισθητήρες.



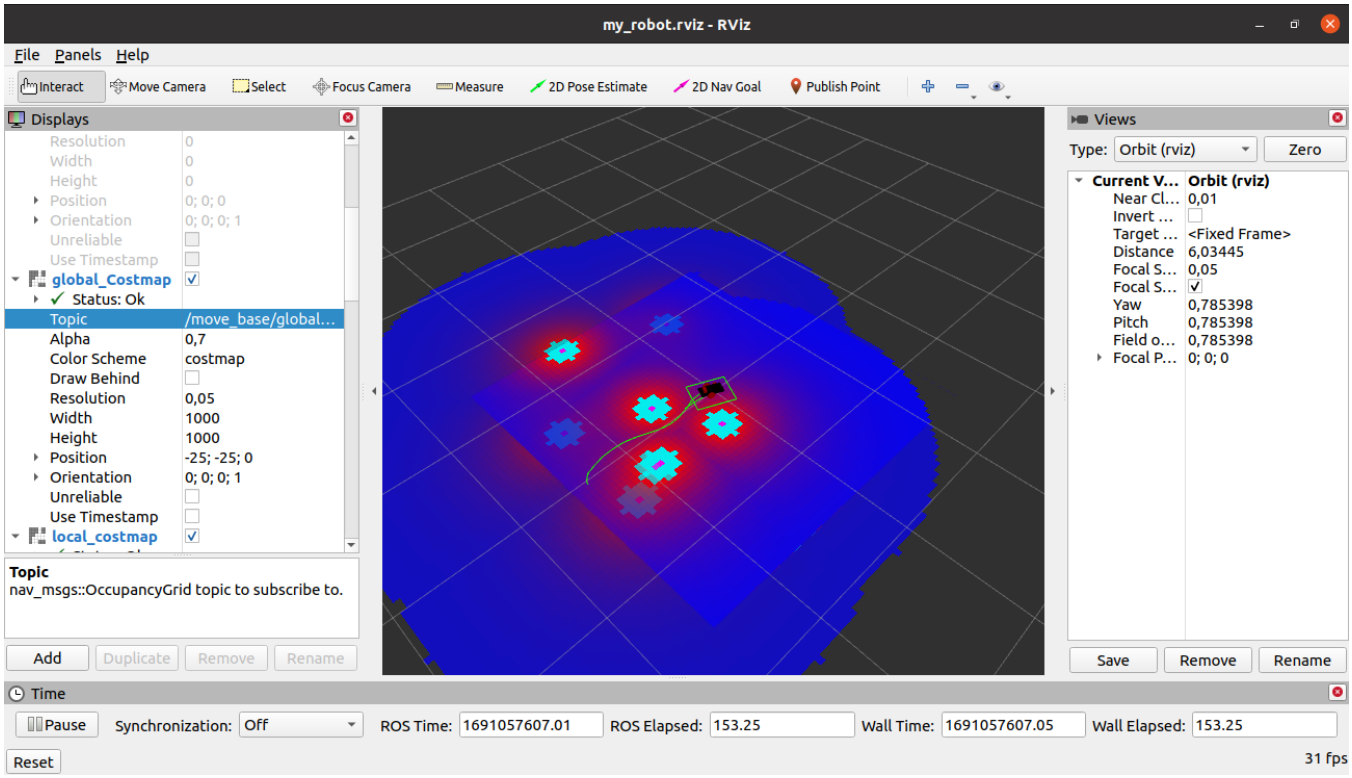
Εικόνα 72. RVIZ

Χρησιμοποιώντας τη γραμμή εργαλείων και πιο συγκεκριμένα το **2D Nav Goal**, μπορούμε να δώσουμε τις συντεταγμένες στο όχημα μας για να κινηθεί.



Εικόνα 73. RVIZ 2D Nav Goal

Όταν θα του δώσουμε τις συντεταγμένες τότε το ROS θα δημιουργήσει το μονοπάτι που πρέπει να ακολουθήσει το όχημά μας για την επίτευξη του στόχου του.



Εικόνα 74. RVIZ δημιουργία μονοπατιού

- **Αυτόνομη πλοήγηση με χάρτη**

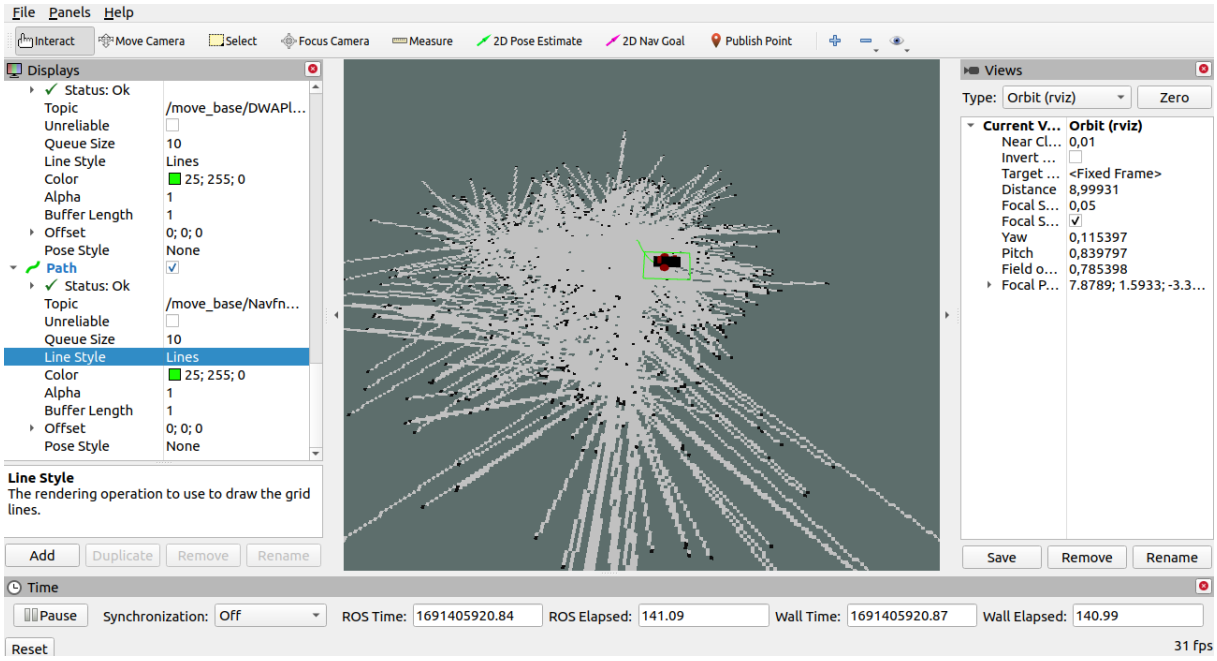
Στο τρίτο και τελευταίο σενάριο έχουμε την αυτόνομη πλοήγηση με χάρτη. Το σενάριο αυτό λειτουργεί σε συνάρτηση με τη πλοήγηση με χαρτογράφηση, διότι χρειάζεται να έχουμε το χάρτη του περιβάλλοντος. Μετά την ολοκλήρωση της χαρτογράφησης φορτώνουμε τον χάρτη με τη πρόσθεση του πακέτου map_server στο αρχείο launch.

```
38  <arg name="map_file" default="$(find my_robot)/map/map.yaml"/>
39
40 <node pkg="map_server" name="map_server" type="map_server" args="$(arg map_file)" />
41
```

Εικόνα 75. Πακέτο Map Server στο launch αρχείο

Έπειτα με την εντολή **roslaunch navstack_pub esp32-amcl.launch** φορτώνουμε τους κόμβους που χρειάζεται για να τρέξει η αυτόνομη πλοήγηση χωρίς χάρτη. Στο RVIZ στη γραμμή εργαλείων, υπάρχει το κουμπί, **2D Pose Estimate,** με αυτό τοποθετούμε το όχημα

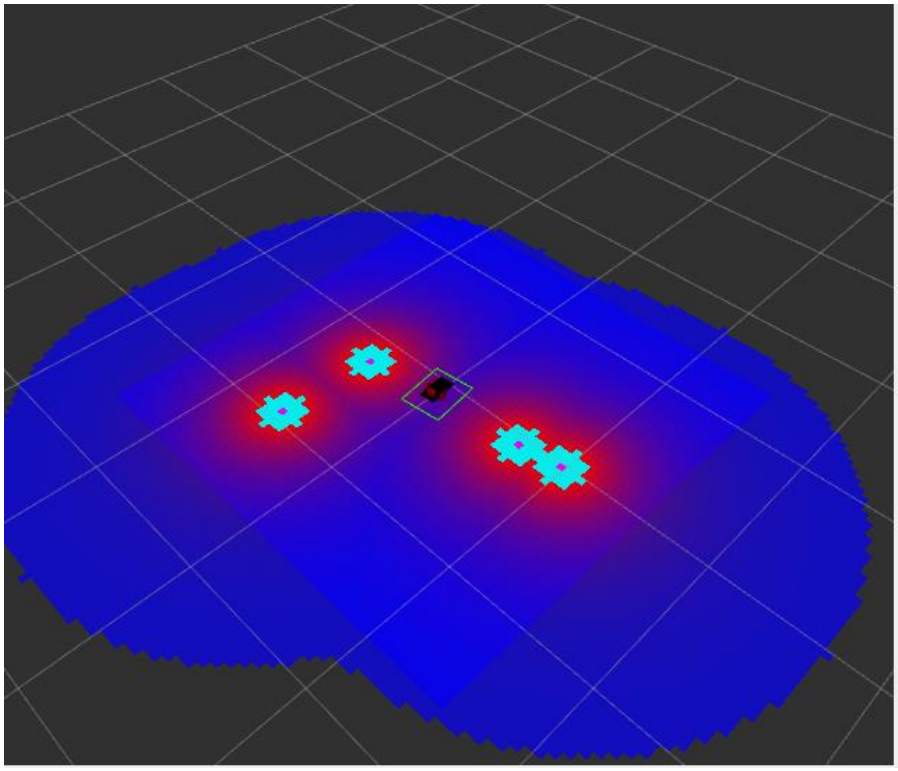
μου στον χάρτη και έπειτα με το **2D Nav Goal** δίνουμε στόχο στον χάρτη για το όχημα μας. Αυτό δημιουργεί το μονοπάτι που πρόκειται να ακολουθήσει με σκοπό την επίτευξη του.



Εικόνα 76. Πρόγραμμα RVIZ

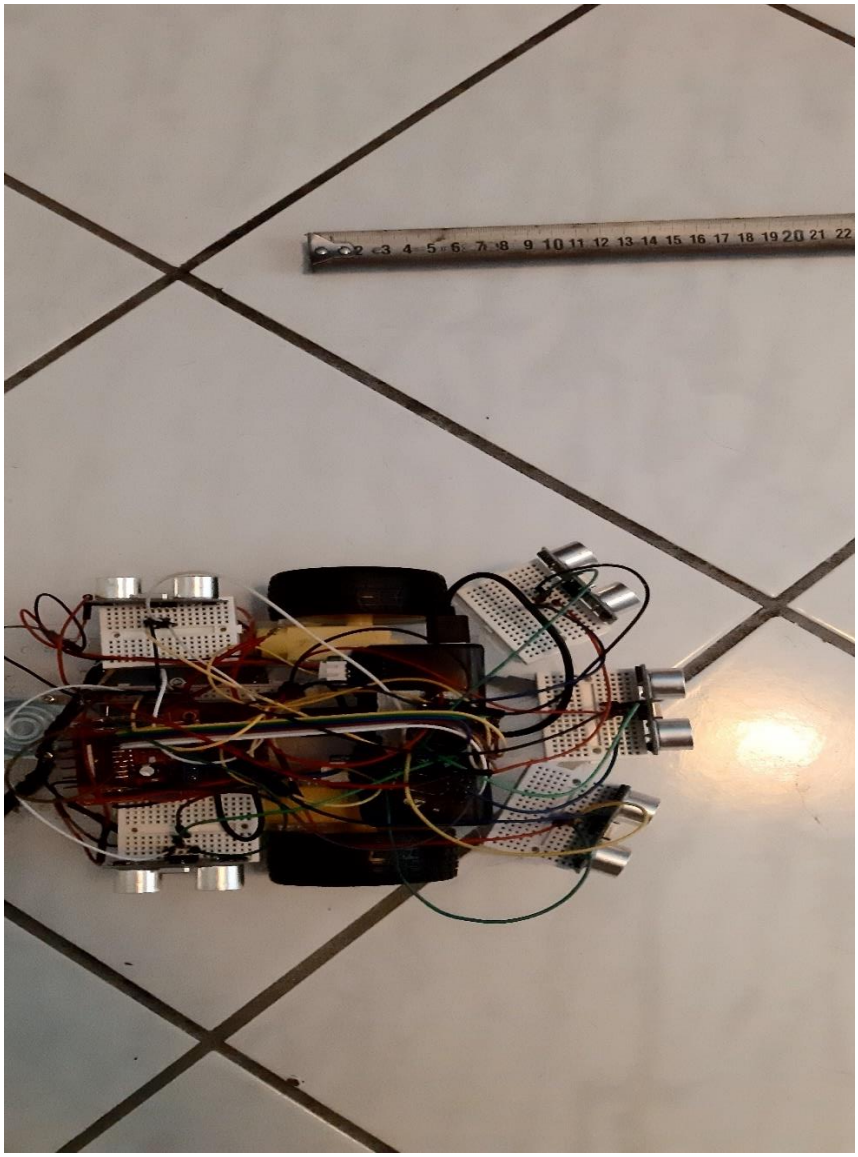
7. Αξιολόγηση συστήματος

Έπειτα από τα σενάρια λειτουργίας, ήρθε η ώρα να αξιολογήσουμε το σύστημα μας, με βάση τα υλικά που διαθέτουμε και τους ανάλογους περιορισμούς. Ξεκινώντας θα δούμε με κάποια πειράματα την κίνηση του οχήματος, δηλαδή κατά πόσο είναι ακριβής σε απόσταση αλλά και όταν του δώσουμε εντολή να στρίψει, επίσης θα δούμε το αποτέλεσμα της οδομετρίας αλλά και της αυτόνομης προήγησης.



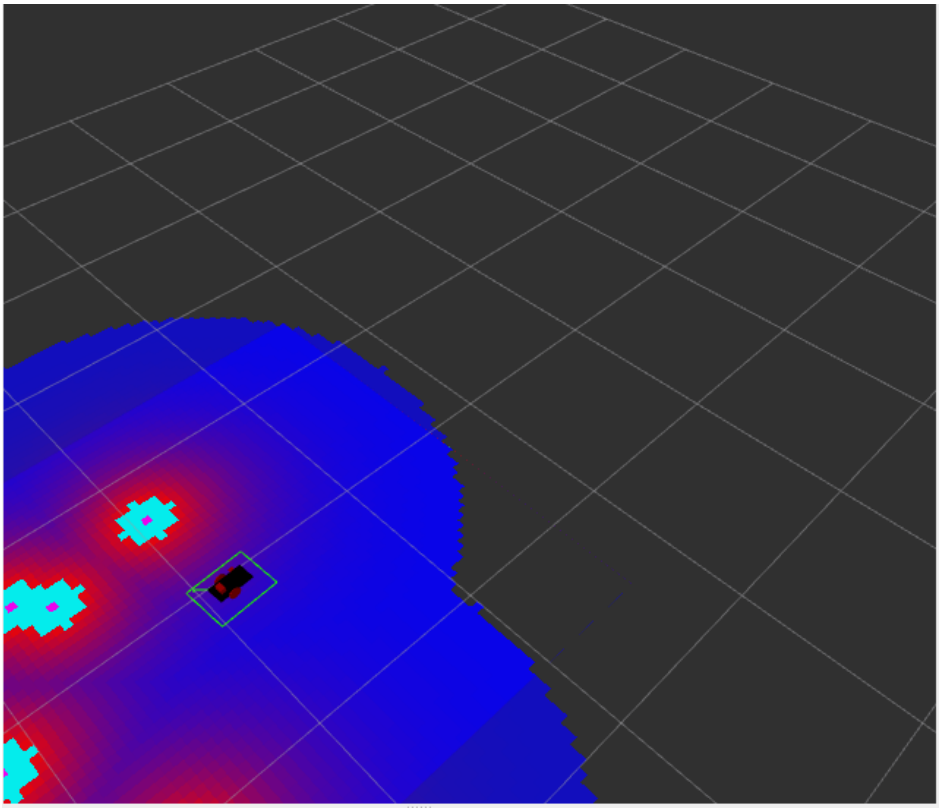
Εικόνα 77. RVIZ

Στην παραπάνω εικόνα βλέπουμε τη θέση του οχήματος στο περιβάλλον. Κάθε τετράγωνο που φαίνεται στο πρόγραμμα αναπαριστά και ένα μέτρο στο περιβάλλον. Θα του δώσουμε εντολή να πάει στην αρχή του επόμενου τετραγώνου. Επίσης έχουμε τοποθετήσει το όχημα στο έδαφος και έχουμε βάλει δίπλα του ένα μέτρο, είναι από ένα εκατοστό μέχρι το ένα μέτρο, η ίδια απόσταση δηλαδή που έχουμε και στο RVIZ.



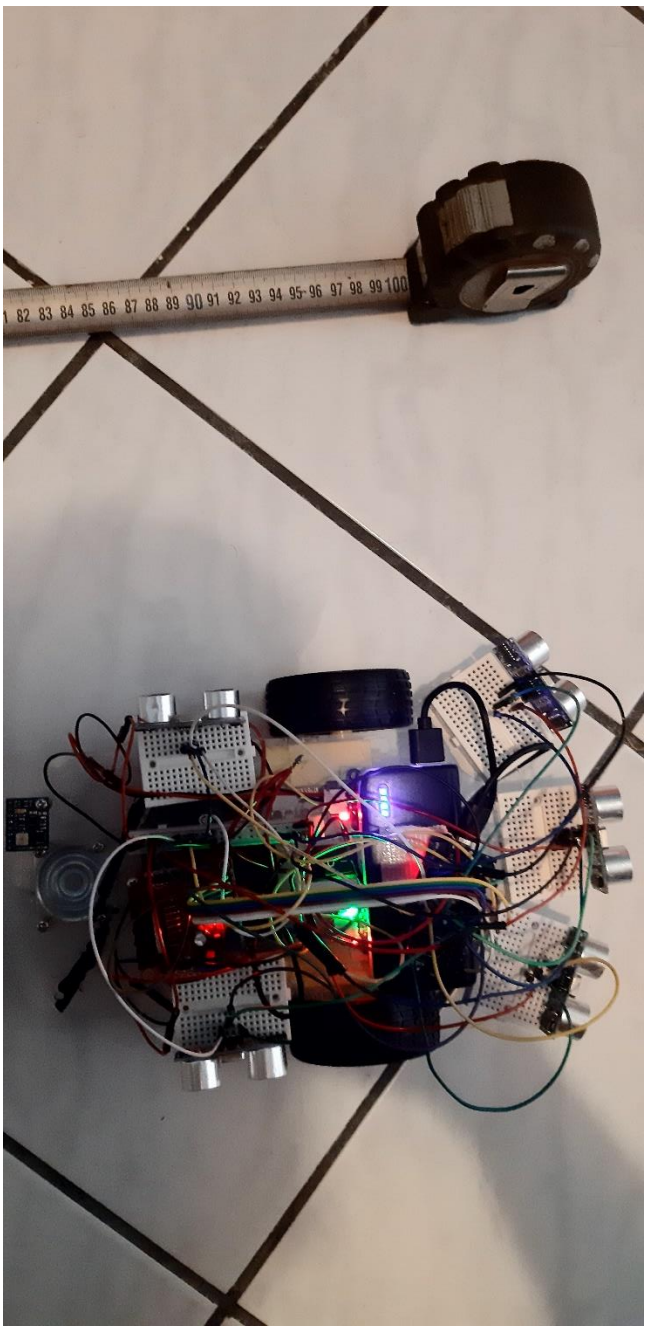
Εικόνα 78. Αρχική θέση οχήματος

Στην παρακάτω εικόνα βλέπουμε στο περιβάλλον του RVIZ το αποτέλεσμα για το σημείο που του δώσαμε. Στον κόμβο move base υπάρχει η παράμετρος goal distance bias, με αυτή καθορίζουμε στο πόσο κοντά χρειάζεται να πάει το όχημα προς το στόχο. Με βάση αυτό που έχουμε βάλει, το όχημα μας λαμβάνοντας αυτή τη παράμετρο σταματάει πλησίον του σημείου. Άρα η πιθανότητα κατά το πόσο το όχημά μας θα κάνει ακριβώς ένα μέτρο δεν είναι μεγάλη λόγω των περιορισμών που έχουμε βάλει.



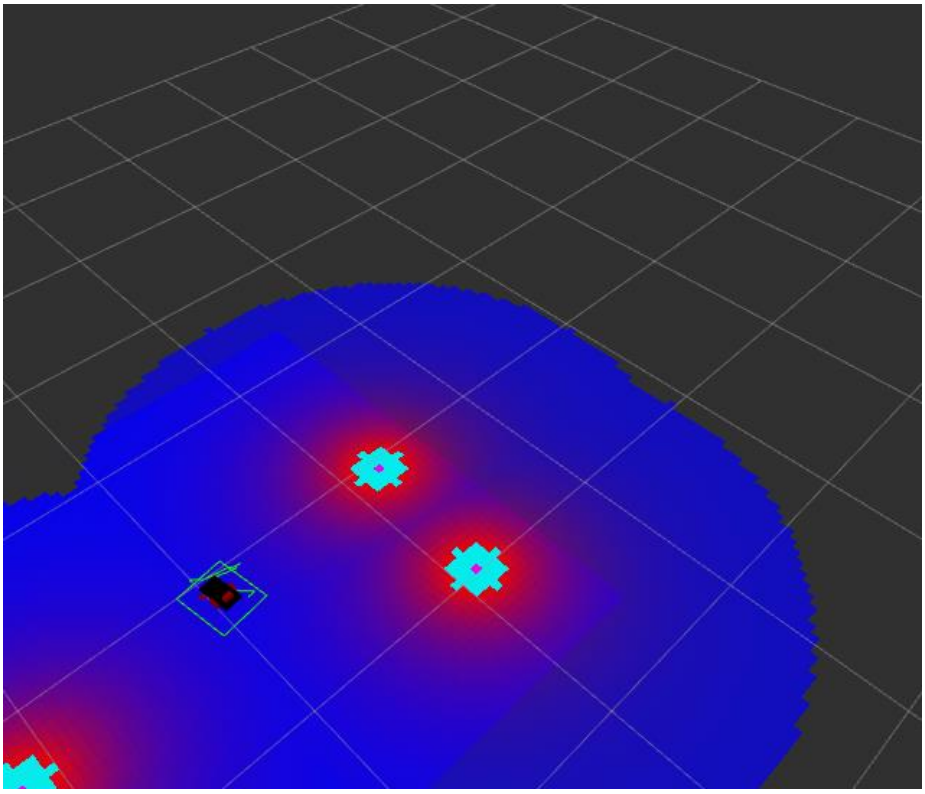
Εικόνα 79. RVIZ

Στην εικόνα που έχουμε βάλει με το μέτρο, βλέπουμε την απόκληση που έχει από το σημείο που δώσαμε μέσω RVIZ. Η διαφορά είναι γύρω στα τρία με τέσσερα εκατοστά, που είναι σύμφωνα με τη παράμετρο που έχουμε δώσει.



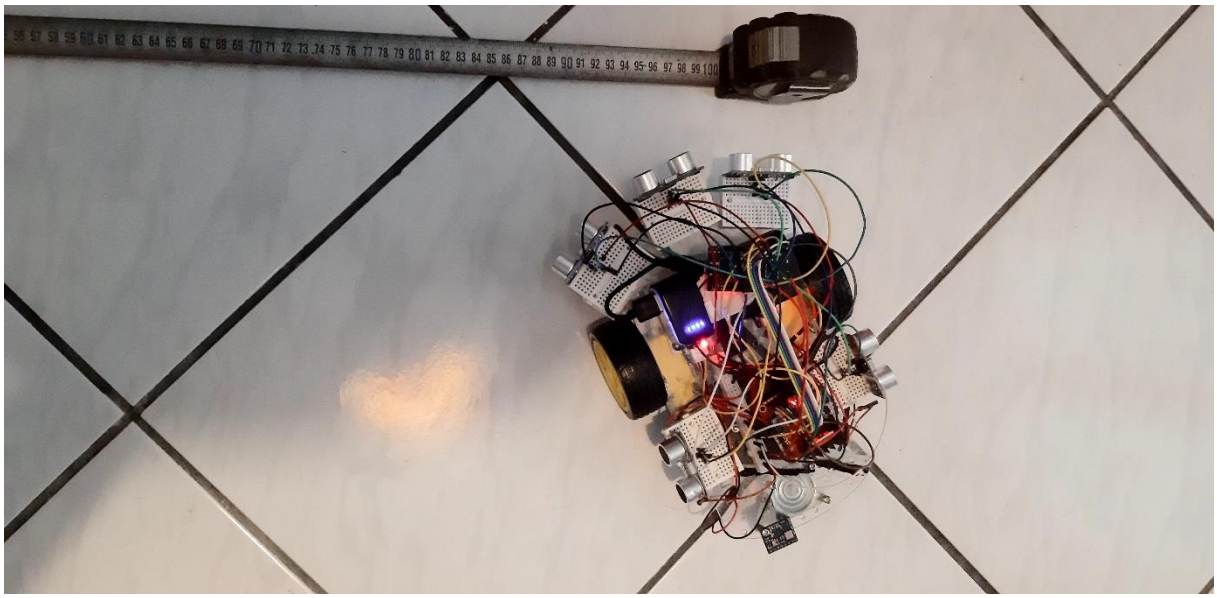
Εικόνα 80. Τελική θέση οχήματος

Στη συνέχεια θα δοκιμάσουμε το στρίψιμο του οχήματος. Στη τελική θέση που σταμάτησε πριν το όχημα του έδωσα εντολή για στρίψιμο ενενήντα μοιρών.



Εικόνα 81. RVIZ

Στη παραπάνω εικόνα βλέπουμε τη θέση του οχήματος μετά τη στροφή ενενήντα μοιρών. Βλέπουμε το όχημα να βρίσκεται σε ορθή γωνία συγκριτικά με τη τελική θέση πριν. Στην από κάτω εικόνα βλέπουμε τη θέση του οχήματος στο περιβάλλον.Διαπιστώνουμε λοιπόν, ότι το όχημα μας δεν είναι ακρίβως σε ενενήντα μοίρες.



Τέλος, μπορούμε να διαπιστώσουμε και για τη λειτουργία της οδομετρίας πόσο καλά λειτουργεί. Αναλύοντας τις παραπάνω εικόνες διαπιστώνουμε ότι η οδομετρία σε μεγάλο βαθμό είναι πολυ ικανοποιητική, βλέπουμε ότι η θέση του οχήματος συμφωνεί με τη θέση που δίνει το πρόγραμμα απεικόνισης του RVIZ. Επίσης, όσων αφορά την IMU, χρησιμοποιήθηκε για τη βελτίωση της οδομετρίας αλλά και μαθησιακούς σκοπούς, δεν λειτούργησε σωστά, αλλά δεν είναι και απαραίτητη για τη σωστή λειτουργία του οχήματος. Το όχημα μας λόγω περιορισμού στους κινητήρες δεν κάνει πίσω κίνηση. Επειδή δεν γίνεται το ROS να υπολογίσει πότε το όχημα μας πάει πίσω λόγω των κινητήρων που διαθέτουμε, έχουμε απενεργοποιήσει τη λειτουργία αυτή.

Μετά θα εξετάσουμε την αυτόνομη πλοήγηση του συστήματος. Στη παρακάτω φωτογραφία φαίνεται η πορεία που έχουμε δώσει να ακολουθήσει το όχημά μας.

Εικόνα. Η πορεία που θα ακολυθήσει το όχημα μας

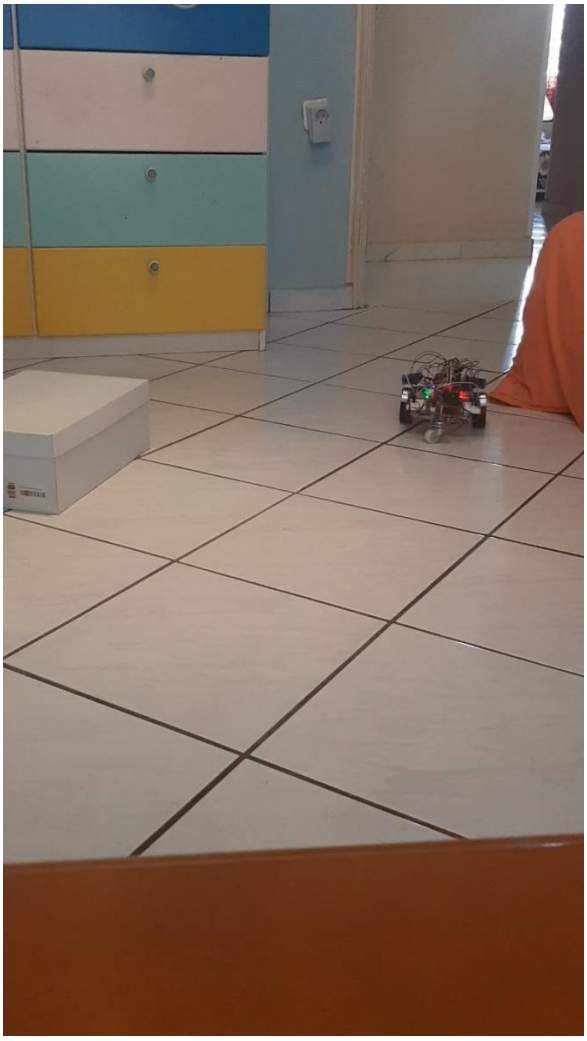
Το όχημα στη αρχή στρίβει δεξιά και πάει παράλληλα με το εμπόδιο δεξια, διάλεξε εκείνο το μονοπάτι αν και θα μπορούσε να κατευθυνθεί ευθεία. Δυστυχώς, οι αισθητήρες απόστασης δεν βοηθούν πολύ, διότι έχουν περιορισμούς.

Εικόνα 82. Κατεύθυνση οχήματος 1

Στη συνέχεια το όχημα θα κάνει μια αριστερή στροφή για να αποφύγει τη γωνία του εμποδίου. Αν και πιθανότητα το όχημα μας δεν “βλέπει” το εμπόδιο, εξαιτίας των αισθητήρων, δείχνει να το αποφεύγει. Ο περιορισμός στον αισθητήρα απόστασης είναι ότι εντοπίζει το εμπόδιο από τα είκοσι εκατοστά και πίσω. Αν πλησιάσει πιο κοντά χάνεται, οπότε υπάρχει μεγάλη πιθανότητα σύγκρουσης με το εμπόδιο.

Εικόνα 83. Κατεύθυνση οχήματος 2

Μετά την αριστερή στροφή, το όχημά μας κάνει απευθείας μια δεξία στροφή με σκοπό να ακολουθήσει το μονοπάτι που του έχουμε δώσει.



Εικόνα 84. Κατεύθυνση οχήματος 3

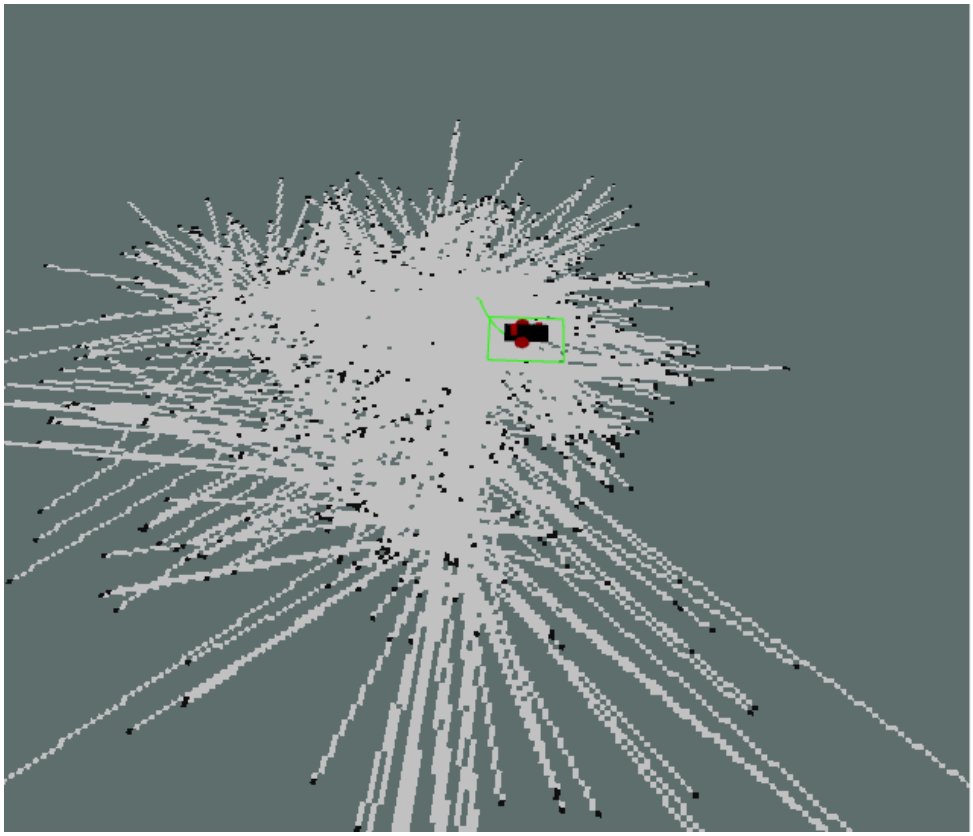
Όπως ανέφερα πιο πάνω επειδή το όχημα πλησίασε υπερβολικά το εμποδίο το έχασε οπότε και έπεσε πάνω του, μετά τη διόρθωση, συνέχισε τη πορεία που του είχαμε δώσει.

Εικόνα 85. Κατεύθυνση οχήματος 4

Τέλος ισιώνει με μια δεξιά στροφή και ακολουθεί το μονοπάτι μέχρι το τέλος. Να επισημάνουμε, ότι τη διαδρομή στο όχημα την δίναμε διακεκομμένα, δηλαδή έφτανε σε ένα σημείο που το είχαμε δώσει και μετά του δίναμε ένα άλλο σημείο για να ακολουθήσει.

Εικόνα 86. Κατεύθυνση οχήματος 5

Στη παρακάτω εικόνα έχουμε εναποθέσει τη χαρτογράφηση που έχουμε κάνει σε ένα δωμάτιο, μπορούμε να διαπιστώσουμε ξεκάθαρα ότι η χαρτογράφηση με τους συγκεκριμένους αισθητήρες, είναι πολύ κακή και δεν βοηθάει στην αξιοποίηση του. Βλέπουμε ότι υπάρχουν παντού κουκίδες που σημαίνουν ότι εκεί υπάρχουν εμπόδια, αλλά δεν είναι αξιόπιστα.



Εν τέλει, το όχημα μας για τα υλικά που διαθέσαμε για το κατασκευάσουμε έχει καλές επιδόσεις, σίγουρα αν διαθέταμε καλύτερους αισθητήρες θα ήταν πολύ πιο αξιόπιστο. Όσο αφορά, το μονοπάτι που ακολουθεί, αν του δώσουμε το ίδιο σημείο, πολύ πιθανόν κάθε φορά να ακολουθεί διαφορετική διαδρομή. Πιστεύω ότι αυτό οφείλεται κυρίως στους αισθητήρες απόστασης που έχουμε. Για τη σωστή λειτουργία του συγκεκριμένου οχήματος, πρέπει πρώτον, τα εμπόδια να είναι σε απόσταση είκοσι εκατοστά και άνω. Δεύτερον, λόγω των αισθητήρων δεν μπορούμε να του δώσουμε μια μακρινή διαδρομή, και πρέπει να δίνουμε διακεκομμένα τα σημεία που θέλουμε να ακολουθήσει. Τρίτον, λόγω των κινητήρων που διαθέτουμε, αν βρει σε ένα εμπόδιο πρέπει τον έλεγχο να τον πάρουμε εμείς και να το τοποθετήσουμε εκτός του εμποδίου.

8. Συμπεράσματα-Μελλοντικές επεκτάσεις

Με την ολοκλήρωση της υλοποίησης του συστήματος, καταφέραμε να δημιουργήσουμε ένα σύνολο από αισθητήρες, ενεργοποιητές και λογισμικό τα οποία συνεργάζονται μεταξύ τους ώστε ένα όχημα με 3 ρόδες να μπορεί να πλοηγείται αυτόνομα. Η συνήθης υλοποίηση αυτών των οχημάτων γίνεται σειριακά, εμείς καταφέραμε μέσω του δικτύου να μεταφέρουμε πληροφορίες για το περιβάλλον από και προς τον υπολογιστή μας.

Το σύστημα μας είναι παρομοίο με άλλα συστήματα αυτόνομης πλοήγησης υψηλότερου κόστους.Η κύρια διαφορά είναι οτί εμείς το υλοποιήσαμε με πολύ λιγότερα χρήματα. Με πολυ φθηνότερους αισθητήρες, κινητήρες, μικροελεγκτες και τον υπολογιστή. Η υλοποιήση αυτή μπορεί να λειτουργήσει για υλοποιήσεις ίδιο κόστους, αλλά και μεγαλύτερου κόστους. Θεωρώ ότι είναι μια υλοποίηση για κάθε τσέπη, το μόνο που χρειάζεται είναι διάφορες πινελιές για να το φέρει κάποιος στα μέτρα του.

Τέλος, εκτός απ'ότι η υλοποίηση μας είναι οικονομική, θεωρώ ότι είναι μια καλή εισαγώγη για άτομα που θέλουν να ασχοληθούν με τη ρομποτική. Από προγραμματιστικής άποψης για άτομα που δεν έχουν ασχοληθεί με το ROS είναι δύσκολα στην αρχή, αλλά με πολύ ψάξιμο και διάβασμα, μετά από λίγο διάστημα θα μπορεί να κατανοεί και να διορθώνει τα λάθη τα οποία θα προκύπτουν.

Όμως, θα ήθελα να τονίσω ότι το πρότζεκτ αφορά χαμηλού κόστους υλικά, άρα υπάρχουν και περιορισμοί σε κάποια θέματα σχετικά με την αυτόνομη πλοήγηση. Οι βελτιώσεις που θα ήθελα να προσθέσω είναι:

- Βελτίωση αισθητήρων απόστασης

Στην υλοποίηση μας, χρησιμοποιούμε αισθητήρες υπερήχων οι οποίοι έχουν πολλούν περιορισμούς. Μερικοί απο αυτούς είναι ότι έχουν συγκεκριμένο εύρος αποστάσεων, ανάλογα με τη θερμοκρασία του περιβάλλοντος αλλάζουν και οι μετρήσεις, αν είναι τοποθετημένοι κοντά ο ένας από τον άλλον, υπάρχει πιθανότητα να λάβει το σήμα του διπλανού αισθητήρα και να βγάλει λάθος ένδειξη. Η λύση που θα πρότεινα είναι η αντικατάσταση με αισθητήρες LIDAR. Πολύ πιο ακριβείς και πιο γρήγοροι στις μετρήσεις.

- Βελτίωση στους κινητήρες των τροχών

Οι κινητήρες που διαθέτουμε δεν μας επιτρέπουν να έχουμε τη πίσω κίνηση στο όχημά μας. Υπάρχουν κινητήρες που έχουν ενσωματωμένο τον αισθητήρα ταχύτητας με τον οποίο εύκολα μπορούμε να εντοπίσουμε τη πίσω κίνηση στο όχημα.

- Αναβάθμιση υπολογιστή

Στην υλοποίηση μας το όχημα μου συνεργάζεται με το λάπτοπ μέσω WiFi. Στις υλοποιήσεις τον περισσοτέρων, διάθετουν υπολογιστές μικρούς σε μέγεθος, οι οποίοι μπορούν να τοποθετηθούν πάνω στο όχημα και με σειριακή σύνδεση να επικοινωνεί το όχημα με το λογισμικό κάνοντας το πολύ πιο γρήγορο και εύχρηστο.

- Επέκταση του συστήματος

Στο σύστημα μας θα μπορούσαμε να προσθέσουμε αισθητήρες για τη βελτίωση της πλοήγησης. Εκτός από την εγκατάσταση του LIDAR αντί των αισθητήρων υπερήχων, θα μπορούσαν αυτοί οι αισθητήρες να συνεργάζονται μαζί για καλύτερα αποτελέσματα. Μια άλλη προσθήκη θα μπορούσε να είναι το GPS που θα βοηθούσε στον προσανατολισμό και τη κατεύθυνση του οχήματος.

9. Βιβλιογραφία

[1] J. Walker, "What are Autonomous Robots?," 2023. [Online]. Available: https://locusrobotics.com/what-are-autonomous-robots. [Accessed 20 May 2023].

[2] "Autonomous robot," 4 February 2022. [Online]. Available: https://en.wikipedia.org/wiki/Autonomous_robot. [Accessed 20 May 2023].

[3] "Autonomous Mobile Robots Market," July 2023. [Online]. Available: https://www.precedenceresearch.com/autonomous-mobile-robots-market. [Accessed 20 July 2023].

[4] "Robot Operating System," 17 June 2021. [Online]. Available: https://en.wikipedia.org/wiki/Robot_Operating_System. [Accessed 23 May 2023].

[5] "Packages," 4 April 2019. [Online]. Available: http://wiki.ros.org/Packages. [Accessed 28 May 2023].

[6] J. Singh, "What is Odometry?," 28 October 2023. [Online]. Available: https://www.tutorialspoint.com/what-is-odometry. [Accessed 28 May 2023].

[7] S. M. Abhay S, "Autonomous Navigation Market," Allied Market Research, 2022.

[8] T. Bennet, "Autonomous navigation autonomous robotics," 3 September 2020. [Online]. Available: https://inertialsense.com/autonomous-navigation-autonomous-robotics-101/. [Accessed 30 May 2023].

[9] Sebastian, "Robot Operating System: Requirements for Autonomous Navigation," 24 January 2022. [Online]. Available: https://dev.to/admantium/robot-operating-system-requirements-for-autonomous-navigation-1nl. [Accessed 31 May 2023].

[10] "Rosserial," 19 November 2011. [Online]. Available: http://library.isr.ist.utl.pt/docs/roswiki/rosserial.html. [Accessed 31 May 2023].

[11] "ESP32," 3 September 2016. [Online]. Available: https://en.wikipedia.org/wiki/ESP32. [Accessed 5 June 2023].

[12] "L298 H Bridge Drive," 6 August 2016. [Online]. Available: https://wiki.eprolabs.com/index.php?title=L298_H_Bridge_Drive. [Accessed 6 June 2023].

[13] "Inertial measurement unit," 8 March 2006. [Online]. Available: https://en.wikipedia.org/wiki/Inertial_measurement_unit. [Accessed 7 June 2023].

[14] "Distributions," [Online]. Available: http://wiki.ros.org/Distributions. [Accessed 4 July 2023].

[15] "Arduino Integrated Development Environment (IDE) v1," [Online]. Available: https://docs.arduino.cc/software/ide-v1/tutorials/arduino-ide-v1-basics. [Accessed 6 July 2023].

[16] "Arduino IDE Setup," 14 December 2022. [Online]. Available:

http://wiki.ros.org/rosserial_arduino/Tutorials/Arduino%20IDE%20Setup. [Accessed 6 July 2023].

[17] "Launch files," [Online]. Available: http://www.clearpathrobotics.com/assets/guides/melodic/ros/Launch%20Files.html. [Accessed 24 July 2023].

[18] "Setup and Configuration of the Navigation Stack on a Robot," [Online]. Available: http://wiki.ros.org/navigation/Tutorials/RobotSetup. [Accessed 27 July 2023].