ΔΙΕΘΝΕΣ ΠΑΝΕΠΙΣΤΗΜΙΟ ΤΗΣ ΕΛΛΑΔΟΣ

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ, ΥΠΟΛΟΓΙΣΤΩΝ ΚΑΙ ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ

ΠΡΟΓΡΑΜΜΑ ΜΕΤΑΠΤΥΧΙΑΚΩΝ ΣΠΟΥΔΩΝ ΣΤΗ ΡΟΜΠΟΤΙΚΗ

«Ανάπτυξη ρομποτικού Rover τύπου Ackermann, με βάση το Robot Operating System»

Εκπονητής: Σπύρου Αλέξανδρος

Επιβλέπων καθηγητής: Καλόμοιρος Ιωάννης

ΣΕΡΡΕΣ, 2022

Βεβαιώνω ότι είμαι συγγραφέας αυτής της Διπλωματικής Εργασίας και πως κάθε βοήθεια την οποία είχα για την προετοιμασία της, είναι πλήρως αναγνωρισμένη και αναφέρεται στη Διπλωματική Εργασία, με κατάλληλη αναφορά. Επίσης, έχω αναφέρει τις όποιες πηγές από τις οποίες έκανα χρήση δεδομένων, εικόνων, ιδεών ή λέξεων, είτε αυτές αναφέρονται ακριβώς είτε παραφρασμένες και αναλαμβάνω πλήρως την ευθύνη για την χρήση που έκανα. Τέλος, βεβαιώνω ότι αυτή η πτυχιακή εργασία προετοιμάστηκε από εμένα προσωπικά, ειδικά για τις απαιτήσεις του Μεταπτυχιακού Προγράμματος στη Ρομποτική.

(Υπογραφή)

ΕΥΧΑΡΙΣΤΙΕΣ

Θα ήθελα να ευχαριστήσω τους καθηγητές του προγράμματος καθώς και τον επιβλέποντα καθηγητή Ιωάννη Καλόμοιρο, για τη δυνατότητα που μου δόθηκε στα πλαίσια του μεταπτυχιακού προγράμματος στη Ρομποτική να πραγματοποιήσω αυτή την ερευνητική εργασία. Η εργασία αυτή, αποτέλεσε μια μοναδική ευκαιρία ακαδημαϊκής μελέτης και προσωπικής ανάπτυξης. Η υποστήριξη και βοήθειά τους καθόλη την διάρκεια της εκπόνησης ήταν εποικοδομητική και καθοριστική.

ΠΕΡΙΛΗΨΗ

Αντικείμενο αυτής της Διπλωματικής εργασίας αποτελεί η μετατροπή ενός τροχήλατου οχήματος τύπου monster track, σε αυτοκινούμενο ρομπότ, αξιοποιώντας το Robot Operating System (ROS). Η οδήγηση του οχήματος πραγματοποιείται μέσω δικτύου, από ειδικά σχεδιασμένο γραφικό περιβάλλον. Στο πρώτο κεφάλαιο τίθεται το πλαίσιο της μελέτης και η περιγραφή των επιδιωκόμενων στόχων. Στο δεύτερο κεφάλαιο, ακολουθεί η επεξήγηση της ορολογίας σχετικά με τα ρομπότ, τα τροχήλατα οχήματα και εν γένει σχετικά με την επιστήμη της ρομποτικής. Επιπλέον, διακρίνονται τα συστατικά στοιχεία ενός ρομπότ και εντοπίζονται χαρακτηριστικά, ιστορικά παραδείγματα. Παρατίθενται ακριβείς επεξηγήσεις της διάταξης Ackermann και του ROS, με βασική περιγραφή των αρχών και δυνατοτήτων τους. Παρουσιάζονται ενδεικτικές μελέτες που έχουν επιστρατεύσει τα ίδια μέσα. Το τρίτο κεφάλαιο παρουσιάζει τα εργαλεία και εξαρτήματα που χρησιμοποιήθηκαν στην κατασκευή, με κύριες οδηγίες ως προς τη χρήση τους. Στο τέταρτο κεφάλαιο βρίσκονται οι περιγραφές των διαδικασιών σχεδίασης, η μεθοδολογία, η ανάπτυξη του ρομπότ, καθώς και επεξηγήσεις σε βασικά σημεία της λειτουργίας του κώδικα, με τα απορρέοντα συμπεράσματα. Σύμφωνα με τις παρατηρήσεις που προέκυψαν, σημειώνονται επιτυχίες, αδυναμίες και πιθανές βελτιώσεις. Συγκεκριμένα, αυτές αφορούν τόσο το προγραμματιστικό όσο και το μηχανολογικό κομμάτι. Τέλος, μελετώνται σχετικές εφαρμογές, κυρίως εκπαιδευτικού χαρακτήρα, και δίνονται παραδείγματα της αξιοποίησης της διάταξης.

Λέξεις - κλειδιά: τροχήλατο, ρομπότ, Ackermann, ROS, Raspberry pi, Arduino

ABSTRACT

The objective of this research work is the transformation of a wheeled monster truck into an Ackermann-driven robot, utilizing Robot Operating System (ROS). Its management will be executed through a wireless network using a specially designed user interface environment. In detail, the first chapter sets the framework of the study and the description of the objectives. In the second chapter, we present the terminology related to robots, wheelers and their categorizations. Moreover, the contribution of robotics to various aspects of everyday life is discussed. In addition, the components of a robot are identified. Thorough explanations of the Ackermann theory and ROS were then sought and basic principles and features were displayed. Similar research and designs presented in the literature are also discussed. The third chapter presents the tools and components used in the construction, with main instructions on their use. In the fourth chapter there are the descriptions of the design processes, the methodology, the development of the robot, as well as explanations in key points of the operation of the code. The conclusions of this study are also given. According to our observations, successes, weaknesses and possible improvements are noted. Finally, relevant applications of our robot are studied, mainly of an educational nature, as examples of future configurations.

Key words: wheeled, robot, Ackermann, ROS, Raspberry pi, Arduino

ΠΕΡΙΕΧΟΜΕΝΑ

ΚΑΤΑΛΟΓΟΣ ΕΙΚΟΝΩΝ

ΚΕΦΑΛΑΙΟ 1: Περιγραφή του προβλήματος

Εισαγωγή

Η ρομποτική ως επιστήμη έχει χαρακτηριστεί ως ένα πολύ εξειδικευμένο αντικείμενο. Παρόλα αυτά, στη σύγχρονη εποχή και χάρη σε άλματα προόδου, έχει επιτύχει να ενσωματωθεί στην καθημερινότητα του ανθρώπου. Η τεχνολογία των ρομπότ έχει εξελιχθεί σε τέτοιο βαθμό, που παρέχει τη δυνατότητα χρήσης τους σε ποικιλία εφαρμογών. Η συνεισφορά τους τόσο στην επιστήμη, έρευνα, εργασία, όσο και στην εξυπηρέτηση αναγκών του ανθρώπου κρίνεται σημαντική. Η συνειδητοποίηση αυτή οδήγησε σε αυξανόμενο ενδιαφέρον ενασχόλησης με τον συγκεκριμένο κλάδο, έχοντας ως αποτέλεσμα τη ραγδαία εξέλιξη του. Εμβαθύνοντας στην ορολογία, πρέπει να αναφερθεί ότι το ρομπότ είναι μια μηχανική συσκευή, η οποία έχει ως στόχο την διεκπεραίωση μιας συγκεκριμένης εργασίας και μπορεί να δρα κάτω από τον άμεσο έλεγχο του χειριστή ή αυτόνομα μέσω ενός υπολογιστικού προγράμματος.

Τα δεδομένα που πρέπει να λάβει κανείς υπόψη για να ορίσει τις κατευθυντήριες γραμμές για τον σχεδιασμό και την υλοποίηση μιας ρομποτικής κατασκευής, καθώς και η συνολική επιτυχία της, εξαρτώνται από αρκετούς παράγοντες. Πρωταρχικό βήμα αποτελεί η ξεκάθαρη διατύπωση του λειτουργικού στόχου. Εξίσου κρίσιμη είναι και η ικανότητα της συσκευής να συλλέγει πληροφορίες από το περιβάλλον και να αλληλοεπιδρά με αυτό, προσαρμόζοντας τη λειτουργία της. Επιπλέον, θα πρέπει να συντονίζει τον λειτουργικό του στόχο με την ασφάλεια του ίδιου του ρομπότ, καθώς και του περιβάλλοντος του. Τέλος, δεν πρέπει να παραληφθεί η δημιουργία προσιτού τρόπου χρήσης από το κοινό στο οποίο θα απευθύνεται.

Για τη διευκόλυνση της ομαλής λειτουργίας του ρομπότ, πρέπει να υπάρχει ένας συνδετικός κρίκος μεταξύ των δεδομένων που συλλέγει μέσω των αισθητήρων και της επεξεργασίας αυτών από τον υπολογιστή. Οι διάφοροι τύποι αισθητήρων που προσφέρονται δίνουν τη δυνατότητα συλλογής ποικίλλων πληροφοριών από το περιβάλλον. Στη συνέχεια, αυτές πρέπει να συντονίζονται από τους κατάλληλους επεξεργαστές και τελικά να υπόκεινται στην διαχείριση του χρήστη μέσω υπολογιστή.

1.1 Περιγραφή του προβλήματος της εργασίας

Το αντικείμενο της παρούσας εργασίας είναι η κατασκευή ενός οχήματος τεσσάρων τροχών με ικανότητα κίνησης σε οριζόντιο επίπεδο, ανίχνευσης πιθανών εμποδίων που βρίσκονται στην ακτίνα ελέγχου του και αποφυγή αυτών με κατάλληλη προσαρμογή της κίνησης του. Η κίνηση του οχήματος ελέγχεται από τον χρήστη με την βοήθεια ενός υπολογιστή και μέσω δικτύου. Η διπλωματική εργασία έχει επιμορφωτικό χαρακτήρα και εκπαιδευτικές βλέψεις μέσα από την μετατροπή ενός ηλεκτρικού monster σε ρομπότ με την διάταξη Ackermann, ελέγχοντάς το μέσω υπολογιστή.

Ως αρχικός στόχος και ιδιάζων χαρακτηριστικό της εργασίας, τίθεται η πρόκληση της μοντελοποίησης και λειτουργίας της κατασκευής τύπου monster truck στο ROS (Robot Operating System). Το ROS είναι ένα framework ανοιχτού κώδικα (open source), το οποίο δίνει την δυνατότητα να κατασκευάζονται εξελιγμένα ρομπότ και εγκαθίσταται σε υπάρχον λειτουργικό σύστημα όπως το GNU/ Linux. Στην ουσία αποτελεί μια πλατφόρμα ανάπτυξης λογισμικού για ρομπότ, η οποία περιέχοντας ένα σύνολο εργαλείων αλλά και βιβλιοθηκών, στοχεύει στην απλούστευση και στην βελτίωση της διαδικασίας ανάπτυξης ρομπότ. Παράλληλα, θα επιχειρηθεί η ομαλή συνύπαρξη και συνεργασία των ελεγκτικών μηχανισμών χαμηλού και υψηλού επιπέδου.

Οι στόχοι που επιδιώκονται στην εργασία είναι:

- Η κατανόηση του τρόπου λειτουργίας, της ανάπτυξης και της κατασκευής ενός ρομποτικού οχήματος, με χρήση κατάλληλων εργαλείων λογισμικού και υλικού.
- Η ανάπτυξη ενός ενσωματωμένου συστήματος βασισμένο στον Arduino και το Raspberry Pi, που ενσωματώνει την τεχνολογία του ROS.

Σύμφωνα με τα παραπάνω, οφείλει να πραγματοποιηθεί μια έρευνα σχετικά με την φύση και την λειτουργία των ρομπότ γύρω από το εν μελέτη πλαίσιο, καθώς και βιβλιογραφική επισκόπηση των έως τώρα δεδομένων και σχετικών εφαρμογών.

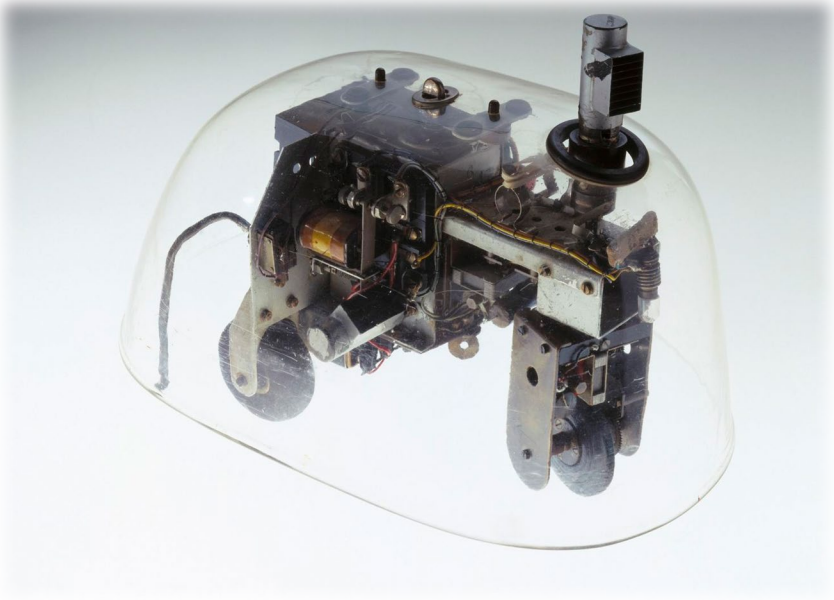
ΚΕΦΑΛΑΙΟ 2: Κατάσταση της Τέχνης

2.1 Τι είναι το ρομπότ

Ανέκαθεν το ρομπότ εμφανιζόταν στην ιστορία ως μια ανθρωποειδής κατασκευή, όπως ισχύει στην ελληνική μυθολογία μέσω του μύθου του Τάλου, αλλά και σε επίδοξα σχέδια του Λεονάρντο ντα Βίντσι. Η λέξη που χρησιμοποιείται σήμερα, «robota», παγίωσε την σημερινή του ταυτότητα μέσα από ένα θεατρικό έργο του Τσέχου Κάρελ Τσάπεκ, το 1921 [1].

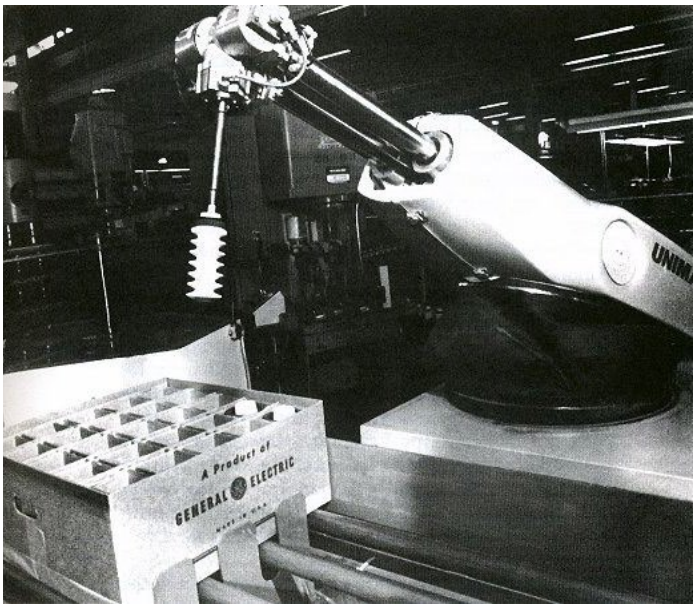
Αν και δεν εντοπίζεται ομόφωνα ένας ορισμός για το ποιες κατασκευές θα νοούνται ως ρομπότ, η επιστημονική κοινότητα έχει καταλήξει σε μια σειρά βασικών χαρακτηριστικών. Ως μια προγραμματιζόμενη κατασκευή, ένα ρομπότ θα πρέπει να μπορεί να είναι ικανό να εκτελεί μια σειρά από ενέργειες. Ακόμη, ως προς τις λειτουργίες τους, πρέπει να αντιλαμβάνονται το περιβάλλον τους, να κινούνται και γενικά να ‘συμπεριφέρονται’ με τρόπο που θυμίζει ζώα ή ανθρώπους [2].

Ένα ρομπότ θα μπορούσε να διακριθεί σε δύο συστήματα, το μηχανικό και το ηλεκτρονικό. Το πρώτο περιέχει το σύστημα κίνησης και το δεύτερο αποτελεί την επαναπρογραμματιζόμενη μνήμη του. Τα ρομπότ αξιοποιούνται σε περίπλοκες για έναν άνθρωπο διαδικασίες και για να πραγματοποιούν εργασίες που είναι αδύνατον, πολύ δύσκολο ή επικίνδυνο να πραγματοποιηθούν από εργαζόμενους. Οι πρώτες φιλόδοξες ανθρωποειδείς κατασκευές έκαναν την εμφάνισή τους σε εκθέσεις το 1928. Βέβαια, τα ρομπότ που συναντώνται κυρίως στην σημερινή πραγματικότητα, δεν έχουν αυτή τη μορφή. Το 1948-49, παρουσιάζονται τα πρώτα αυτόνομα ρομπότ, Elmer και Elsie, από τον William Walter. Διέθεταν τρεις ρόδες και κινούνταν με μπαταρία, που όταν έφτανε σε χαμηλά επίπεδα, αναζητούσαν πηγή επαναφόρτισης. Λόγω της μορφής τους, χαρακτηρίστηκαν ως «χελώνες» [1].



Εικόνα 1. Το ρομπότ «χελώνα» του William Walter [3]

Ο πρώτος βιομηχανικός βραχίονας, που ήταν ψηφιακά προγραμματιζόμενος, έκανε την εμφάνισή του το 1954 από τον George Devol, ενώ το πρώτο ρομπότ που μπορούσε να κινηθεί, αντιληφθεί και να αλληλεπιδράσει με το περιβάλλον του, αναπτύχθηκε στο διάστημα 1966-1972, από το κέντρο ερευνών του πανεπιστημίου Stanford και ονομάστηκε «Shakey» [1].



Εικόνα 2. Βραχίονας «UNIMATE» [4]

Εικόνα 3. Κινητό ρομπότ «Shakey» [5]

2.1.1 Σύστημα ελέγχου

Η επιβίωση του ανθρώπου βασίζεται στην ανατροφοδότηση. Ειδικότερα, αντιλαμβανόμαστε τις συνθήκες στο περιβάλλον μας και δρούμε αντίστοιχα. Στην περίπτωση των ρομπότ, τον ρόλο του εγκεφάλου αναλαμβάνει το σύστημα ελέγχου του, δηλαδή αυτό που ονομάζεται CPU (central processing unit) [6].

2.1.2 Αισθητήρες

Οι αισθητήρες λειτουργούν ως τα δεκτικά όργανα του ρομπότ. Τα δεδομένα που συλλέγουν μεταβιβάζονται σε πραγματικό χρόνο στο κέντρο επεξεργασίας, προκειμένου να επιλεχθεί η κατάλληλη αντίδραση - ενέργεια. Θα παρουσιαστούν αναλυτικά και στην συνέχεια [6].

2.1.3 Επενεργητές

Σε μια αναλογική αντιστοίχηση, οι επενεργητές καταλαμβάνουν τον ρόλο των ανθρώπινων μυών. Συγκεκριμένα, προκειμένου να υποστηριχθεί η κίνηση του ρομπότ στο περιβάλλον του, χρησιμοποιούνται εργαλεία όπως τα μοτέρ και ρόδες [6].

2.1.4 Πηγή ενέργειας

Για την λειτουργία τους, συνήθως χρησιμοποιείται ο ηλεκτρισμός. Στην περίπτωση των αυτόνομων ρομπότ με μορφή μπαταριών, ενώ εκείνα που αποστέλλονται στο διάστημα και οι δορυφόροι, εκμεταλλεύονται και αξιοποιούν την ηλιακή ενέργεια [6].

2.2 Είδη ρομποτικών μηχανισμών

Τα ρομπότ κατατάσσονται σε κατηγορίες, ανάλογα με τη δομή και τη λειτουργική τους φύση [1,7]:

- Σταθερής βάσης, που διαρθρώνονται μέσω συνδέσμων από στερεά μέλη. Το ένα άκρο αποτελεί πάντα την σταθερή βάση και ο χώρος εργασίας του ρομπότ είναι προκαθορισμένος.
- Κινούμενα, mobots (mobile robots), τα οποία εκμεταλλεύονται τον μηχανισμό τους για να κινηθούν σε ένα περιβάλλον. Αυτά κατηγοριοποιούνται ανάλογα με τον βαθμό αυτονομίας τους. Διακρίνονται στα:

Εικόνα 4. Αυτόματο καθοδηγούμενο όχημα (AGV) [2]

- AGVs (Automatic Guided Vehicles), που έχουν περιορισμένο χώρο κίνησης λόγω του περιορισμού της απόστασης στη μετάδοση εντολών μέσω σήματος ή και καλωδίωσης.
- Αυτόνομα έντροχα ρομπότ, τα οποία λειτουργούν με αρκετά υψηλό βαθμό αυτονομίας.
- Βαδίζοντα ρομπότ, με ανθρωποειδή μορφή και μηχανικά πόδια. Σε αυτή την κατηγορία συμπεριλαμβάνονται και αυτά που ονομάζονται ανδροειδή. Πλεονέκτημα αποτελεί η πλοήγησή τους σε μη επίπεδες επιφάνειες και η ευκολότερη αποφυγή εμποδίων. Βέβαια, κάποιο ρομπότ μπορεί να είναι βαδίζον, αλλά να έχει περισσότερα από δύο πόδια.
- ROVs (Remotely Operated Vehicles), ως υποβρύχια ρομπότ (underwater robot). Ελέγχονται εξ αποστάσεως, αλλά ο χώρος πλοήγησης είναι σε αυτή τη περίπτωση περιορισμένος, προκειμένου να διατηρείται η επικοινωνία με το πλοίο-χρήστη.
- AUVs (Autonomous Underwater Vehicles), είναι αυτόνομα, έχουν σχήμα τορπιλών και μετακινούνται πολύ γρήγορα, σε αντίθεση με τα ROVs.
- Εναέρια ρομπότ, δηλαδή τα μη επανδρωμένα ιπτάμενα ρομπότ. Λόγω της αστάθειας, ο έλεγχός τους είναι απαιτητικός και οι εφαρμογές τους έχουν μεγάλο υπολογιστικό κόστος.

Εικόνα 5. ROVs Τηλεχειριζόμενο όχημα [2]

2.3 Τι είναι ρομποτική

Ετυμολογικά, η λέξη πρωτοεμφανίστηκε σε κείμενο του συγγραφέα Ισαάκ Ασίμοβ, ως απόρροια αναζήτησης όρου που θα περιέγραφε τον κλάδο ενασχόλησης με ρομπότ. Στην συνέχεια, αυτή υιοθετήθηκε από όλη την κοινότητα. Ως προς το ακριβές αντικείμενο μελέτης, εφόσον η φύση του πεδίου μελέτης δεν καθορίζεται, έχουν προκύψει αρκετές θεωρήσεις. Ειδικότερα, θα έλεγε κανείς πως η ικανότητα των ρομπότ να αντιλαμβάνονται και να παράγουν λόγο, εμπίπτει στον κλάδο της γλωσσολογίας, ενώ τα θέματα της εικόνας του εαυτού, της προγραμματιζόμενης συνειδητότητας, της θεωρίας του νου, παραπέμπουν στις επιστήμες της νευροβιολογίας και της ψυχολογίας. Άλλοι επιστήμονες και ειδικοί του κλάδου προσφέρουν στις δημοσιεύσεις τους γνώση και εκφέρουν άποψη αποκλειστικά για το σχεδιαστικό και μηχανολογικό κομμάτι. Έτσι, πολλοί επιστήμονες μελετούν την κινηματική των ρομπότ, τον έλεγχο, την ρομποτική όραση. Χαρακτηριστικά, αναφέρονται έργα των Russell & Norvig (1995), «Artificial Intelligence-A modern approach», μια καθολική εισαγωγή στην ερμηνεία της ρομποτικής μέσα από την σκοπιά της τεχνητής νοημοσύνης και οι Pfeifer & Scheier (1999) στο «Understanding Intelligence» [2, 8].

Στην παρούσα εργασία, θα δοθεί έμφαση στο σχεδιαστικό μέρος, επομένως ένας ορισμός που θα μπορούσε να αποτελέσει κατευθυντήρια γραμμή, είναι: Ρομποτική ονομάζεται ο κλάδος της τεχνολογίας που εμπερικλείει την ενασχόληση με τα συστήματα προγραμματισμού, ελέγχου και διάδρασης των ρομπότ και την καθαυτή μελέτη, σχεδιασμό, δημιουργία και αξιοποίηση τους [2].

2.4 Η συμβολή της Ρομποτικής

2.4.1 Βιομηχανική συμβολή

Στον τομέα της βιομηχανίας, τα ρομπότ διαδραματίζουν καίριο ρόλο, αφού έχουν καθορίσει σε μεγάλο βαθμό τα σημερινά δεδομένα. Αποτελούν πλέον το κυριότερο παράδειγμα εργαλείου αυτοματισμού ευρείας χρήσης. Έχουν εξελιχθεί πολύ γρήγορα και για αυτό προτιμώνται σε αρκετές περιπτώσεις εργασιών. Ενδεικτικά, συναντώνται στην πλειονότητα των βιομηχανιών και βιοτεχνιών, βιομηχανίες τσιμέντου, στις βιομηχανίες αυτοκινήτων, βιοτεχνίες μεταλλικών κατασκευών, αυτοκινητοβιομηχανία και βιοτεχνίες ενδυμάτων. Το βασικό πλεονέκτημα ενός ρομπότ είναι η ευελιξία του. Πραγματοποιεί επαναλαμβανόμενες κινήσεις με σταθερή συχνότητα και ακρίβεια. Μπορεί να προσαρμοστεί σε διάφορα προϊόντα στην ίδια γραμμή παραγωγής, όπως απαιτούν οι αλλαγές της αγοράς και να επαναπρογραμματιστεί, έτσι ώστε να είναι χρήσιμο για ποικίλλες αλλαγές του παραγόμενου προϊόντος. Με τον τρόπο αυτό, είναι ικανό να ανταπεξέλθει στις μεταβολές που επιτάσσει η κουλτούρα της βιομηχανίας μαζικής παραγωγής ή και των κατηγοριών του προϊόντος που παράγεται. Έτσι, παρατηρείται ότι το πλήθος των ρομπότ που αξιοποιούνται σε παραγωγικές μονάδες παγκοσμίως, αναπτύσσεται ακόμα και σήμερα αρκετά. Το μεγαλύτερο μέρος από αυτά είναι βιομηχανικά ρομπότ. Αυτό γίνεται γιατί αποδεικνύεται πως είναι, σε κάποια πράγματα, πιο ικανά από τον άνθρωπο, όπως το να σηκώνουν και να μεταφέρουν βαριά αντικείμενα. Εντοπίζονται ακόμη ρομπότ κατασκευασμένα με αυτό που θα χαρακτήριζε κανείς υψηλή νοημοσύνη, αφού αντιλαμβάνονται χάρη σε αισθητήρες τον χώρο που χρησιμοποιούν και μπορούν να καταλαβαίνουν τις αλλαγές. Αυτά τα είδη χρησιμοποιούνται συνήθως σε επαναλαμβανόμενες κινήσεις. Με αυτό τον τρόπο φέρνουν εις πέρας και σωστά το ζητούμενο αποτέλεσμα. Επίσης, στις βιομηχανίες, χρησιμοποιούνται ρομπότ τα οποία προορίζονται για επικίνδυνες εργασίες, οπότε σε τέτοιες περιπτώσεις προστατεύουν τον άνθρωπο. Εκεί αυτός είναι απαραίτητος ως χειριστής της συσκευής και με αυτό τον τρόπο φυλάσσεται από την επικίνδυνη θέση εργασίας [1,2,9].

2.4.2 Παιχνίδια

Ακόμη, ιδιαίτερα διαδεδομένη είναι πλέον η εμφάνιση ρομπότ ως παιχνιδιών. Από παιχνίδια-σκύλους μέχρι ανθρωποειδή ρομπότ για παιδιά και όσους ενασχολούνται με την ρομποτική. Ενδεικτικά, ξεχωρίζει η εταιρεία WowWee με την σειρά Robosapien X [10].

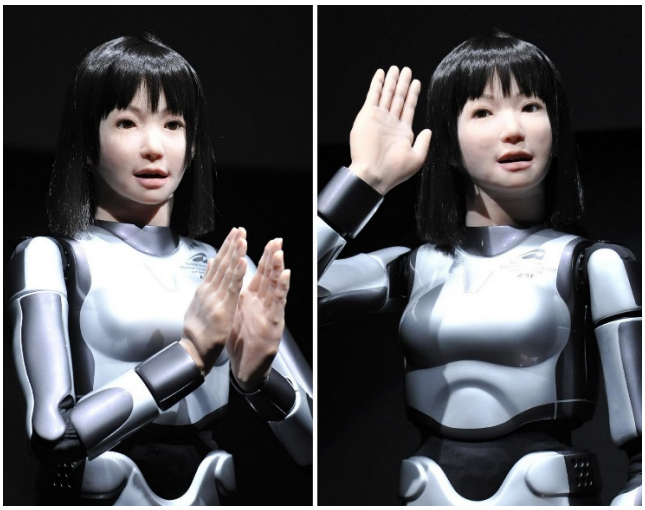
Εικόνα 6. «Robosapien X» [10]

2.4.3 Ιατρική

Από ρομπότ προσομοίωσης έως την υποβοήθηση στη διεξαγωγή περίπλοκων εγχειρήσεων, η καθημερινότητα σε όποια ιατρική εγκατάσταση μπορεί να αποκτήσει νέα πνοή με την χρήση ρομπότ. Ήδη υπάρχουν παραδείγματα σε όλο το κόσμο και συγκεκριμένα στην διανομή φαρμάκων σε ασθενείς στο νοσοκομείο, με το προγραμματιζόμενο ρομπότ να τα παρέχει σε αυτούς και να μετακινείται στο χώρο χρησιμοποιώντας ακόμη και τον ανελκυστήρα. Επιπλέον, σημειώνονται καινοτόμες χρήσεις των ρομπότ, πέραν των ήδη πολυσυζητημένων, ως αρωγών δηλαδή σε διαδικασίες εγχειρήσεων. Παράδειγμα είναι η χρήση των "manipulandum" ρομπότ, τα οποία έχουν από έναν ως δύο βαθμούς ελευθερίας και, όπως μαρτυρά η λατινική ρίζα της λέξης, πρόκειται για μια επέκταση, την οποία μπορεί κάποιος να κινήσει. Αυτά ασκούν ανάλογα με τον προγραμματισμό τους πιέσεις και αναταραχές (petrubations), ώστε κρατώντας μια λαβή, μπορεί κάποιος να αντιδρά στις κινήσεις του ρομπότ και ο χρήστης να κινεί κατά το δοκούν τις λαβές. Το αποτέλεσμα αυτής της αλληλεπίδρασης είναι ικανό να δώσει στοιχεία για την κατάσταση των μυών του χρήστη, την απόκριση των αντανακλαστικών της σπονδυλικής στήλης και άλλα δεδομένα νευροβιολογικής φύσεως [11].

2.4.4 Κοινωνική συμβολή

Ορισμένα ρομπότ λειτουργούν σε περιβάλλοντα που απαιτείται η επικοινωνία με ανθρώπους. Επικοινωνούν με ομιλία, ήχους ή μουσική. Παραδείγματα αποτελούν το Telenoid R1 του Hiroshi Ishiguro [12] και το ανθρωποειδές ρομπότ HRP-4C [13], που αναπτύχθηκε από την Ιαπωνική AIST, με τη μορφή νεαρής γυναίκας. Το πρώτο χρησιμοποιείται για την μεταβίβαση προφορικών μηνυμάτων, μιμούμενο όμως το ύφος και την κινησιολογία του εκάστοτε πομπού. Το δεύτερο, έχει την δυνατότητα να κινείται, να εμφανίζει εκφράσεις στο πρόσωπό της, να αναγνωρίζει ομιλία, να απαντά και να τραγουδά. Άλλα πλαίσια στα οποία συναντώνται ρομπότ στην καθημερινότητα είναι τα εστιατόρια, αναλαμβάνοντας τις εργασίες της κουζίνας, η διανομή των προϊόντων και γενικότερα η εξυπηρέτηση των πελατών, η βοήθεια σε ηλικιωμένους πολίτες και η αξιοποίησή τους από την αστυνομία για την καταδίωξη και εντοπισμό εγκληματιών. Αξίζει να γίνει αναφορά στην πρωτοβουλία μιας καφετέριας στην Ιαπωνία μόνο με υπαλλήλους-ρομπότ (Dawn Avatar Robot Café). Όμως αυτά, τα διαχειρίζονται εξ αποστάσεως εργαζόμενοι με κινητικές δυσκολίες, οι οποίοι βρίσκονται σπίτι τους με τον κατάλληλο εξοπλισμό. Τέλος, το πιο χαρακτηριστικό παράδειγμα είναι η χρήση των ρομπότ ως οικιακές συσκευές, με ηλεκτρικές σκούπες που κινούνται αυτόνομα και χαρτογραφούν το σπίτι μας. Όλα τα αναφερθέντα παραδείγματα στο κοινωνικό πεδίο, ανήκουν σε αυτά που ονομάζονται ρομπότ υπηρεσίας (service robots) [14,15].



Εικόνα 7. Ρομπότ HRP-4C [13]

Εικόνα 8. Ρομπότ σερβιτόροι [15]

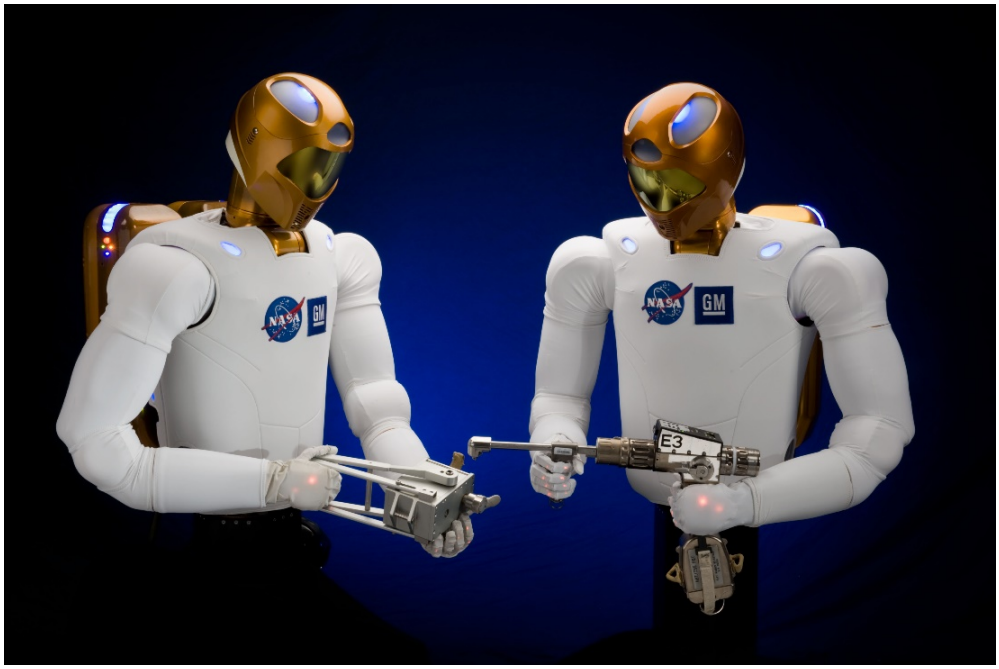
2.4.5 Εκπαιδευτική συμβολή

Στον τομέα της Εκπαίδευσης υπάρχει πλέον μεγάλη κινητοποίηση και ευρεία εξοικείωση σχετικά με την ρομποτική. Αναλυτικά, αναδύεται ο όρος της εκπαιδευτικής ρομποτικής, που έχει ως σκοπό να φέρει τον μαθητή σε επαφή με έννοιες, όπως ανάλυση δεδομένων και προγραμματισμός. Επιπρόσθετα, στοχεύει στην απόκτηση εφοδίων που θα του επιτρέψουν να κάνει σπουδαία έργα. Παράλληλα, θα μπορεί να θέτει στόχους και να κατανοεί την διαδικασία που ακολούθησε, προσφέροντάς του με αυτό το τρόπο αυτοπεποίθηση και φιλόδοξες βλέψεις. Αυτό επιτυγχάνεται με την βοήθεια ρομποτικών συσκευών που χρησιμοποιούνται στα μαθήματα, με αποτέλεσμα να κάνει πιο ψυχαγωγική την εκμάθηση. Έτσι μεγιστοποιούνται οι πιθανότητες οι μαθητές να αποκτήσουν μία καλή σχέση με την τεχνολογία [1]. Ως αντικείμενο της εκπαιδευτικής ρομποτικής, εμφανίζονται κυρίως δραστηριότητες γύρω από ένα προγραμματιζόμενο ρομπότ. Αναζητώντας στο διαδίκτυο, διατίθεται πλέον πληθώρα εκπαιδευτικών κιτ, τα οποία μπορεί να επιλέξει και εντάξει στην διδασκαλία του ένας καθηγητής. Αυτά περιλαμβάνουν όλα τα απαραίτητα επιπρόσθετα εργαλεία που θα επιτρέψουν στο ρομπότ να έχει αυτονομία (αισθητήρες, κινητήρες κ.α). Έχοντας σαν βασική αρχή το «αισθάνομαι, σκέφτομαι και δρω», το ρομπότ εμπεριέχει έναν μικροϋπολογιστή, προκειμένου να διευθετεί τις καθορισμένες ενέργειες που θα θέσει ο εντολέας και να τις χρησιμοποιεί με τον τρόπο που θα επιτάσσει το περιβάλλον του. Έπειτα, οι μαθητές το προγραμματίζουν με μία γλώσσα προγραμματισμού προσαρμοσμένη στο επίπεδο τους, έχοντας πάντα προσχεδιάσει την όποια καθορισμένη λειτουργία Έτσι, μπορεί να αποτελέσει ένα χρήσιμο εργαλείο για την εξέλιξη γνωστικών δομών στα παιδιά ενώ παράλληλα είναι μια δραστηριότητα η οποία προϋποθέτει την ενεργή συμμετοχή των μαθητών με την χρήση της ομαδοσυνεργατικής μεθόδου. Παραδείγματα αποτελούν το διαδεδομένο υλικό της εταιρείας «LEGO», τα «Mindstorms» [16], έως πιο εξειδικευμένα και κοστοβόρα ρομπότ, όπως ο βραχίονας «NiryoOne» [17].

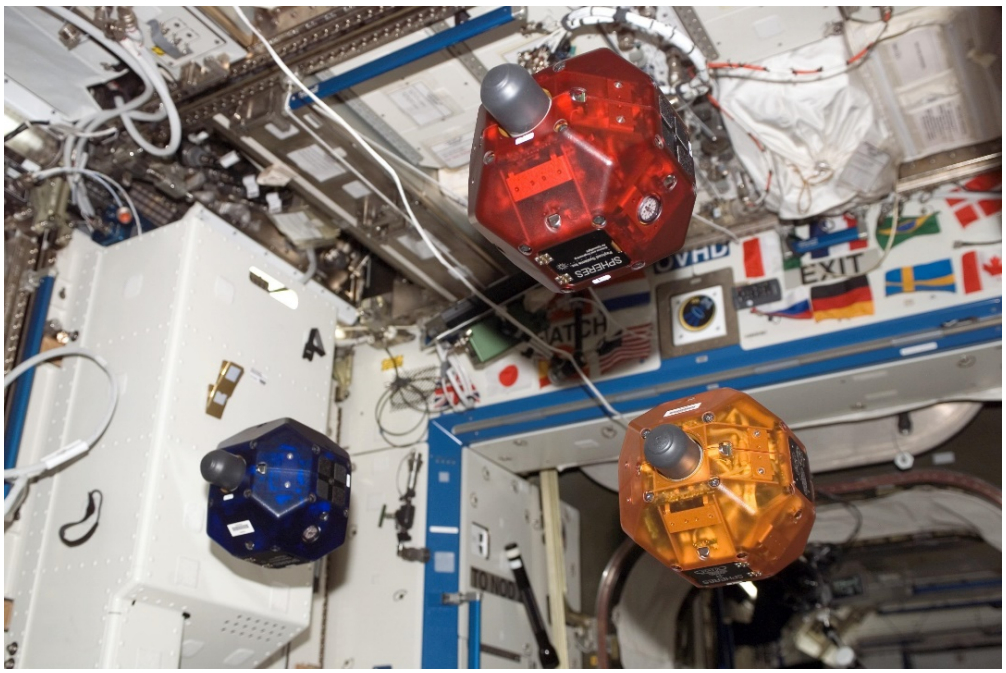
2.4.6 Διάστημα

Όσο αφορά το διάστημα, η συμβολή των ρομπότ είναι σημαντική, αφού αναλαμβάνουν όπως ορίζει και η φύση τους, δραστηριότητες σχεδόν αδύνατες για έναν άνθρωπο. Από την πρώτη επιχείρηση αποστολής στο διάστημα, το πρώτο ταξίδι και προσπάθειες επικοινωνίας μέχρι και την σημερινή λειτουργία των δορυφόρων, τα ρομπότ έχουν συντελέσει στο μεγαλύτερο άλμα ανάπτυξης γνώσεων του ανθρώπου, τόσο στη γη όσο και στο διάστημα. Ένα φιλόδοξο παράδειγμα είναι και το «Robonaut» της NASA, το οποίο μοιάζει με άνθρωπο-αστροναύτη και θα εκτελεί

απαραίτητες εργασίες, καθώς και οι σφαίρες, «SPHERES», οι οποίες έχουν ήδη δοκιμαστεί στον Διεθνή Διαστημικό Σταθμό [18].



Εικόνα 9. «Robonaut» [19]



Εικόνα 10. «Spheres» [18]

2.4.7 Στρατός

Στον τομέα του στρατού, δεν παύουν οι εξελίξεις στην ανάπτυξη μη επανδρωμένων και αυτόνομων ρομπότ. Αξιοποιούνται ευρείας και γενικής χρήσης μηχανισμοί ακόμα και από πολίτες, όπως τα drone, μέχρι εξειδικευμένους στην ανίχνευση και άμυνα.

Εν κατακλείδι, αξίζει να σημειωθεί πως η θέση που κατέχουν τα ρομπότ στην καθημερινή ζωή του ανθρώπου είναι διάχυτη αλλά κάποιες φορές ακόμα και ανεπαίσθητη. Τα βρίσκει κανείς στην χρήση του αυτόματου ταμείου στο σούπερ μάρκετ ή και στην αγορά εισητηρίων από μηχάνημα. Δεν πρέπει όμως να παραβλεφθούν και οι επιφυλάξεις που συνοδεύονται με την ύπαρξη των ρομπότ στην ζωή των ανθρώπων.

2.5 Θετικά και αρνητικά στοιχεία στην εξέλιξη της ρομποτικής

Παλαιότερα, η κατασκευή ενός ρομποτικού συστήματος αποτελούσε μία εξειδικευμένη και κοστοβόρα διαδικασία, που πραγματοποιούταν από ελάχιστες επιστημονικές ομάδες. Ένας επιπλέον παράγοντας ήταν και εξακολουθεί πάντα να είναι το ζήτημα ανάπτυξης των συστημάτων αυτών με γνώμονα την υψηλή ακρίβεια. Πλέον, έχουν αναπτυχθεί μέσα που καθιστούν την δημιουργία ενός ρομπότ απλούστερη και πιο οικονομική. Τα θετικά τα οποία προκύπτουν είναι ότι αναγνωρίζοντας την χρησιμότητα της ρομποτικής στην βιομηχανία, τόσο στη βελτιστοποίηση του παραγώμενου προϊόντος, όσο και στον ρυθμό παραγωγής του, πραγματοποιήθηκαν ραγδαία βελτιώσεις και εξελίξεις. Επιπρόσθετα, αναδύεται ξανά και ο παράγοντας της επικινδυνότητας. Αν και πολλές εργασίες είναι εν γένει δύσκολες, εδώ γίνεται μνεία σε δραστηριότητες ή καταστάσεις που είναι δύσκολο να παρέμβουν άνθρωποι, όπως στις περιπτώσεις που απαρριθμώνται στην προηγούμενη ενότητα. Με λίγα λόγια, εκτός από χρήσιμα, κατατάσσονται και ως ιδιαίτερα σημαντικά [1].

Πέραν του πρακτικού ζητήματος, όπως διαφαίνεται και παραπάνω, τα τελευταία χρόνια έχει μεταβληθεί και η ποιότητα του βιωτικού επιπέδου χάρη στα ρομπότ, από τις ισότιμες ευκαιρίες για δουλειά με το «Dawn Avatar Robot Café» μέχρι την ανάδυση νέων επαγγελμάτων και οριζόντων στην ανθρώπινη εξέλιξη. Από την άλλη, τα αρνητικά δεδομένα που έρχονται στην επιφάνεια, περιλαμβάνουν τις αλλαγές που έχουν επέλθει στον εργασιακό τομέα, ως προς την υπερτέρηση των ρομπότ εναντίον του ανθρώπινου δυναμικού. Τα ρομπότ δεν χρειάζονται να πληρώνονται για να ζήσουν, δεν έχουν ανάγκη κάποια ασφάλεια και δεν εξασθενούν λόγω κούρασης, Δηλαδή, σε συνδυασμό με την εξοικονόμηση πόρων, προτιμώνται από τις βιομηχανίες αντί των ανθρώπων που έχουν ανάγκες. Το γεγονός αυτό έχει φέρει αναστάτωση στις εργασίες

και σε αρκετούς τομείς γενικά, γιατί οι άνθρωποι βλέπουν τις επαγγελματικές προοπτικές και συγκεκριμένες θέσεις εργασίας να ελαχιστοποιούνται.

Βέβαια, οφείλει να σημειωθεί ότι οποιαδήποτε εξέλιξη γύρω από τα ρομπότ είναι θετική και ωφέλιμη, αρκεί να μεριμνάται η διασφάλιση και ευημερία του ανθρώπου. Εάν, λοιπόν, η ρομποτική επιστρατεύεται με καλούς σκοπούς και με σύνεση, καθίσταται ικανή να βοηθήσει σε πάρα πολλούς χώρους, να διευκολύνει και ταυτόχρονα να προστατεύει τον άνθρωπο.

2.6 Τροχήλατα ρομπότ

Στα κινούμενα ρομπότ, διακρίνεται μια επιμέρους κατηγορία, αυτή των τροχήλατων. Ειδικότερα, εντοπίζει κανείς διαφορές στην κινησιολογία τους, ανάλογα με την επιλογή των τροχών. Όπως επιτάσσει και η παρούσα μελέτη, θα διερευνηθούν τα είδη και οι λειτουργίες τους [2, 20].

2.6.1 Είδη τροχών και σχεδιασμός τροχηλάτων ρομπότ

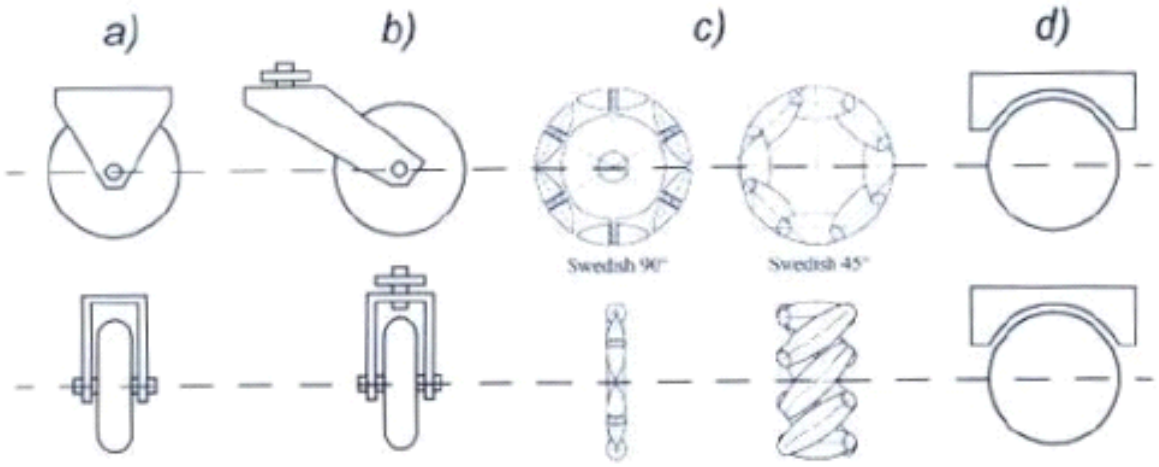
Τα ρομπότ εν δυνάμει, είναι σχεδιασμένα για εφαρμογές σε ποικίλλες περιπτώσεις. Για να υλοποιηθεί λοιπόν ένα τροχήλατο ρομπότ, πρέπει ο σχεδιαστής να αποφασίσει μεταξύ αρκετών διατάξεων και να επιλέξει το είδος των τροχών. Η επιλογή αυτή θα εξασφαλίσει την ευστάθεια, την ευελιξία αλλά και τον έλεγχο του ρομπότ [2].

Συγκεντρωτικά, οι επιλογές τροχών που προσφέρονται για ένα σχεδιαστή, παρατίθονται παρακάτω:

- Κανονικός τροχός, με δύο βαθμούς ελευθερίας και δυνατότητα κίνησης εμπρός-πίσω (a)
- Τροχός orientable (castor), που χρησιμοποιείται κυρίως για να στηρίζει το όχημα. Σε αυτή τη περίπτωση ο έλεγχος του ρομπότ είναι δύσκολος. Διακρίνεται σε centered & off-centered (b)
- Τροχός Omni, γνωστός και ως Swedish, που κινείται σε πολλαπλές κατευθύνσεις. Γνωστή παραλλαγή είναι ο Mecanum (c)
- Σφαιρικός τροχός, ικανός να περιστρέφεται σε όποια κατεύθυνση θέλει και χρειάζεται για την ισορροπία του οχήματος (d) [1]

Τα τέσσερα αυτά βασικά είδη τροχών διαφέρουν αρκετά ως προς το κινηματικό τους αποτέλεσμα. Η κατευθυντικότητα εξασφαλίζεται με τον κανονικό τροχό και τον Castor, χάρη στον βασικό άξονα περιστροφής που έχουν. Δηλαδή, για να μετακινηθεί προς άλλη πορεία, ο τροχός υποχρεώνεται αρχικά να περιστραφεί κατά μήκος ενός κατακόρυφου άξονα. Η διαφορά τους είναι ότι αυτή η διαδικασία μεταβολής κατεύθυνσης είναι πιο εύκολη για τον τυπικό τροχό. Από την

άλλη, ο σουηδικός τροχός και σφαιρικός τροχός δεν θα παρουσιάσουν προβληματισμούς ως προς την αλλαγή πορείας. Ο σουηδικός χρησιμοποιείται όπως ο κανονικός τροχός, αλλά με την βοήθεια των μικρότερων τροχών-κυλίνδρων που υπάρχουν γύρω από την περιφέρεια του κυρίως τροχού, επιτυγχάνεται χαμηλή αντίσταση και είναι δυνατό να μετακινείται η κατασκευή κόντρα στην συμβατική πορεία, δηλαδή κάθετα αυτής με χαμηλή τιμή τριβής. Ο σφαιρικός τροχός δεν έχει περιορισμούς και περιστρέφεται προς όποια κατεύθυνση θέλει [2].



Εικόνα 11. Είδη Τροχών [2]

Τα κινητά ρομπότ σχεδιάζονται και υλοποιούνται για να εξυπηρετήσουν σε εργασίες σε οποιεσδήποτε εδαφικές συνθήκες και περιβάλλον. Τα κύρια χαρακτηριστικά που θα επηρεάζουν την κίνηση ενός τροχήλατου ρομπότ είναι η ευελιξία, η δυνατότητα ελέγχου και η σταθερότητα, που θα καθοριστεί από την επιλογή των τροχών, με γνώμονα, δηλαδή, την γεωμετρία τους. Ακόμη, ο ελάχιστος αριθμός των τροχών που χρειάζονται για στατική σταθερότητα είναι δύο. Θα πρέπει όμως να ισχύει η συνθήκη πως η κύρια μάζα του σώματος είναι κάτω από τον άξονα των τροχών. Λόγω των ιδιαίτερων αυτών παραμέτρων, προτιμάται η χρήση τριών τουλάχιστον τροχών. Όσο περισσότεροι οι τροχοί, τόσο μεγαλύτερη θα είναι και η σταθερότητα που προκύπτει, σε συνδυασμό με σύστημα ανάρτησης. Ως επακόλουθο βέβαια, τίθεται και ο αποτελεσματικός έλεγχος αυτών. Συμπληρωματικά, η ισορροπία θα επιτυγχάνεται όταν το πολύγωνο που θα σχηματίζουν τα σημεία επαφής τροχών με το επίπεδο, εμπεριέχει το κέντρο βάρους [2]. Ως μελέτη περίπτωσης, θα παρουσιαστεί ένα πανκατευθυντικό τροχήλατο ρομπότ για να παρουσιαστούν τα πλεονεκτήματα και μειονεκτήματα ως προς συγκεκριμένες επιλογές.

Ο όρος «παγκατευθυντικό» χρησιμοποιείται για να περιγράψει ένα ρομπότ ικανό να κινηθεί προς όποια κατεύθυνση θέλει. Για να πετύχει αυτό, χρειάζονται τροχοί που είναι ικανοί να μετακινηθούν σε πολλές κατευθύνσεις και αυτό μπορούν να το υποστηρίξουν εκ φύσεως οι σουηδικοί ή οι σφαιρικοί. Λόγω, όμως, της γεωμετρίας των τροχών, δημιουργούνται μικρές αποστάσεις από το επίπεδο, μειώνοντας την ευελιξία του και ταυτόχρονα καθιστώντας λίγο πιο δυσχερή τον έλεγχό του.

Εικόνα 12. Uranus παν-κατευθυντικό κινητό ρομπότ με τροχούς Mecanum. [2]

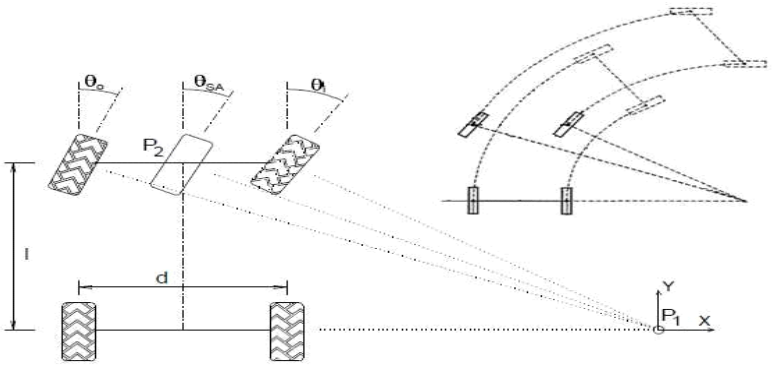
Παράδειγμα αποτελεί ο Uranus, που χρησιμοποιεί τέσσερις σουηδικούς τροχούς. Επικράτησε δηλαδή η αντιστρόφως ανάλογη σχέση ικανότητας ελιγμών και ελέγχου. Για αυτό, μια πρόταση είναι η αντικατάσταση με σφαιρικούς τροχούς. Εναλλακτικά, μια ακόμη αποδοτική διάταξη είναι με σουηδικούς τροχούς 45 μοιρών, με τον καθένα να διαθέτει μοναδικό κινητήρα. Οπότε ο εκάστοτε τροχός μπορεί να επιτύχει καλές αποδόσεις, με μια απλή μηχανική εφαρμογή. Αντίστοιχα, ένα όχημα με τυπικούς τροχούς και με τη διάταξη Ackermann, αν και υστερεί σε ευελιξία, υπερτερεί στον βαθμό ελέγχου [2]. Για αυτό, στην κατασκευή που ακολουθεί έχει επιλεχθεί η διάταξη τροχών των αυτοκινήτων Ackermann.

2.7 Ανάλυση Ackermann

Το κινηματικό μοντέλο Ackermann είναι το πιο γνωστό και αναγνωρισμένο κινηματικό μοντέλο στην αυτοκινητοβιομηχανία. Αποτελεί επινόηση του μηχανικού Georg Lankensperger, όμως κατοχυρώθηκε από τον Rudolph Ackermann. Βασικό μέλημα του μοντέλου είναι να αποφεύγεται, κατά την κίνηση σε καμπυλωτή τροχιά, η πλευρική ολίσθηση [21]. Για την επίτευξη αυτού του αποτελέσματος**,** το κινηματικό μοντέλο Ackermann βασίζεται στην ομώνυμη συνθήκη, στην οποία η σχέση των τροχών στρέψης ενός οχήματος έχει ως αποτέλεσμα την ροή των τροχών δίχως πλευρική ολίσθηση, αν αυτή είναι επιτυχημένη. Η συνθήκη Ackermann, λοιπόν, ορίζει πως οφείλουν «οι κάθετοι στους τροχούς άξονες να τέμνονται σε ένα κοινό σημείο, το οποίο ονομάζεται Στιγμιαίο Κέντρο Περιστροφής (Instantaneous Center of Rotation - ICR) και αποτελεί το κέντρο της στιγμιαίας κυκλικής τροχιάς που ακολουθεί το όχημα» [21].

Σε μια διάταξη τεσσάρων τροχών, είτε με τους πρόσθιους ή οπίσθιους τροχούς ως κατευθυντικούς, η συνθήκη ικανοποιείται με βάση τη γωνία που σχηματίζεται κατά την στροφή. Ειδικότερα, η γωνία του εξωτερικού τροχού είναι λίγο μεταλύτερη από του εσωτερικού. «Αυτό εξασφαλίζει την ελαχιστοποίηση της ολίσθησης των τροχών διευθυνσιοδότησης» [22]. Ακόμη, μεγιστοποιεί την ενεργειακή απόδοση και προστατεύει τους τροχούς από φθορές [23]. Ως επέκταση της συγκεκριμένης διάταξης, αναπτύχθηκε το κινηματικό μοντέλο τετραδιεύθυνσης [21].

Εμβαθύνοντας, σύμφωνα με την εικόνα, διαφαίνεται πως η έκταση των αξόνων των δυο τροχών διευθυνσιοδότησης στον αντίστοιχο άξονα των κινητήριων, συγκλίνει σε ένα κοινό σημείο [24]. Η διαφορά στη γωνία διεύθυνσης μεταξύ των μπροστινών τροχών και η διαφορά στη γωνιακή ταχύτητα για τους πίσω τροχούς, αντιμετωπίζεται μηχανικά από τον μηχανισμό διεύθυνσης και το διαφορικό σύστημα οδήγησης αντίστοιχα. Αυτό πραγματοποιείται επειδή χρησιμοποιείται μόνο ένας ενεργοποιητής για κάθε λειτουργία διεύθυνσης και έλξης. Η κίνηση μπορεί να περιγραφεί από τη γωνία διεύθυνσης και τη ταχύτητα μετατόπισης, καθώς και από την ταχύτητα διεύθυνσης. Βέβαια, οι επιταχύνσεις μετατόπισης και διεύθυνσης λαμβάνονται υπόψη μόνο κατά τη δημιουργία εκτιμήσεων για την τροχιά [24]. Χάρη στις γεωμετρικές συνθήκες που εμφανίζονται, εξασφαλίζεται παράλληλα η διασφάλιση αξιόπιστων μετρήσεων οδομετρίας, που βοηθούν με την σειρά τους σε ακριβή προσδιορισμό θέσεων ενός κινούμενου τροχήλατου ρομπότ [22].



Εικόνα 13. Σχέδιο γεωμετρίας κίνησης τροχών Ackermann [22]

2.7.1 Μελέτη Ackermann

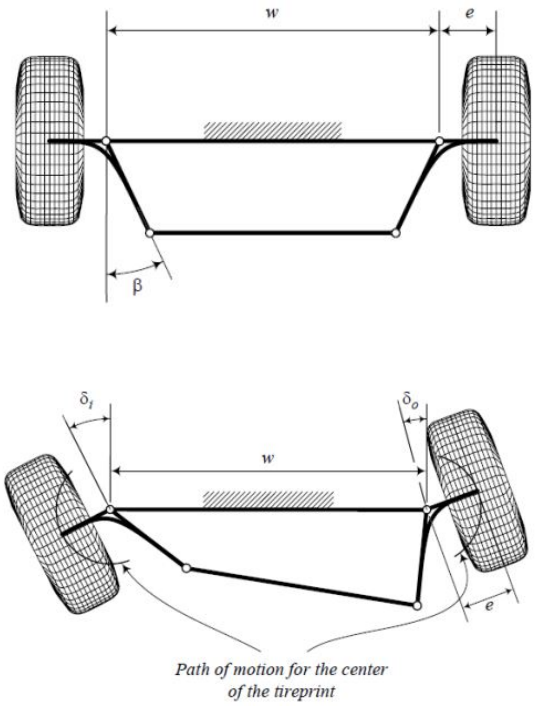
Αναζητώντας σε αντίστοιχες δημοσιεύσεις, εντοπίζεται μια ερευνητική εργασία με σκοπό την ανάπτυξη και αξιολόγηση ενός μοντέλου ρομπότ που θα εφαρμόζει την συνθήκη Ackermann για ελιγμούς σε χαμηλές ταχύτητες [23]. Η συνθήκη αυτή εκ φύσεως περιορίζει το εύρος της κίνησης και τους ελιγμούς, οπότε συγκρίνεται με άλλες και μελετάται η επιλογή εναλλακτικών λύσεων που θα την βελτιστοποιήσουν. Αφορμή για την κατασκευή αυτή αποτελεί η φιλοδοξία των ερευνητών Michael Cullen και συνεργατών (2013), με σχετική δημοσίευσή τους για την ανάπτυξη συστήματος οδήγησης για ρομπότ (Optimal Driveline Robot Base – ODRB), το οποίο θα έχει σταθερά επίπεδα ελέγχου σε υψηλές ταχύτητες, καλή ικανότητα ελιγμών σε χαμηλές ταχύτητες και παράλληλα καλή ενεργειακή απόδοση με τον περιορισμό της ολίσθησης [25]. Ένας επιπλέον στόχος είναι να πληρούνται ταυτόχρονα συγκεκριμένα κριτήρια από τον διαγωνισμό FIRST Robotics Competition.

Όσο αφορά την διάταξη Ackermann, που θα υλοποιηθεί και σε αυτήν τη Διπλωματική εργασία, αναγνωρίζονται τα πλεονεκτήματα αλλά και οι δυσκολίες. Στην πορεία αυτή αναζήτησης λύσεων, διατάσσονται τρεις [23]:

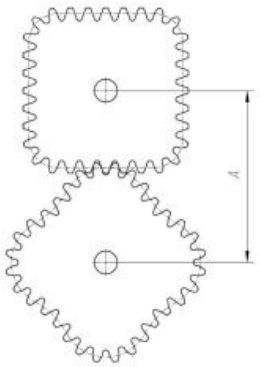
1. Χρήση τραπεζοειδούς σύνδεσης των τροχών αντί για παράλληλης, καθώς η εφαρμογή στην κίνηση θα ταίριαζε περισσότερο (Εικόνα 14Α.)
2. Μη στρογγυλά γρανάζια, με ποικιλία στη διάμετρο βήματος (Εικόνα 14Β.)
3. Σύστημας οδήγησης με κινητήρες να κινούν τους τροχούς (swerve steering)

Η πρώτη επιλογή αποκλείεται σύντομα, αφού ενώ είναι εύκολη στην κατασκευή, δεν είναι δυνατό να εξυπηρετεί επακριβώς την συνθήκη και θα πρέπει να χρησιμοποιηθούν σύνδεσμοι με

προσαρμοσμένα κάθε φορά μήκη. Οι δύο επόμενες, θεωρητικά, θα πληρούν πιστά τις συνθήκες. Όμως, στην περίπτωση των γραναζιών, είναι λίγο δύσκολη η κατασκευή τους ενώ στην τρίτη, θα χρειαστεί ένας κινητήρας για κάθε κατευθυντικό τροχό και θα ήταν πρόκληση η διατήρηση των απαιτούμενων γωνιών στην διάταξη κατά την μεταβολή διεύθυνσης [23].



Εικόνα 14Α. Τραπεζοειδής σύνδεση για Ackermann [23]



Εικόνα 14Β. Παράδειγμα μη-κυκλικών τροχών [23]

2.8 Robot Operating System (ROS)

> Το ROS (Robot Operating System) είναι ένα ευέλικτο σύστημα για τη σύνταξη λογισμικού ρομπότ. Είναι ένα meta-operating σύστημα, δηλαδή δεν παρέχει μόνο τις τυπικές υπηρεσίες λειτουργικού συστήματος, αλλά και λειτουργίες υψηλού επιπέδου (ασύγχρονες και σύγχρονες κλήσεις, κεντρικές βάσεις δεδομένων και συστήματα διαμόρφωσης ρομπότ). Έχει ευρεία εφαρμογή στο χώρο της ρομποτικής γιατί είναι ένα λογισμικό ανοιχτού κώδικα (open source) [26,27].

Στα μοντέρνα ρομποτικά συστήματα συναντώνται περιπτώσεις που θα χρειαστεί να συνυπάρξουν αρκετοί υπολογιστές. Έτσι αναδύεται η ανάγκη για διαχείριση των αισθητήρων και επενεργητών σε συνδυασμό με τις επιμέρους διεργασίες του συστήματος. Ως συνέπεια, τα ρομπότ θα πρέπει να μπορούν να επικοινωνούν και να συνεργάζονται μεταξύ τους. Για να γίνει αυτή η επικοινωνία διεργασιών από έναν υπολογιστή σε έναν άλλον, απαιτείται το ROS, το οποίο παρέχει αυτή την δυνατότητα [28].

Συγκεκριμένα, αυτό που προσφέρει το ROS είναι ένα σύνολο βιβλιοθηκών με λύσεις, προκειμένου να διευκολύνει τον εκάστοτε χρήστη σε όποια υλοποίηση, χωρίς να χρειαστεί να την σχεδιάσει από την αρχή για οποιοδήποτε σύνολο συστημάτων. Το ROS το επιτυγχάνει αυτό με:

- Τα τυπικά πακέτα (standard packages)
- Με την διεπαφή των ROS messages, χάρη στην οποία το λογισμικό μπορεί να υποστηρίζει κάθε μεταβολή του hardware.

Γενικότερα, μπορεί να χρησιμοποιηθεί στα σχέδια του κάθε ρομπότ. Διαθέτει λειτουργίες που μπορούν να ορίσουν τον βαθμό αυτονομίας στην κίνηση ενός ρομπότ, όπως την δημιουργία χαρτών, χειρισμό βάθους χάρτη και εντοπισμού εμποδίων. Ένα ακόμη θετικό στοιχείο, είναι πως υπάρχει ευελιξία στην επιλογή γλώσσας προγραμματισμού για τον πηγαίο κώδικα. Αν και η γραφή αυτού μπορεί να είναι λίγο χρονοβόρα και περίπλοκη, το ROS προσφέρει κάποια έτοιμα πακέτα, όπως για την κίνηση. Οι κύριες διαθέσιμες γλώσσες είναι η Python, C++ και Lisp. Επιπρόσθετα, διατίθονται δοκιμαστικά και βιβλιοθήκες για γλώσσες όπως Java και Lua [26].

Ακόμη ένα προτέρημα του ROS, είναι η υποβοήθηση στην διαδικασία του testing. Αυτή αφορά τον έλεγχο της απόδοσης ενός ρομπότ, όσο αφορά την κίνησή του. Για την παροχή αυτών των δεδομένων, το ROS αναλαμβάνει την διάκριση του υψηλού και χαμηλού συστήματος ελέγχου. Με αυτόν τον τρόπο, επιτρέπεται η αντικατάσταση τμήματος του χαμηλού επιπέδου

χρησιμοποιώντας προσομοίωση και ελέγχοντας μόνο τις λειτουργίες του υψηλού. Έτσι αποφεύγεται η επαναλαμβανόμενη λειτουργία είτε τροχών και μοτέρ ή και των αισθητήρων, προλαμβάνοντας βλάβες. Επιπλέον, πέρα από τον προσομοιωτή, μπορεί κανείς να καταγράφει και αποθηκεύει με το ROS, τα δεδομένα που προέρχονται από τους αισθητήρες. Με αυτά, θα πραγματοποιούνται μετρήσεις και παραμετροποιήσεις ανάλογα με το πώς θα επιλέξει να τα επεξεργαστεί ο χρήστης, βελτιστοποιώντας την τελική απόδοση [28].

2.8.1 Βασικές λειτουργίες του ROS

- Σύστημα ανταλλαγής μηνυμάτων: Ένα σύστημα επικοινωνίας είναι από τους θεμελιώδεις λίθους κατά την σχεδίαση ενός ρομπότ, καθώς θα επηρεάζει άμεσα το παραγώμενο αποτέλεσμα. Το ROS διευκολύνει την διαδικασία αυτή με το σύστημα δημοσίευσης και εγγραφής [27, 29]
- Σύστημα δημοσίευσης και εγγραφής: Σε ένα πρώτο επίπεδο, αποθηκεύονται τα δεδομένα από έναν αισθητήρα, ώστε στην συνέχεια αυτά να υπεσέλθουν σε επεξεργασία. Το ROS αναλαμβάνει την απρόσκοπτη και αποτελεσματική αποθήκευση των εισερχόμενων από τους αισθητήρες δεδομένων σε ένα αρχείο για να αναδημοσιευθούν όταν χρειαστούν (εντολή rosbag). Επιπλέον, καθιστάται εύκολος ο εντοπισμός και η κατανόηση λαθών, αν αυτά προκύψουν
- Περιγραφή και μοντελοποίηση του ρομπότ: Για αυτή την διαδικασία, χρησιμοποιούνται συγκεκριμένα εργαλεία για τις φυσικές ιδιότητες, ώστε να είναι κατανοητά από το ROS (tf, robot_state_publisher, rviz). Η περιγραφή γίνεται με URDF (Unified Robot Description Format) σε XML αρχείο. Όταν αυτή ολοκληρωθεί, επιστρατεύεται η βιβλιοθήκη tf για την μοντελοποίηση
- Κατάσταση του ρομπότ: Υπάρχει η δυνατότητα να συλλέγονται, με τυπικό τρόπο, βασικά διαγνωστικά στοιχεία, ώστε να εκτιμάται η κατάσταση του ρομπότ σε κάθε δεδομένη στιγμή.
- Απομακρυσμένη επικοινωνία: Σε αυτή τη περίπτωση, το λογισμικό ROS λειτουργεί ως διαμεσολαβητής στα αιτήματα του συστήματος, με αποκρίσεις, χρησιμοποιώντας υπηρεσίες [27, 29]
- Εκτίμηση θέσης και πλοήγηση: Υποστηρίζεται, χάρη σε κάποια πακέτα, ο εντοπισμός θέσης του ρομπότ, η δημιουργία χάρτη και η πλοήγηση μέσω κινητού.
- Παρακολούθηση διαδικασίας: Ως ενέργειες, ορίζεται το σύνολο των υπηρεσιών που αξιοποιούνται με την παρακολούθηση του ρομπότ. Μια ενέργεια είναι μια ισχυρή

δυνατότητα του οικοσυστήματος ROS. Ενδεικτικά, είναι δυνατό κανείς να καθοδηγήσει ένα ρομπότ σε κάποια τοποθεσία, να παρακολουθήσει την πρόοδό του, να κρίνει αν θα το σταματήσει ή θα του αλλάξει την πορεία και να ελέγξει πότε έχει ολοκληρωθεί η διαδικασία ή όχι.

- Καθολική βάση δεδομένων: Επιτρέπεται η διαμοίραση και η συνεργασία με άλλους χρήστες, μέσω ενός παγκόσμιου «συννέφου» αποθήκευσης τιμών-κλειδιά. Οι όποιες ρυθμίσεις μπορούν να τροποιηθούν για να βελτιστοποιηθεί η ποιότητα του τελικού αποτελέσματος [27, 29]

Επιπλέον, αξίζει να παρατεθούν οι δομικές έννοιες του ROS και βασικές εντολές στον τερματικό (terminal), καθώς και το πως λειτουργούν στην ολότητα ενός συστήματος. Αυτές χρησιμοποιούνται και στην πλειοψηφία τους στη συγκεκριμένη εργασία [27, 28].

1. **ROS Basics**

- Nodes: Πρόκειται για ένα πρόγραμμα που αναλαμβάνει την διαχείριση μιας λειτουργίας (π.χ εντοπισμός θέσης, διαχείριση αισθητήρων). Σε ένα ρομποτικό σύστημα μπορούν να δημιουργηθούν αρκετά nodes, τα οποία θα χρειαστεί να επικοινωνήσουν τις πληροφορίες που έχουν υπό τον έλεγχό τους. Όλα τα επιμέρους nodes διαχειρίζονται απο το ROS master node, το οποίο επιτρέπει την αναγνώριση των nodes μεταξύ τους. Η επικοινωνία πραγματοποιείται μέσω topics και services.
- Messages: Τα μηνύματα αποθηκεύουν δομές δεδομένων, ώστε αυτά να αποσταλούν είτε μέσω των topics ή services. Συμπεριλαμβάνουν όλους τους «standard primitive τύπους (integer, floating point, Boolean κ.α)». Αν το επιθυμεί ο χρήστης, υπάρχει ευελιξία ως προς την δημιουργία νέου τύπου μηνύματος, με την σύνταξη των απαραίτητων msg files.
- Topics: Σκοπός των topics είναι η διασύνδεση των nodes μεταξύ τους. Αυτό προϋποθέτει τον ορισμό της κατεύθυνσης της επικοινωνίας. Δηλαδή, αν ένα node θέλει να αποστείλει ένα μήνυμα, θα πρέπει να το δημοσιοποιήσει (publish) μέσω ενός topic και το άλλο node να το εγγράψει (subscribe). Για να γίνει αυτό, θα πρέπει προγραμματιστικά να έχει οριστεί μια συνάρτηση callback στο δεύτερο node, ώστε να γνωρίζει πότε θα δεχθεί ένα μήνυμα.

- Services: Πέρα από το σύστημα publish-subscribe που διενεργείται μέσω των topics, ένα ακόμη σύστημα που προσφέρεται είναι το request-reply. Εξυπηρετεί διεργασίες με το RPC πρωτόκολλο μηνυμάτων (Remote Procedure Call). Αποστέλλεται ένα ζεύγος μηνυμάτων, ένα με το εκάστοτε αίτημα και ένα για την απάντηση-ικανοποίηση αυτού.

2. ROS Commands

- Roscore: Εκκινεί και θέτει σε ετοιμότητα όλες τις βασικές παραμέτρους για την λειτουργία του συστήματος (Master, πακέτα κ.α)
- Rosrun: Τρέχει ένα εκτελέσιμο πρόγραμμα και ενεργοποιεί τα nodes
- Rosnode: Προσφέρει πληροφορίες για τα nodes και εμφανίζει την λίστα αυτών
- Rostopic: Προσφέρει πληροφορίες για τα topics
- Rosmsg: Προσφέρει πληροφορίες για τους τύπους μηνυμάτων
- Rosservice: Προσφέρει πληροφορίες για τα services και επιτρέπει να παρακολουθούνται τα μηνύματα που αποστέλλονται
- Rosparam: Χρησιμοποιείται για να πάρουμε και να θέσουμε παραμέτρους (data) που χρησιμοποιούνται απο τα nodes

Ενδεικτικά, η εκκίνηση ενός node πραγματοποιείται ως εξής με δύο εναλλακτικές:

1. Δίνουμε την εκκίνηση σε ένα terminal με την εντολή roscore. Έπειτα, σε νέο τερματικό:

rosrun <package> <node name>

2. Με launch αρχεία, τα οποία βρίσκονται σε μορφή XML και μπορούμε να ενεργοποιούμε πολλά nodes μαζί:

roslaunch <package> <launch file name>

2.8.2 Εργαλεία Ανάπτυξης

Ένα από τα πιο δυνατά χαρακτηριστικά του ROS είναι το ισχυρό σύνολο εργαλείων ανάπτυξης. Τα εργαλεία αυτά υποστηρίζουν την ενδοσκόπηση, τον εντοπισμό σφαλμάτων, τη σχεδίαση και την απεικόνιση της κατάστασης του συστήματος που αναπτύσσεται.

1. **Εργαλείο rviz**

Ένα αρκετά γνωστό εργαλείο στο ROS είναι το rviz, το οποίο προσφέρει τρισδιάστατη οπτικοποίηση αρκετών τύπων δεδομένων και αισθητήρων. Το rviz μπορεί να απεικονίσει πολλούς από τους κοινούς τύπους δεδομένων - μηνυμάτων που υπάρχουν στο ROS, όπως σάρωση λέιζερ, και εικόνες κάμερας. Η οπτικοποίηση όλων των δεδομένων στην ίδια εφαρμογή δεν είναι απλά συναρπαστική, αλλά μας δίνει την δυνατότητα να παρακολουθήσουμε αυτό που βλέπει το ρομπότ και να εντοπίσουμε προβλήματα, όπως οι εσφαλμένες ευθυγραμμίσεις αισθητήρων ή οι ανακρίβειες του μοντέλου ρομπότ [27,29].

2. **Rqt**

Το ROS διαθέτει επίσης rqt, ένα πλαίσιο ανάπτυξης γραφικών διεπαφών για τα ρομπότ. Επιτρέπει την δημιουργία προσαρμοσμένων διεπαφών συνθέτοντας και διαμορφώνοντας την εκτεταμένη βιβλιοθήκη ενσωματωμένων προσθηκών rqt σε καρτέλες, διαχωρισμένη οθόνη και άλλες διατάξεις. Επιπλέον, προσφέρεται για εισαγωγή νέων στοιχείων διεπαφής, συνθέτοντας εξατομικευμένα πρόσθετα rqt. Με την προσθήκη rqt_plot, μπορεί κανείς να παρακολουθεί κωδικοποιητές, τάσεις ή οτιδήποτε μπορεί να αναπαρασταθεί ως αριθμός που μεταβάλλεται με την πάροδο του χρόνου. Η προσθήκη rqt_plot βοηθά στην επιλογή του παρασκηνίου σχεδίασης (π.χ. matplotlib, Qwt) που επιθυμεί να προσομοιώσει ο εκάστοτε χρήστης. Για καταγραφή δεδομένων και αναπαραγωγή, το ROS χρησιμοποιεί το bag format. Εδώ μπορούν να δημιουργηθούν και να υπεσέλθουν υπό επεξεργασία αρχεία γραφικών με την προσθήκη rqt_bag. Όμοια με την εντολή που δημιουργεί το αρχείο αποθήκευσης πληροφοριών, rosbag, αυτή μπορεί να εγγράψει δεδομένα σε "σάκους", δηλαδή το σύνολο των δεδομένων που προέκυψαν, να αναπαράγει επιλεγμένα στοιχεία από εκεί και να απεικονίσει το περιεχόμενο, συμπεριλαμβανομένης της εμφάνισης εικόνων και της γραφικής παράστασης των αριθμητικών τιμών με την παρέλευση του χρόνου [27,29].

2.9 Ερευνητικές εργασίες ανάπτυξης συστήματος με ROS και Ackermann

Οι χρήσεις του ROS είναι σχεδόν καθολικές, χάρη στην ευελιξία του και την ποικιλία των εφαρμογών που είναι ικανό να υποστηρίξει. Παραπάνω παρατέθηκαν κάποια βασικά

χαρακτηριστικά και σε συνδυασμό με τις δυνατότητες που παρέχει, θα αναζητηθούν σχετικές εφαρμογές του. Αυτές, πέρα από την ανάδειξη της σημαντικότητας του ROS, θα αποτελέσουν αντικείμενο έμπνευσης και κατευθυντήριες σε αυτή τη μελέτη.

2.9.1 Ρομπότ Αναψυχής

Ως ένα φιλόδοξο εγχείρημα, μια ομάδα ερευνητών οραματίζεται την χρήση των ρομπότ ως βοηθητικών μηχανισμών σε συγκεντρώσεις ανθρώπων για αναψυχή και διασκέδαση [30]. Συγκεκριμένα, το σχέδιό τους περιλαμβάνει ένα κινούμενο ρομπότ, τρεις υπολογιστές και ένα τάμπλετ. Επιστρατεύουν ρομποτική cloud και το πλαίσιο που προσφέρει το ROS, για να επιτύχουν την επικοινωνία μεταξύ αυτών των συσκευών αλλά παράλληλα εξοικονομούν πόρους και ελαττώνουν το κόστος. Στους στόχους για τις λειτουργίες του ρομπότ διακρίνονται:

- Η περιπλάνηση: η μετακίνηση από ένα προκαθορισμένο σημείο (checkpoint) σε ένα άλλο
- Ο εντοπισμός πλήθους: έλεγχος από το σημείο στάσης για κόσμο
- Η μετακίνηση προς το πλήθος: να βρεθεί κοντά στο πλήθος
- Η ετοιμότητα αλληλεπίδρασης με το πλήθος: να είναι αρκετά κοντά, ώστε να λαμβάνει εντολές
- Η αλληλεπίδραση: να βγάζει φωτογραφίες και να παρέχει πληροφορίες με φωνητική εντολή
- Η επιστροφή: να μεταβαίνει στο τελευταίο προκαθορισμένο σημείο που είχε επισκεφτεί

Τα στοιχεία αναψυχής σε αυτό το πείραμα περιλαμβάνουν την αναγνώριση ομιλίας, τη λήψη εικόνων και κοινοποίησή τους σε μέσα κοινωνικής δικτύωσης και την αναπαραγωγή τραγουδιών, ενώ στο κομμάτι της αλληλεπιδραστικότητας, εμπεριέχεται η ικανότητα επικοινωνίας και ο εντοπισμός πλήθους.

Το ROS αποτελεί τον συνδετικό κρίκο στην επικοινωνία των επιμέρους συσκευών, που τελούν ξεχωριστά προγράμματα έκαστα και όλα αυτά την ίδια χρονική στιγμή. Ως φυσική πλατφόρμα χρησιμοποιείται το «PeopleBot» της «Adept MobileRobots». Κάποιοι από τους υπολογιστές και επεξεργαστές δεν βρίσκονται πάνω στο ρομπότ, αλλά αλληλεπιδρούν με τα δεδομένα που συλλέγει και αναπαράγει μέσω της cloud ρομποτικής και του ROS [30].

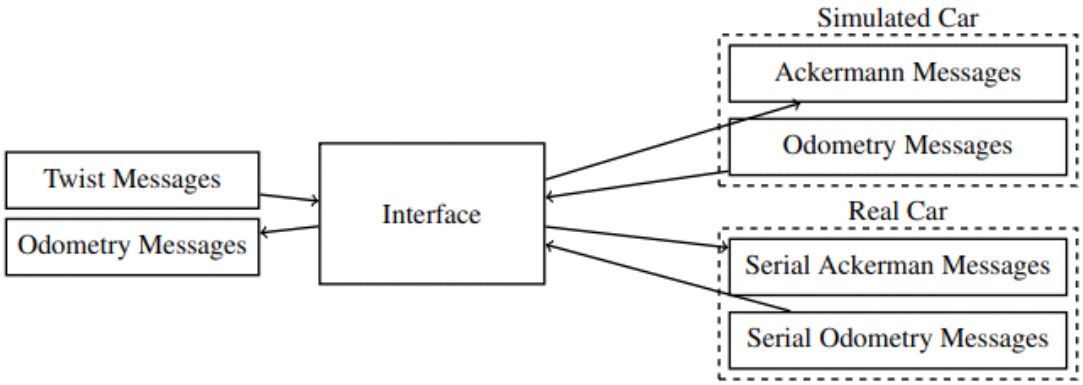
2.9.2 Μηχανισμός παρακολούθησης και ελέγχου για ένα UGV

Αυτή η εργασία παρουσιάζει έναν μηχανισμό παρακολούθησης και ελέγχου για ένα σύνολο UGV με την ενσωμάτωση του ROS. Ο γενικός σκοπός αυτής της εργασίας είναι η δημιουργία ρομπότ με διάταξη Ackermann για τη διεξαγωγή μελετών στον τομέα της αυτόνομης οδήγησης. Είναι εξοπλισμένο με πίνακα Arduino για τον έλεγχο των ενεργοποιητών του οχήματος και ένα Raspberry Pi για τη φιλοξενία του διακομιστή ROS. Το ρομπότ βασίζεται σε αγωνιστικό αυτοκίνητο RC με τιμόνι Ackermann. Κατά την διαδικασία σχεδίασης, επιστρατεύτηκε προσομοίωση, καθώς κρίθηκε απαραίτητη για τη μοντελοποίηση του πρότυπου αυτοκινήτου στις δοκιμές. Ο υπολογισμός και ο έλεγχος του ρομπότ επιτυγχάνεται μέσω του Raspberry Pi και ενός Arduino Uno. Επιπλέον, το όχημα προσομοιώνεται με το Gazebo [31].

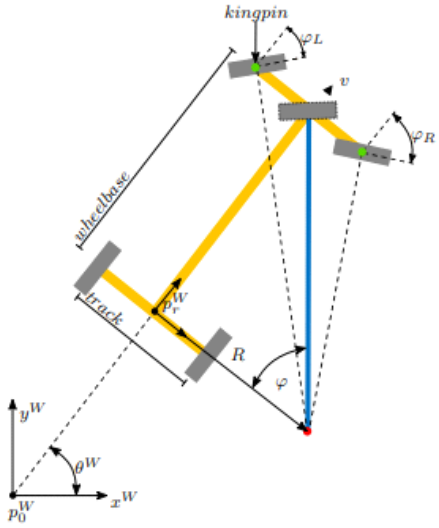
Αναφέρεται πως η επιλογή του ROS αποτελεί κλειδί για τον συνδυασμό πακέτων, όπως είναι αυτά της πλοήγησης. Τα Ackermann μηνύματα σχετικά με τον έλεγχο του ρομπότ και την οδομετρία μεταφράζονται στο ROS, ώστε να παρακολουθείται κάθε μεταβολή και να εκτελούνται οι κατάλληλες κινήσεις. Έτσι, όλα αυτά τα μηνύματα, είναι προσβάσιμα σε μία μοναδική διεπαφή.

Η διεπαφή μετατρέπει τα twist messages (γραμμική και γωνιακή ταχύτητα), σε μηνύματα Ackermann. Παρακάτω δίνεται η δομή της διεπαφής (Εικόνα 15). Τα twist messages χρησιμοποιούνται συνήθως ως εντολές κίνησης, επειδή μπορούν να αποθηκεύσουν έξι παραμέτρους και παρέχουν αρκετές πληροφορίες για τον καθορισμό κινήσεων στον τρισδιάστατο χώρο. Τα μηνύματα Ackermann μεταφέρουν πληροφορίες για την ταχύτητα, την επιτάχυνση και το όποιο τράνταγμα. Η γεωμετρία του οχήματος βοηθά στον υπολογισμό της γωνίας διεύθυνσης που δίνεται στα μηνύματα Ackermann [31].

Το σύστημα παρακολούθησης βασίζεται στο μοντέλο κίνησης ταχύτητας που λαμβάνει υπόψη τα τυχόν σφάλματα. Αν και αυτό που χρησιμοποιείται σε αυτήν την εργασία, έχει σχεδιαστεί για ρομπότ που είναι σε θέση να περιστρέφονται γύρω από τον δικό τους άξονα, τροποποιείται εύκολα για να εξυπηρετήσει ρομπότ με την διάταξη Ackermann. Το μοντέλο κίνησης υπολογίζει την θέση του ρομπότ και τα περιθώρια σφαλμάτων με βάση τις εντολές κίνησης. Οπτικά, εφαρμόζεται σε έναν επίπεδο χώρο που αντιπροσωπεύεται από x και y και την παράμετρο θ, που επιδεικνύει τον προσανατολισμό των ρομπότ (Εικόνα 16).



Εικόνα 15. Η αλληλεπίδραση των διαφορετικών μηνυμάτων της διεπαφής [31]



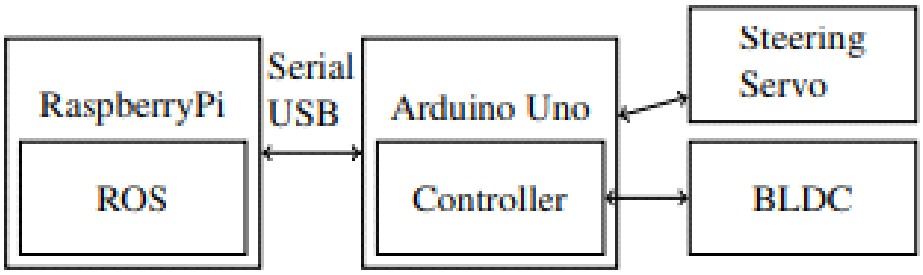
Εικόνα 16. Η γεωμετρία του ρομπότ Ackermann στον δισδιάστατο χώρο [31]

Για την εφαρμογή, ένα αυτοκίνητο Tamiya RC-race σε κλίμακα 1:10, χρησιμοποιείται ως βασικό πλαίσιο για το ρομπότ Ackermann (Εικόνα 17). Το λειτουργικό σύστημα είναι το Raspbian, αυτό που βρίσκεται στα Raspberry Pi, επειδή βασίζεται στο Debian, το οποίο με την σειρά του υποστηρίζει ROS. Όσο αφορά την τροφοδοσία, το όχημα διαθέτει έναν κινητήρα BLDC (Brushless Direct Current) και ένας σερβοκινητήρας χρησιμοποιείται για πηδαλιούχηση. Επιπρόσθετα, τοποθετείται ένας μικροελεγκτής Arduino Uno, λόγω της παροχής των δυνατοτήτων του σε πραγματικό χρόνο και του ειδικού υλικού του. Ένα stick W-LAN είναι εγκατεστημένο στο Raspberry Pi, για να παραχωρεί πρόσβαση από άλλους σταθμούς εργασίας. Το Arduino Uno αναλαμβάνει κάποιες από τις λειτουργίες χαμηλού επιπέδου, δηλάδη είναι

υπεύθυνο για την ανάγνωση αισθητήρων, τον έλεγχο του αυτοκινήτου και παρουσίαση των δεδομένων με χρήσιμο τρόπο. Τα σειριακά μηνύματα από τη διεπαφή είναι τα μέσα επικοινωνίας μεταξύ του Arduino Uno και του Raspberry Pi. Ο υπολογισμός της πόζας και η συνδιακύμανσή της αξιοποιούν μοντέλο κίνησης ταχύτητας που αναφέρθηκε προηγουμένως και πραγματοποιείται από το Arduino Uno [31].

Εικόνα 17. Το πραγματικό ρομπότ (α) και η προσομοίωσή του (β) [31]

Η επαλήθευση και επικύρωση συστημάτων και αλγορίθμων είναι ένα σημαντικό έργο στη ρομποτική κινητών συσκευών. Έτσι, χρησιμοποιήθηκε το Gazebo για την οπτικοποίηση και φυσική προσομοίωση του ρομπότ. Τα απαραίτητα στοιχεία εισάχθηκαν στο Gazebo με URDF.



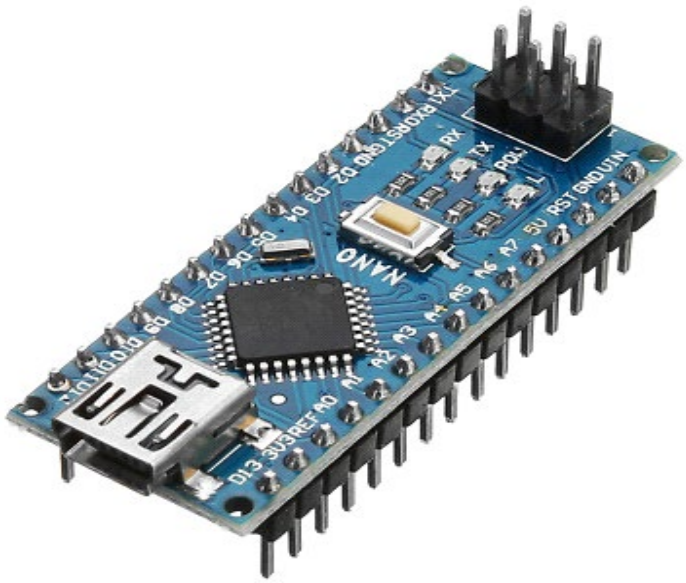
Εικόνα 18. Η ιεραρχία ελέγχου από το ROS στους ενεργοποιητές αυτοκινήτων [31]

ΚΕΦΑΛΑΙΟ 3: Εργαλεία και μέθοδοι ανάπτυξης

Ακολουθεί η παράθεση των εξαρτημάτων και εργαλείων που χρησιμοποιήθηκαν για την ανάπτυξη του ρομποτικού οχήματος, καθώς και η γενική μεθοδολογία που ακολουθείται για την διασύνδεση αυτών.

3.1 Έλεγχος Χαμηλού Επιπέδου

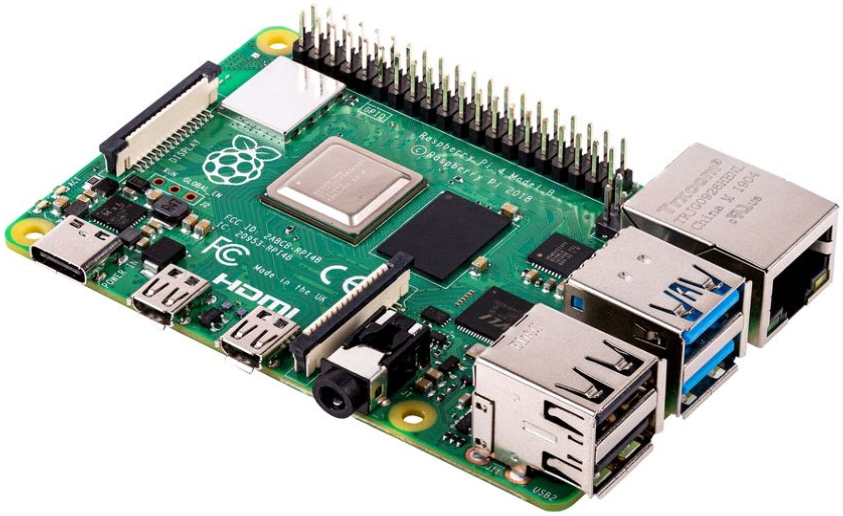
Για τον έλεγχο του χαμηλού επιπέδου χρησιμοποιήθηκε ο μικροελεγκτής Arduino. Στον Arduino συνδέονται διάφορων ειδών αισθητήρες, αναλογικοί αλλά και ψηφιακοί, καθιστώντας το πολύ χρήσιμο στον τομέα της ρομποτικής. Με το Arduino δημιουργούμε εφαρμογές, οι οποίες αλληλοεπιδρούν μεταξύ των υπολογιστών, των χρηστών και του περιβάλλοντος. Ο προγραμματισμός γίνεται με τις γλώσσες προγραμματισμού: C (Wiring) , C++. Στη συγκεκριμένη εργασία χρησιμοποιήθηκε ένας Arduino Nano και προγραμματίστηκε με τη γλώσσα προγραμματισμού C++. Ο Arduino Nano δεν έχει διαφορές με το δημοφιλή Arduino UNO παρά μόνο στο μέγεθος, κάνοντας τον ιδιαίτερα βολικό για κατασκευές, καθώς είναι αρκετά μικρός. Υποστηρίζει 8 αναλογικές εισόδους και 14 ψηφιακές όπου συνδέθηκαν τα 3 αποστασιόμετρα του ρομπότ, το μοτέρ και πλακέτα του που αφορά την κίνηση, το σερβομοτέρ που αφορά την στροφή των τροχών, ο οπτικός αποκωδικοποιητής για την μέτρηση των RPM στροφών του μοτέρ και ένας LED φακός σε περίπτωση που το ρομπότ χρειαστεί να κινηθεί στο σκοτάδι.



Εικόνα 19. ATMEGA328P NANO V3.0 [32]

3.2 Κεντρική Μονάδα επεξεργαστή του Ρομπότ

Για την κεντρική μονάδα επεξεργαστή χρησιμοποιήθηκε ένα Raspberry Pi 4 model B. Το Raspberry Pi είναι ένας υπολογιστής χαμηλού κόστους, μεγέθους πιστωτικής κάρτας που συνδέεται σε οθόνη υπολογιστή ή τηλεόραση και χρησιμοποιεί ένα τυπικό πληκτρολόγιο και ποντίκι. Είναι μια ικανή μικρή συσκευή που επιτρέπει σε κάθε χρήστη να εξερευνήσει τον υπολογιστή και να μάθει πώς να προγραμματίζει σε γλώσσες, όπως τη Python. Έχει τις ίδιες δυνατότητες με έναν επιτραπέζιο υπολογιστή, από την περιήγηση στο διαδίκτυο και την αναπαραγωγή βίντεο υψηλής ευκρίνειας έως την κατασκευή υπολογιστικών φύλλων, επεξεργασίας κειμένου και παιχνιδιών. Επιπλέον, το Raspberry Pi έχει τη δυνατότητα να αλληλεπιδρά με τον έξω κόσμο και έχει χρησιμοποιηθεί σε μια ευρεία γκάμα έργων ψηφιακών δημιουργών, από μουσικές μηχανές έως μετεωρολογικούς σταθμούς. Το Raspberry Pi μπορεί να χρησιμοποιηθεί με μια οθόνη και προσθέτοντας πληκτρολόγιο και ποντίκι, συναρμολογείται με αυτό τον τρόπο ένας πλήρης υπολογιστής, ο οποίος υποστηρίζει συγκεκριμένες διανομές Linux. Άλλες δυνατότητες αξιοποίησης που υποστηρίζει, εξαρτώνται από κάποιες παραμετροποιήσεις. Στο Minecraft, μπορεί να αξιοποιηθεί το Raspberry Pi αποκλειστικά με την ειδική έκδοση Minecraft Pi edition. Ακόμη, κάνοντας σύνδεση έναν εξωτερικό δίσκο, μπορεί να γίνει μετατροπή του Raspberry Pi σε ένα αποκλειστικό σύστημα για λήψη Torrent για όλη την μέρα, χωρίς να χρησιμοποιούνται οι πόροι του υπολογιστή. Μπορεί να χρησιμοποιηθεί επίσης ως εγκέφαλος σε συστήματα οικιακού αυτοματισμού.



Εικόνα 20. Raspberry Pi [33]

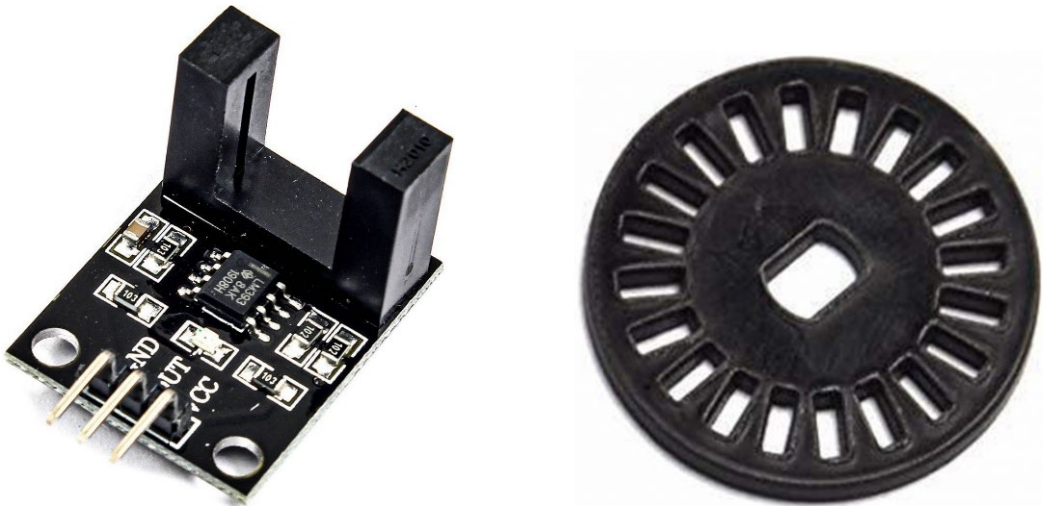
3.3 Αισθητήρας Αποφυγής Εμποδίων

Ο Ultrasonic Sensor HC-SR04 που αξιοποιήθηκε για την εργασία είναι ένας αισθητήρας ο οποίος είναι ικανός να μετρήσει την απόσταση. Εκπέμπει έναν υπέρηχο 40kHz ο οποίος κινείται στον αέρα και αν υπάρχει κάποιο εμπόδιο στην διαδρομή του θα αναπηδήσει πίσω στη μονάδα. Οι αισθητήρες υπερήχων λειτουργούν εκπέμποντας ηχητικά κύματα σε συχνότητα πολύ υψηλή για να τα ακούσουν οι άνθρωποι. Στη συνέχεια περιμένουν την ανάκλαση του ήχου, υπολογίζοντας την απόσταση με βάση τον απαιτούμενο χρόνο. Αυτό είναι παρόμοιο με την μέθοδο που χρησιμοποιεί ένα ραντάρ, μετρώντας δηλαδή το χρόνο που χρειάζεται ένα ραδιοκύμα για να επιστρέψει μετά την ανάκλαση σε ένα αντικείμενο. Για την διπλωματική εργασία χρησιμοποιήθηκαν τρεις ultrasonic sensors, οι οποίοι είναι συνδεδεμένοι με το Arduino, για να μετράνε την απόσταση αριστερά, δεξιά και στην μέση του ρομπότ ώστε να κάνει αποφυγή εμποδίων.

Εικόνα 21. Αποστασιόμετρο υπερήχων HC-SR04 [34]

3.4 Οπτικός Αποκωδικοποιητής

Στην κατασκευή, χρησιμοποιήθηκε ένας οπτικός κωδικοποιητής για την μέτρηση των στροφών RPM του Ρομπότ. Ο οπτικός περιστροφικός κωδικοποιητής είναι μια μηχανική συσκευή που έχει έναν περιστρεφόμενο άξονα στο εσωτερικό του κυλινδρικού περιβλήματος, η κατασκευή μοιάζει με τον κινητήρα. Πρόκειται για έναν κυκλικό επίπεδο δίσκο με δύο σετ υποδοχής. Οι οπτικοί αισθητήρες είναι προσαρτημένοι και στις δύο πλευρές αυτού του δίσκου, ο πομπός στη μία πλευρά και ο δέκτης στην άλλη. Έτσι, όταν ο δίσκος με εγκοπές περιστρέφεται μεταξύ του αισθητήρα, κόβει τον οπτικό αισθητήρα και το σήμα δημιουργείται στα άκρα του δέκτη. Ο δέκτης συνδέεται στην συνέχεια με έναν μικροελεγκτή για την επεξεργασία του παραγόμενου σήματος και με αυτόν τον τρόπο μπορούμε να γνωρίζουμε πόσο περιστρέφεται ο άξονας.



Εικόνα 22. Οπτικός αποκωδικοποιητής [35]

3.5 Κίνηση του Ρομπότ

Για την κίνηση του ρομπότ χρησιμοποιήθηκε ένα Brushed motor με όνομα Crawler Motor 55t της εταιρίας HPI. Είναι ένα αρκετά δυνατό μοτέρ.

Τα χαρακτηριστικά επίδοσής του είναι:

- RPM 7,2V : 8000, ρεύμα 0,75Amp (Χωρίς φορτίο)
- RPM 7,2V : 6800, ρεύμα 3,35 Amp (Μέγιστη απόδοση)
- Σχεδιασμένο εύρος θερμοκρασίας κατά την λειτουργία: -30 βαθμοί C έως +80 βαθμοί C
- Επίπεδο θορύβου: < 60db

Εικόνα 23. RC4WD 540 Crawler Brushed Motor (55T) [36]

3.6 Στροφή τροχών

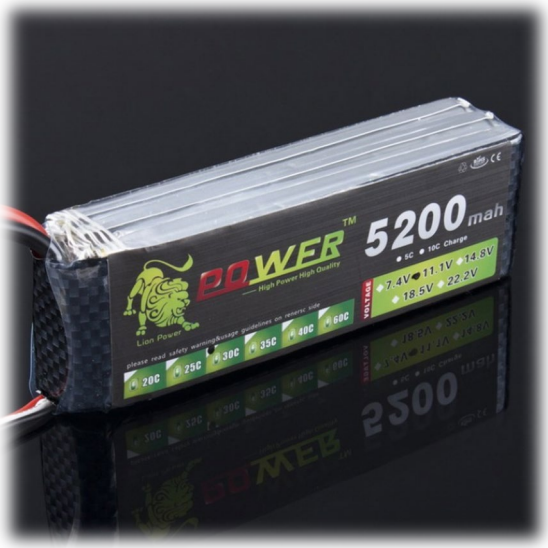
Χρησιμοποιήθηκε 1x servo το οποίο είναι εγκατεστημένο πάνω σε ένα άξονα του οχήματος. Οι τιμές που παίρνει ο άξονας του σερβοκινητήρα σε μοίρες είναι από 0 έως 180. Στις 0 μοίρες του άξονα οι τροχοί είναι στραμμένοι αριστερά. Σε 90 μοίρες του άξονα οι μπροστινοί τροχοί είναι σε ευθεία και σε 180 μοίρες του άξονα οι τροχοί είναι στραμμένοι δεξιά.

Εικόνα 24. Κινητήρας σέρβο [37]

3.7 Ενεργειακή αυτονομία του ρομπότ

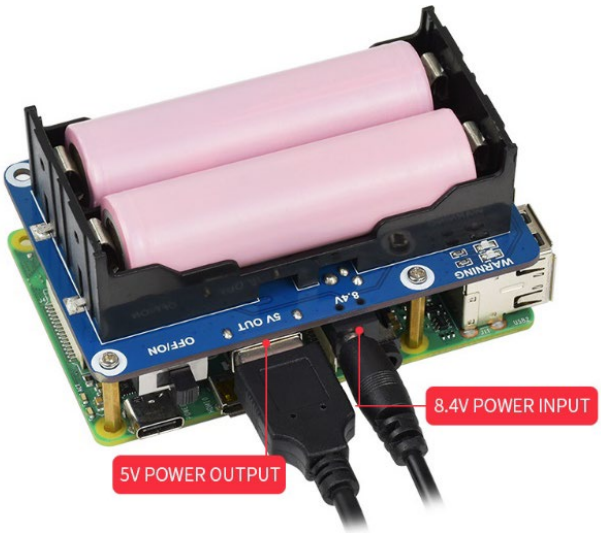
Για το μοτέρ χρησιμοποιείται μπαταρία τύπου LiPo δύο στοιχείων με τάση 7.4V και χωρητικότητα 5200mAh. Για τον σερβοκινητήρα χρησιμοποιήθηκαν δύο επαναφορτιζόμενες μπαταρίες 18650 Li-ion 3500mAh 4.2V. Για το raspberry χρησιμοποιήθηκαν δυο μπαταρίες Li-ion, αλλά για λόγους ασφαλείας και προστασίας της πλακέτας του Raspberry pi, χρειάστηκε και μία πλακέτα παροχής ενέργειας.

Εικόνα 25. Επαναφορτιζόμενη Μπαταρία 18650 Li-ion 3500mAh 4.2V [38]

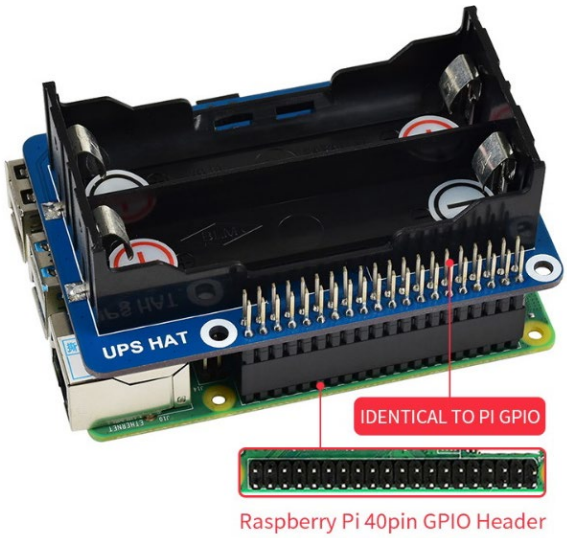


Εικόνα 26. Lion Power 7.4V 5200mAh Lipo Battery [39]

Η είσοδος είναι 8.4V για ένα τροφοδοτικό. Αυτό το τροφοδοτικό μας δίνει την δυνατότητα να φορτίζεται η μπαταρία όσο ταυτόχρονα το Raspberry Pi βρίσκεται σε λειτουργία. Σε περίπτωση που αφαιρεθεί η είσοδος, το Raspberry Pi θα συνεχίζει να λειτουργεί κανονικά λόγω των μπαταριών μέχρι να σταματήσει η ενέργεια αυτών. Όπως παρατηρούμε παρακάτω στην εικόνα 25, μας δίνει την δυνατότητα να χρησιμοποιούμε οι ακροδέκτες του Raspberry Pi χωρίς να τα δεσμεύει. Είναι μία Τυπική κεφαλίδα Raspberry Pi 40PIN GPIO, προσαρμοσμένη για πίνακες της σειράς Raspberry Pi.



Εικόνα 27. Έξοδος USB 5V [40]



Εικόνα 28. Πλακέτα φόρτισης μπαταριών σχεδιασμένη για το Raspberry Pi 4 [40]

3.8 Σύνδεση RPi με Υπολογιστή Εργαστηρίου

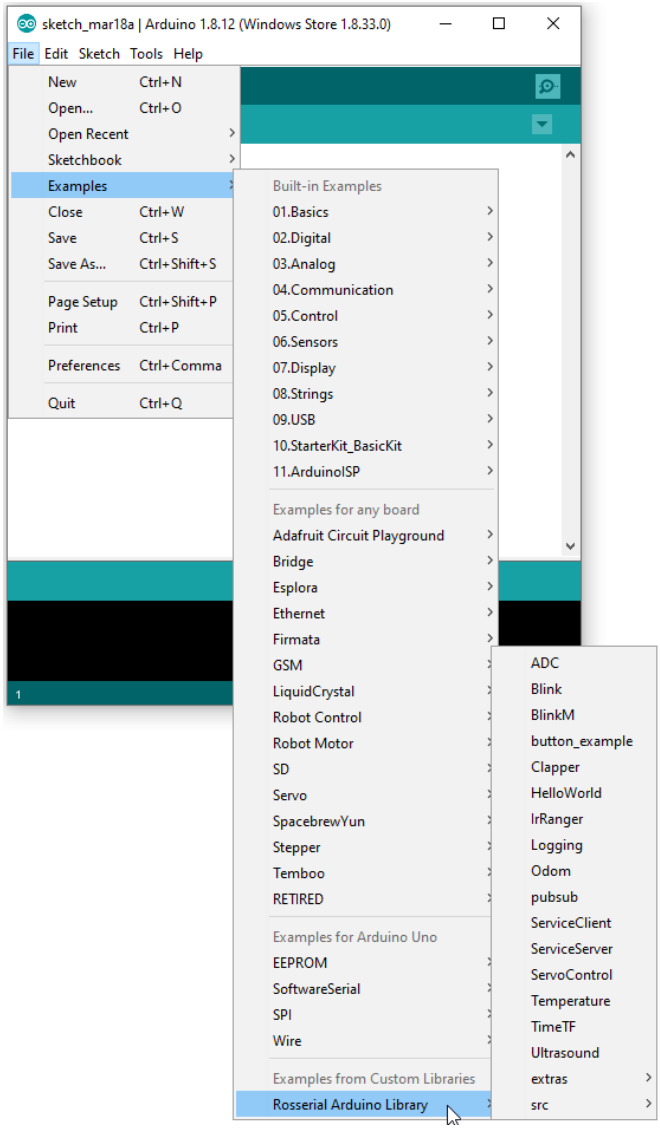
Η επικοινωνία του Raspberry Pi με τον υπολογιστή του εργαστήριου πραγματοποιείται μέσω του τοπικού δικτύου. Το SSH (Secure Socket Shell), είναι ένα πρωτόκολλο δικτύου που παρέχει στους χρήστες έναν ασφαλή τρόπο πρόσβασης σε έναν υπολογιστή μέσω μη ασφαλούς δικτύου. Το SSH επίσης αναφέρεται στη σειρά βοηθητικών προγραμμάτων που υλοποιούν το πρωτόκολλο SSH. Το Secure Shell παρέχει ισχυρό έλεγχο ταυτότητας με κωδικό πρόσβασης και έλεγχο ταυτότητας δημοσίου κλειδιού, καθώς και κρυπτογραφημένες επικοινωνίες δεδομένων μεταξύ δύο υπολογιστών που συνδέονται μέσω ανοιχτού δικτύου, όπως το διαδίκτυο. Εκτός από την ισχυρή κρυπτογράφηση, το SSH χρησιμοποιείται ευρέως από διαχειριστές δικτύου για τη διαχείριση συστημάτων και εφαρμογών από απόσταση, επιτρέποντάς τους να συνδεθούν σε άλλο υπολογιστή μέσω δικτύου, να εκτελέσουν εντολές και να μεταφέρουν αρχεία από τον έναν υπολογιστή στον άλλο. Το Secure Shell δημιουργήθηκε για να αντικαταστήσει ανασφαλή προγράμματα εξομοίωσης ή σύνδεσης τερματικού. Το SSH ενεργοποιεί τις ίδιες λειτουργίες (σύνδεση και εκτέλεση τερματικών συνεδριών σε απομακρυσμένα συστήματα). Αντικαθιστά επίσης προγράμματα μεταφοράς αρχείων, όπως το πρωτόκολλο μεταφοράς αρχείων (FTP) και το rcp (απομακρυσμένο αντίγραφο) [41]. Για την διπλωματική εργασία, αυτό που απαιτείται είναι αρχικά οι δύο συσκευές να ανήκουν στο ίδιο δίκτυο, στη συνέχεια θα χρειαστεί να εντοπιστεί η τοπική διεύθυνση δικτύου και το όνομα χρήστη του RPi και τέλος σε ένα τερματικό να εκτελεστεί η παρακάτω εντολή.

ΕΝΤΟΛΗ SSH

Εάν η σύνδεση πραγματοποιηθεί με επιτυχία, θα εμφανιστεί το όνομα χρήστη του RPi στο τερματικό και θα υπάρχει η δυνατότητα πλοήγησης μέσω του τερματικού. Για την πλοήγηση στο γραφικό περιβάλλον του RPi τρέχει ένας VNC Server και από την πλευρά του υπολογιστή θα χρειαστεί ο περιηγητής του VNC Server που παίρνει ως παράμετρους την τοπική διεύθυνση δικτύου του RPi. Ο VNC Server είναι ένα γραφικό σύστημα κοινής χρήσης επιφάνειας εργασίας που χρησιμοποιεί το πρωτόκολλο Remote Frame Buffer (RFB) για τον απομακρυσμένο έλεγχο άλλου υπολογιστή. Μεταδίδει την είσοδο του πληκτρολογίου και του ποντικιού από τον έναν υπολογιστή στον άλλο, μεταδίδοντας τις ενημερώσεις οθόνης γραφικών, μέσω ενός δικτύου.

3.9 ROS Serial

Για την αξιοποίηση του Arduino με το ROS, είναι απαραίτητη η χρήση του πακέτου serial. Αυτό επιτρέπει την αναγνώριση της συσκευής του Arduino, καθώς περιέχει επεκτάσεις ανεπτυγμένες για αυτό. Έτσι, μπορεί να αναγνωρίζεται και κωδικοποιείται όποιο υλικό συνδέεται με το Arduino (π.χ αισθητήρες). Συγκεκριμένα, το Arduino θα νοείται για το ROS ως node, το οποίο θα αποστέλει και θα εγγράφει δεδομένα (publish – subscribe). Από την πλευρά του μικροελεγκτή, εισάγεται μια βιβλιοθήκη, η ros_lib, αφού έχει προηγηθεί η εγκατάσταση του rosserial. Με την ολοκλήρωση της διαδικασίας, εντοπίζουμε την βιβλιοθήκη στα Παραδείγματα (Εικόνα 29) [42].



Εικόνα 29. Βιβλιοθήκη ros_lib στο περιβάλλον Arduino [42]

3.10 Γραφικό Περιβάλλον Χρήστη

Το Γραφικό Περιβάλλον Χρήστη ή Graphical User Interface (GUI) δημιουργήθηκε με τη βοήθεια του Qt. Το Qt είναι ένα framework το οποίο χρησιμοποιείται για την ανάπτυξη γραφικών διεπαφών χρήστη (GUI) και εφαρμογών πολλαπλών πλατφορμών, που τρέχουν σε όλες τις μεγάλες πλατφόρμες επιτραπέζιων υπολογιστών και στις περισσότερες κινητές ή ενσωματωμένες πλατφόρμες. Το Qt Framework περιέχει ένα ολοκληρωμένο σύνολο ιδιαίτερα διαισθητικών και διαμορφωμένων τάξεων βιβλιοθηκών C++. Το Qt παράγει εξαιρετικά αναγνώσιμο, εύκολα συντηρήσιμο και επαναχρησιμοποιήσιμο κώδικα με υψηλή απόδοση χρόνου εκτέλεσης και μικρό αποτύπωμα. Επίσης, συνεργάζεται άψογα με το ROS καθώς οι ρυθμίσεις για την συνεργασία των δυο frameworks είναι ιδιαίτερα συμβατές.

ΚΕΦΑΛΑΙΟ 4: Εφαρμογή

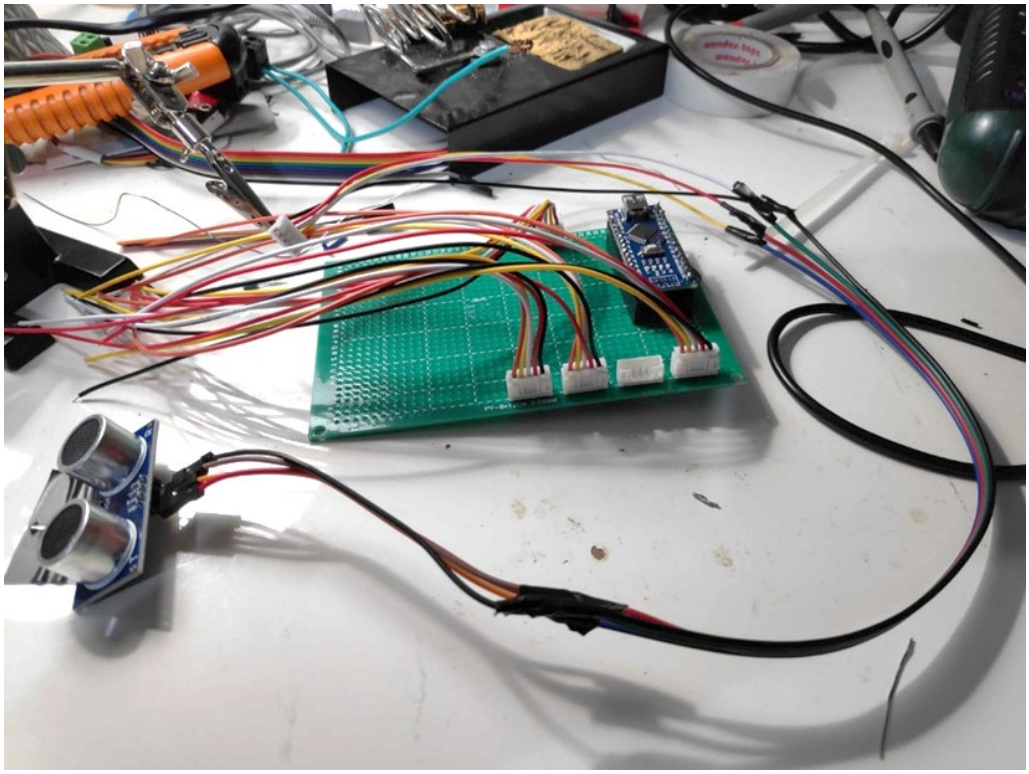
Σε αυτή την ενότητα θα παρατεθεί η διαδικασία σχεδίασης, σύνθεσης του υλικού, ο προγραμματισμός και σχετικό φωτογραφικό υλικό. Αρχικά θα περιγραφεί το ζητούμενο και στην συνέχεια τι χρησιμοποιήθηκε, με ποιον τρόπο και δυσκολίες που ανέκυψαν, μηχανολογικής και προγραμματιστικής φύσεως. Τέλος, αξιολογείται το αποτέλεσμα, σχολιάζονται πιθανές εφαρμογές του ρομποτικού οχήματος και μελετώνται μελλοντικές προεκτάσεις.

4.1 Περιγραφή κατασκευής και μεθοδολογία

Συνοπτικά, για την πραγματοποίηση της εργασίας, χρησιμοποιήθηκε ένα κινητό όχημα με τέσσερις τροχούς τύπου monster truck. (Εικόνα 30)

Εικόνα 30. Monster truck

Για την κίνηση του οχήματος, τοποθετήθηκαν dc μοτέρ και κινητήρες σέρβο, έτσι ώστε να πραγματοποιείται η κίνηση με τις συνθήκες Ackermann. Στη συνέχεια, ενσωματώθηκαν σε αυτό τρεις αισθητήρες υπερήχων για τη μέτρηση της απόστασης και ένας οπτικός κωδικοποιητής για τη μέτρηση των στροφών (/min) του μοτέρ.



Εικόνα 31. Σύνδεση αποστασιόμετρου

Πέρα από τα αμιγώς μηχανικά στοιχεία της κατασκευής, τοποθετήθηκαν και 2 μονάδες ελέγχου αυτών. Συγκεκριμένα, τοποθετήθηκε το Arduino uno, ένας ελεγκτής για «low level control», που συντελεί δηλαδή στη σύνδεση των επιμέρους στοιχείων της κατασκευής όπως προαναφέρθηκαν. Επιπλέον, υπάρχει και το Raspberry Pi, το οποίο ως «high level» ελεγκτής συντονίζει ολόκληρη τη διαδικασία, έχοντας σύνδεση με το Arduino μέσω usb και με τον υπολογιστή μέσω δικτύου.

Έχοντας περιγράψει το βασικό κορμό της κατασκευής, μεταβαίνουμε στο στοιχείο - κλειδί που επιστρατεύεται για τη διευκόλυνση και διασφάλιση της απρόσκοπτης λειτουργίας της. Αυτό είναι η λειτουργία ολόκληρου του συστήματος μέσα στο πλαίσιο του ROS (Robot Operating System). Αναλυτικά, το ROS έχει την δυνατότητα να παρέχει βασικές υπηρεσίες ενός λειτουργικού συστήματος, να χειρίζεται χαμηλού επιπέδου έλεγχο συσκευών, να διαχειρίζεται πακέτα και να ανταλλάσσει μηνύματα μεταξύ των διεργασιών. Το ROS βέβαια από μόνο του δεν αποτελεί λειτουργικό σύστημα πραγματικού χρόνου, οπότε καθιστάται αναγκαίο να ενσωματωθεί κώδικας πραγματικού χρόνου.

Πρόκειται, λοιπόν, για τον μεσολαβητή, που έχει την ευθύνη να χειρίζεται την επικοινωνία μεταξύ των προγραμμάτων στο υπό ανάπτυξη σύστημα. Ακόμη, ένα από τα πλεονεκτήματα τα οποία έχει

το ROS είναι ότι ο κώδικας θα προσαρμόζεται εύκολα και σε πολλά είδη ρομπότ και εκ φύσεως υπάρχουν πολλά πακέτα για εφαρμογές σχετικά με την ρομποτική. Ένα ακόμα θετικό χαρακτηριστικό, είναι ότι μπορεί εύκολα κάποιος να επικοινωνήσει μεταξύ ενός κόμβου Python και ενός κόμβου C++. Έτσι, οι βιβλιοθήκες με τα διαθέσιμα προγράμματα έχουν επεκταθεί δίνοντας την δυνατότητα δημιουργίας πιο σύνθετων ρομπότ. Στην γκάμα των λειτουργιών που προσφέρει το ROS είναι και η πλοήγηση, ανίχνευση εμποδίων και ο τηλεχειρισμός. Επίσης το ROS επιτρέπει τη δημιουργία νέων προσαρμοσμένων λειτουργιών, ανάλογα με την απαίτηση της κάθε συνθήκης. Με αυτόν τον τρόπο, επεκτείνει το χρόνο και τις δυνατότητες των ερευνητών να αναπτύξουν επιπρόσθετες λειτουργίες που μπορεί να χρησιμοποιήσει ο οποιοσδήποτε χρήστης σε αντίστοιχα προβλήματα που θα συναντήσει. Παράλληλα, είναι κατάλληλο για μια εκπαιδευτική διαδικασία, καθώς παρέχει τα βασικά εργαλεία για την ανάπτυξη ενός ρομπότ.

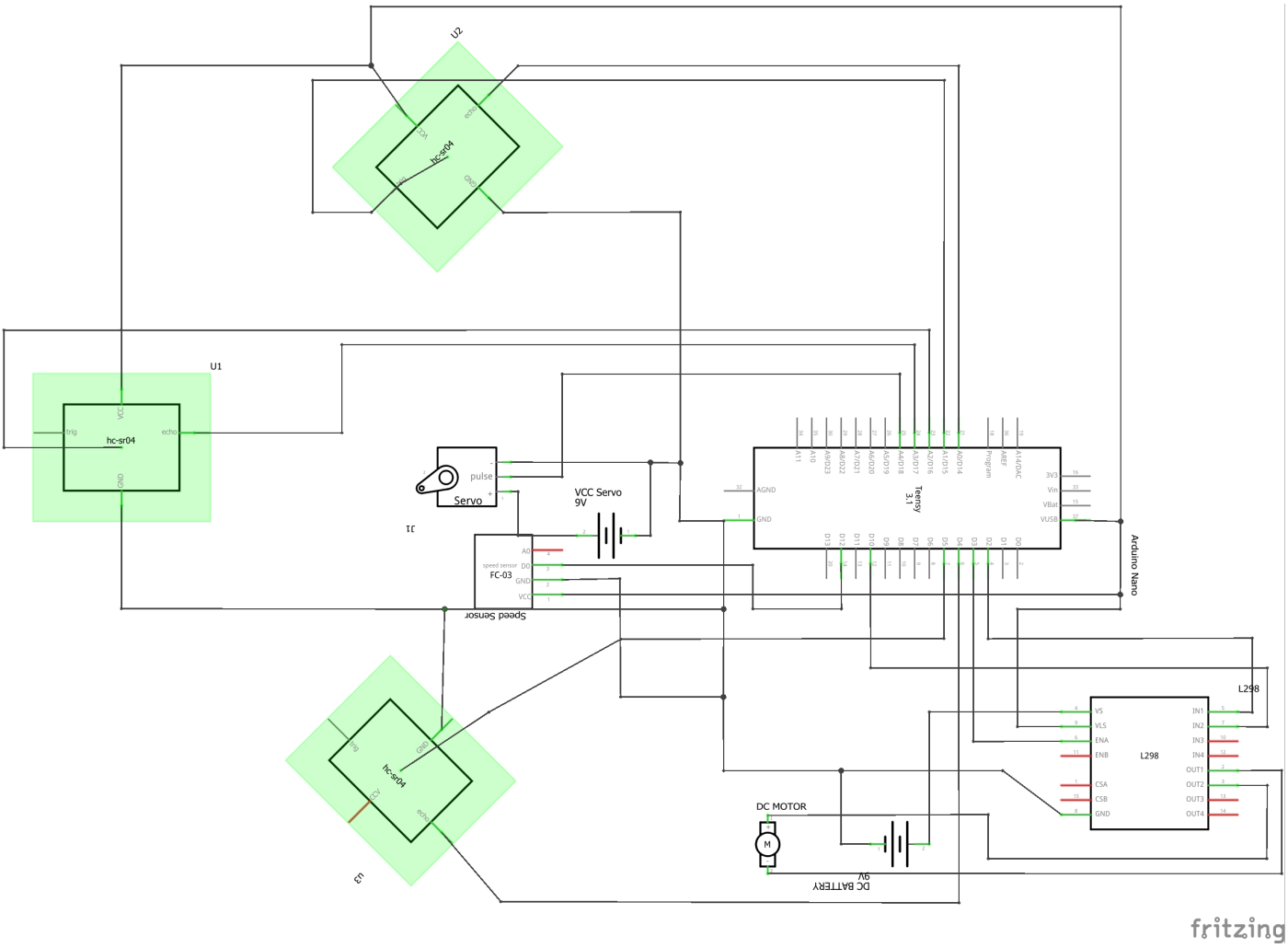
Στη συγκεκριμένη εργασία χρησιμοποιείται η έκδοση ROS melodic, που περάστηκε στον ελεγκτή υψηλού επιπέδου Raspberry Pi με το λειτουργικό σύστημα Ubuntu 18.04.5 LTS (Bionic Beaver). Η γλώσσα προγραμματισμού που επιλέχθηκε για την ανάπτυξη κώδικα και τη χρήση των ήδη υπαρχόντων Open Source πακέτων είναι η C++.

Αρχικά, επιχειρήθηκε η συναρμολόγηση ενός απλού τροχήλατου και ο έλεγχος αυτού με Arduino. Σε πρώτο επίπεδο, διερευνήθηκε το κομμάτι του προγραμματισμού με τον κατάλληλο κώδικα και εν συνεχεία εξετάστηκαν παράγοντες, όπως η ομαλή λειτουργία των επιμέρους εξαρτημάτων και αισθητήρων και η αποτελεσματικότητα στην κίνηση. Σε δεύτερο επίπεδο, προστέθηκε για δοκιμές και το Raspberry Pi, με την χρήση της βιβλιοθήκης της Python, serial, για την επικοινωνία με τον Arduino. Πραγματοποιήθηκαν αρκετές δοκιμές για την κατανόηση της δομής ως προς την κίνηση των δεδομένων, επαληθεύοντας μέσα από απλές, στοιχειώδεις εντολές κίνησης. Ο έλεγχος πραγματοποιήθηκε με το πληκτρολόγιο, κωδικοποιώντας συγκεκριμένα πλήκτρα με τις επιθυμητές κατευθύνσεις (μπροστά - πίσω, δεξιά - αριστερά). Αυτό έγινε με την βοήθεια της βιβλιοθήκης pygame. Ως συμπέρασμα, προέκυψε από την αρχή η συνειδητοποίηση της καθυστέρησης εκτέλεσης εντολών. Για έλεγχο πραγματικού χρόνου, η χρήση μόνο του raspberry pi φαντάζει ιδανικότερη. Στόχος, λοιπόν, είναι η χειροκίνητη κίνηση με την βοήθεια του περιβάλλοντος αλληλεπίδρασης (manual), όσο και η αυτόνομη, με την βοήθεια των αποστασιόμετρων (auto).

Στο επόμενο στάδιο, υλοποιήθηκε το κύκλωμα στο monster truck, ξεκινώντας πάλι μόνο με τον Arduino, τους αισθητήρες και επενεργητές για έναν αρχικό έλεγχο. Με την βιβλιοθήκη του ros

serial, έγινε η εισαγωγή του Raspberry και πραγματοποιήθηκε επικοινωνία μέσω της γραμμής εντολών (command line).

4.2 Συνδεσμολογία



Εικόνα 32. Κυκλωματικό διάγραμμα [43]

Παραπάνω δίνεται το αναλυτικό διάγραμμα του κυκλώματος, προσομοιώνοντας την αντίστοιχη θέση όλων των επιμέρους εξαρτημάτων όπως αυτά τοποθετήθηκαν στο ρομπότ. Αριστερά με πράσινο διακρίνονται τα αποστασιόμετρα, τα οποία αναλαμβάνουν τον έλεγχο της κίνησης μετρώντας αποστάσεις και προειδοποιώντας για τυχόν εμπόδια. Συνδέεται η τάση και το καλώδιο της γείωσης αντίστοιχα, με τα trig και echo των δύο εξ' αυτών να συνδέονται στις θύρες A0 έως A3 του Arduino nano και του τρίτου στην 6 και 7, από την άλλη πλευρά της πλακέτας. Το servo με την σειρά του, λαμβάνει τάση από την μπαταρία li-ion και συνδέεται στην θύρα A4. Τέλος, για

την λειτουργία του dc motor, χρησιμοποιείται η απαραίτητη πλακέτα L298. Αυτή τροφοδοτείται από μια μπαταρία. Συνδέεται με το Arduino και το μοτέρ συνδέεται μόνο στην πλακέτα, στις θύρες OUT. Ανάλογα με την θέση που θα συνδεθούν τα δύο καλώδια του μοτέρ, καθορίζεται και η φορά περιστροφής. Δίπλα στο servo διαφαίνεται ο speed sensor, δηλαδή ο οπτικός αποκωδικοποιητής. Για τον σχεδιασμό του κυκλώματος, αξιοποιήθηκε το πρόγραμμα fritzing [43].

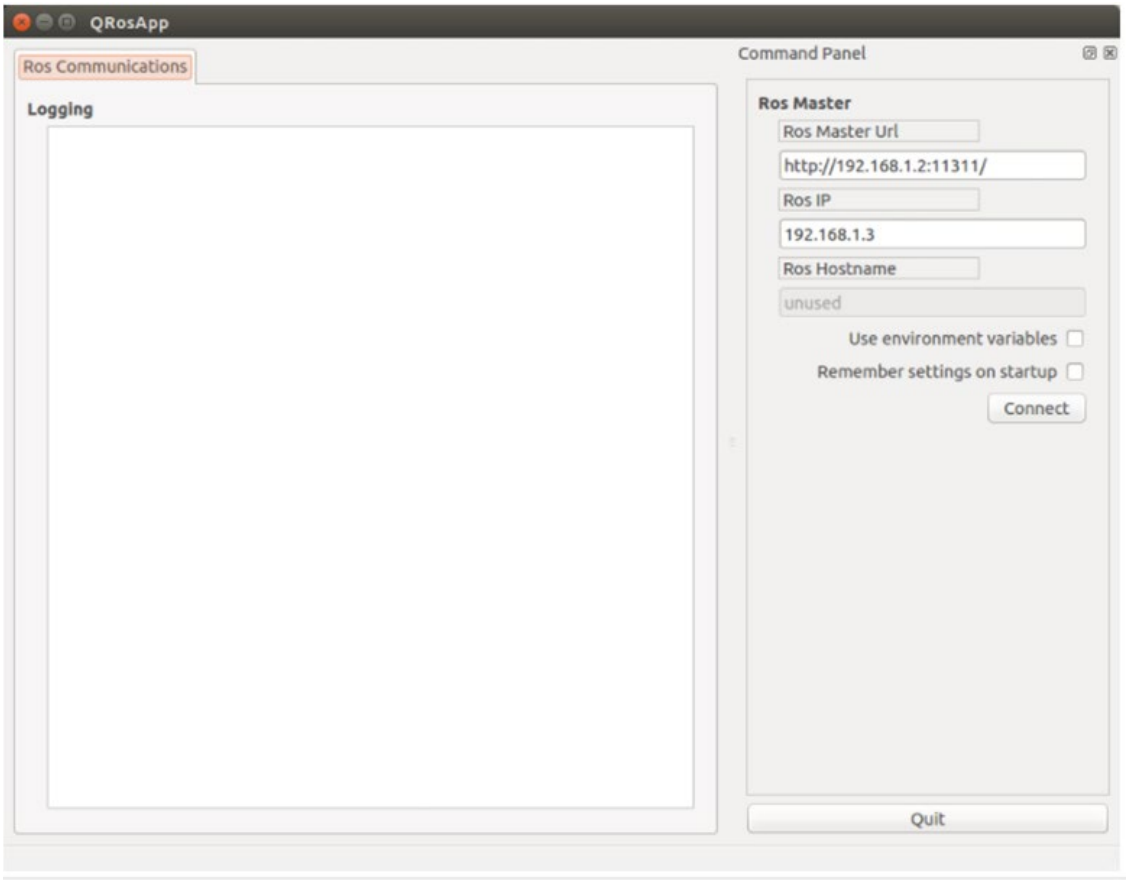
4.3 Κώδικας και γραφικό περιβάλλον

Όσο αφορά τον κώδικα, αυτός διακρίνεται σε μεμονωμένα κομμάτια (scripts), για να εξυπηρετείται η ξεκάθαρη διαρρύθμισή του ως προς τις επιμέρους λειτουργίες και συστήματα δράσης. Ειδικά, ξεκινούν οι δηλώσεις των μεταβλητών που θα χρησιμοποιηθούν και θα κληθούν καθόλη την έκταση του κώδικα. Για παράδειγμα, αναθέτονται ονόματα στα pin του Arduino και γίνεται η δήλωση των αντίστοιχων για το ROS (callbacks). Οι callback συναρτήσεις λειτουργούν ως διαχειριστές μηνυμάτων. Ο χρήστης ορίζει το είδος του μηνύματος που αναμένει και αναθέτει τον έλεγχο σε αυτήν, ώστε όταν προκύψει ειδοποίηση, να μας ενημερώνει. Για παράδειγμα, μέσα στον κώδικα εντοπίζονται οι συναρτήσεις callback_motors και callback_servo. Αυτές αναμένουν στοιχεία για την θέση των μοτέρ και του σερβοκινητήρα, έτσι ώστε αυτά τα δεδομένα να χρησιμοποιηθούν ανάλογα στην συνέχεια του κώδικα από τον Arduino. Ενδεικτικά, ανάλογα με το ποια είναι η θέση του σερβοκινητήρα θα οριστεί η στροφή και επαναφορά του.

Ακόμη, δημιουργήθηκαν τα κομμάτια του κώδικα για το ROS, που ορίζουν την λειτουργία του κάθε node ως publisher/subscriber και ορίστηκαν τα topics που θα συντελούσαν στην επικοινωνία αυτών. Αναλυτικά, στον κώδικα εντοπίζονται αυτές για τον οπτικό κωδικοποιητή και τα αποστασιόμετρα ως publishers και για τα μοτέρ, τους σερβοκινητήρες, το qt και το led ως subscribers.

Στη συνάρτηση setup, όπου αποδίδονται ιδιότητες εισόδων/εξόδων πληροφορίας με την συνάρτηση pinMode, αρχικοποιούνται οι απαραίτητες μεταβλητές (σερβοκινητήρες, αποστασιόμετρα, nodes κ.α) για τον έλεγχο και την εκκίνηση των διαδικασιών. Επιπρόσθετα, με την serial.begin(speed), ορίζεται η ταχύτητα μεταφοράς των πληροφοριών σε bit ανά second (baud rate). Εδώ προτιμήθηκε η τιμή 115200. Το Arduino θα νοείται ως ολότητα ένα node και τα επιμέρους εξαρτήματα και συνδεδεμένοι αισθητήρες με εκείνο ως topics αυτού.

Ακολούθησε η δημιουργία του γραφικού περιβάλλοντος (GUI) για την διευκόλυνση της αλληλεπίδρασης. Όπως προαναφέρθηκε, το ROS προσφέρει το qt. Η εντολή που δίνεται για την δημιουργία του είναι: catkin_create_qt_pkg.

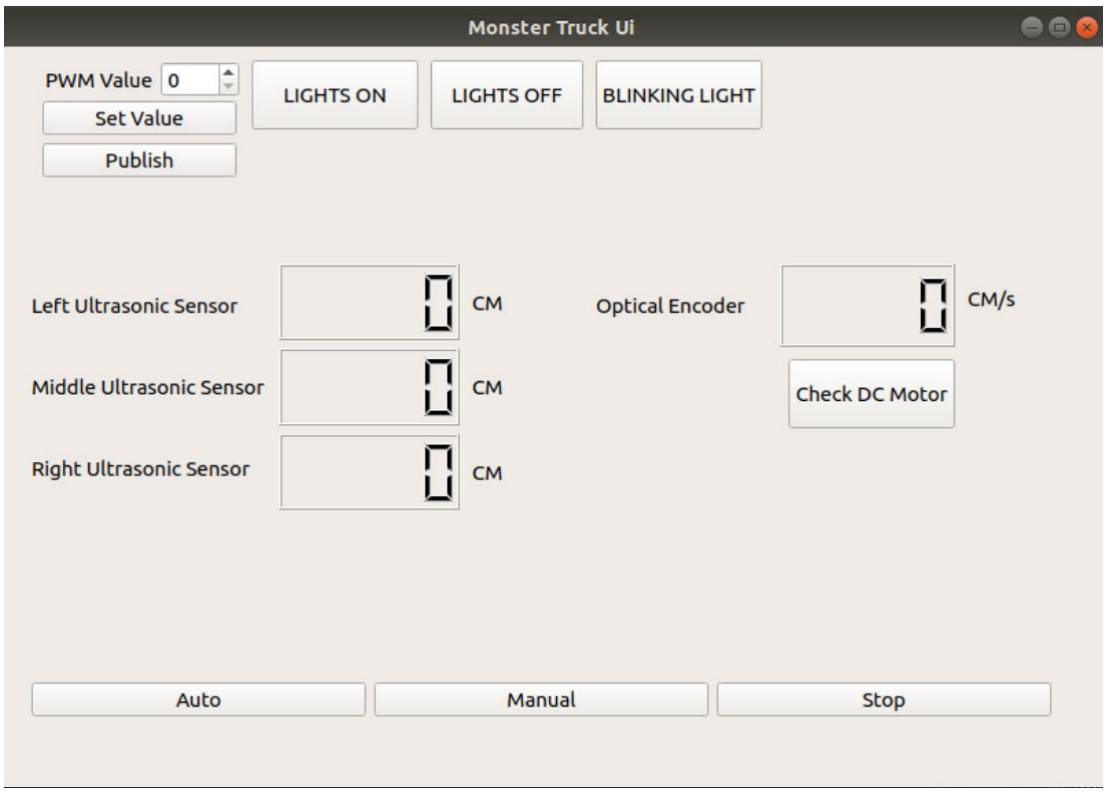


Εικόνα 33. Default γραφικό περιβάλλον ROS [44]

Σε αυτό το σημείο ανέκυψαν δυσκολίες, καθώς ενώ το ROS προσφέρει ένα default qt περιβάλλον, η οποιαδήποτε αλλαγή που θα εξυπηρετούσε την επιθυμητή δομή, δεν γινόταν δεκτή και εμφανίζονταν μηνύματα λάθους. Παράλληλα, έπρεπε να εντοπιστεί η συμβατή έκδοση με αυτή του ros [42]. Για την επίλυση του θέματος, έγιναν αρκετές παραμετροποιήσεις στο ROS app μέσω των αρχείων cmakelists, προκειμένου το περιβάλλον να είναι εκτελέσιμο. Το CMake είναι ένα ανοιχτού κώδικα ελεύθερο λογισμικό και χρησιμοποιείται για τη δημιουργία εκτελέσιμων προγραμμάτων και βιβλιοθηκών από πηγαίο κώδικα (build automation), έλεγχο (testing), πακέτα (packaging) και εγκατάσταση (installation) λογισμικού. Δεν δημιουργεί εν γένει τον κώδικα, αλλά παράγει τα απαραίτητα στοιχεία ενός άλλου συστήματος. Χρησιμοποιεί κομμάτια κώδικα (scripts) στην μορφή Cmakelists.txt. Έγιναν αρκετές δοκιμές και διερευνήσεις για την συγκεκριμένη δραστηριότητα, η οποία αποδείχτηκε χρονοβόρα [45, 46].

Συνοπτικά, ο χρήστης αλληλεπιδρά με το γραφικό περιβάλλον, επιλέγοντας την κατεύθυνση στην οποία θα κινηθεί το ρομπότ. Το δεδομένο αυτό μεταβιβάζεται κωδικοποιημένο στο πρόγραμμα

και ενεργοποιείται η συνάρτηση που διατάσσει την επιθυμητή ενέργεια, δηλαδή προς τα που θα κινηθούν οι σερβοκινητήρες και με ποια ταχύτητα τα μοτέρ. Παράλληλα, τα αποστασιόμετρα επιβλέπουν και ενημερώνουν τον χρήστη για την απόσταση που έχει το ρομπότ από τα εμπόδια μπροστά, δεξιά και αριστερά του. Η ταχύτητα ορίζεται από το qt και αποστέλλεται ως πληροφορία με το κουμπί publish. Καίριο ρόλο στην όλη διαδικασία έχουν οι συναρτήσεις callback, οι οποίες αναμένουν τις πληροφορίες.



Εικόνα 34. Κυρίως παράθυρο γραφικού περιβάλλοντος (qt)

Στην void loop, καλούνται σε επανάληψη οι συναρτήσεις που εμπεριέχονται, αναμένοντας την επαλήθευση κάποιας από αυτές προκειμένου να ακολουθήσει η αντίστοιχη διαδικασία δράσης της. Συγκεκριμένα, ακολουθείται η σειρά ως εξής: auto_GUI, manual_GUI, stop_GUI, check_GUI, led_on_off, nh→spinOnce. Η πρώτη δίνει σήμα για την αυτόνομη πλοήγηση του οχήματος, μόλις γίνει η προκαθορισμένη επιλογή από το γραφικό περιβάλλον (κουμπί auto) και δοθεί ταχύτητα. Αν επιλεχθεί ο χειροκίνητος έλεγχος, εμφανίζονται τα τέσσερα κουμπιά κατεύθυνσης. Στον κώδικα, εφόσον δοθεί τιμή μεγαλύτερη του μηδενός, εκκινεί το όχημα την πορεία του, ανάλογα με το κουμπί που έχει πατηθεί. Για παράδειγμα, ο αριθμός 1 σηματοδοτεί την δεξιά στροφή. Όποια κατεύθυνση και να επιλεχθεί, μετά την ολοκλήρωση της στροφής, οι

ρόδες επαναφέρονται στο αρχικό σημείο, ευθεία, έτοιμες για την επόμενη εντολή κίνησης (myservo.write(90);). Η check_GUI παραδίδει τις μετρήσεις των αποστάσεων από τα εμπόδια που τυχόν υπάρχουν και υπολογίζει τις στροφές των τροχών με την βοήθεια του οπτικού κωδικοποιητή. Αν δεν λαμβάνεται κάποια εντολή, η συνάρτηση stop_GUI ακινητοποιεί το ρομπότ και παράλληλα επιτάσσει και αυτή την μέτρηση των αποστάσεων. Η λειτουργία του led είναι σταθερή καθόλη την διάρκεια του επαναλαμβανόμενου γύρου της loop, ενώ ως τελευταία και ίσως κυριότερη εμφανίζεται η nh→spinOnce. Είναι απαραίτητη όταν θέλει κανείς να εγγράψει μηνύματα, υπηρεσίες ή δράσεις καλώντας τους subscribers [47].

4.4 Εκτέλεση

Τα βήματα που απαιτούνται μετά και την ολοκλήρωση της κατασκευής, είναι η διασύνδεση του ρομπότ μέσω του συστήματος στον υπολογιστή. Αρχικά, έχοντας ανοίξει ένα παράθυρο του τερματικού, πληκτρολογείται η εντολή SSH. Έτσι, όπως περιγράφηκε αναλυτικά στην σχετική ενότητα, εκκινείται η επικοινωνία. Προϋπόθεση είναι να ανήκουν στο ίδιο δίκτυο. Αφού είναι επιτυχής η σύνδεση, στο τερματικό θα εμφανίζεται πλέον η ονομασία του RPi. Ως προαιρετικό και εναλλακτικό βήμα προτείνεται η επιλογή του VNC Viewer, θέτοντάς το δηλαδή ως ένα VNC Server. Για την σύνδεση του ROS με το RPi, με την εντολή sudo nano .bashrc, πραγματοποιούνται οι απαραίτητες αλλαγές στο workstation για τις IP. Έτσι, μπορεί κανείς να ελέγχει μέσω του terminal του υπολογιστή το ρομπότ.

#WORKSTATION CONFIG

export ROS_IP=192.168.1.1 *#IP υπολογιστή*

export ROS_HOSTNAME=192.168.1.1 *#IP υπολογιστή*

export ROS_MASTER_URI=http://192.168.1.9:11311 *#IP Rpi*

#ΣΤΟ RPI

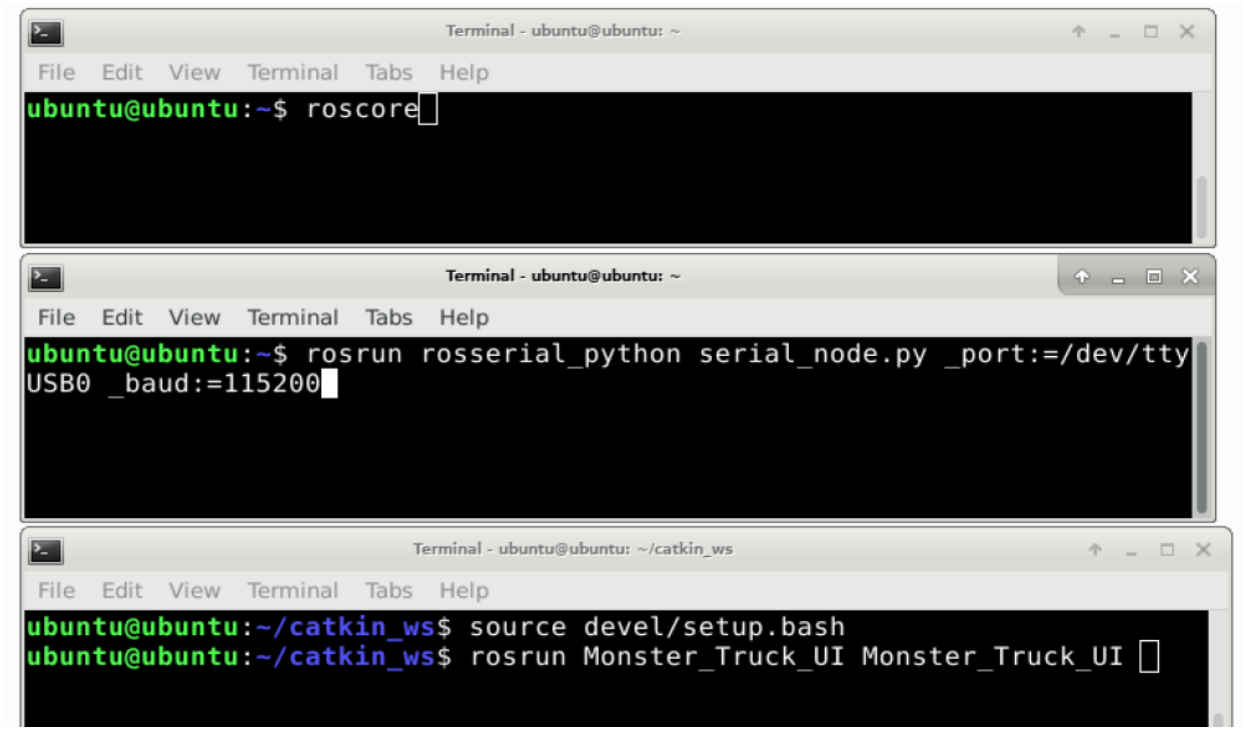
#ROS-RPI CONFIGURATION

export ROS_MASTER_URI=http://localhost:11311 *#SET DEFAULT HOST*

export ROS_HOSTNAME=192.168.1.9 *#IP Rpi*

export ROS_IP=192.168.1.9

Επόμενο βήμα είναι η εντολή roscore, η οποία εκκινεί και θέτει σε ετοιμότητα όλες τις βασικές παραμέτρους για την λειτουργία του συστήματος. Ακολουθεί η εντολή rosserial (rosrun rosserial_python serial_node.py port:=/dev/ttyACM0 baud:=115200), ώστε να νοείται το Arduino ως node στο ROS, και τίθεται σε αναμονή η επικοινωνία έως ότου ληφθεί εντολή κίνησης. Πρακτικά αναλαμβάνει η συνάρτηση loop, περιμένοντας την εκπλήρωση των όποιων απαραίτητων προϋποθέσεων ανά περίπτωση. Για την εκκίνηση του προγράμματος, εισάγεται η εντολή: rosrun Monster_Truck_UI Monster_Truck_UI στον τερματικό. Η διαδικασία ξεκινά με την σειρά που περιγράφεται και παραπάνω (Εικόνα 35). Εμφανίζεται το γραφικό περιβάλλον και ανάλογα με τις εντολές του χρήστη, ακολουθεί και η αντίστοιχη αξιοποίηση του κατάλληλου κώδικα.



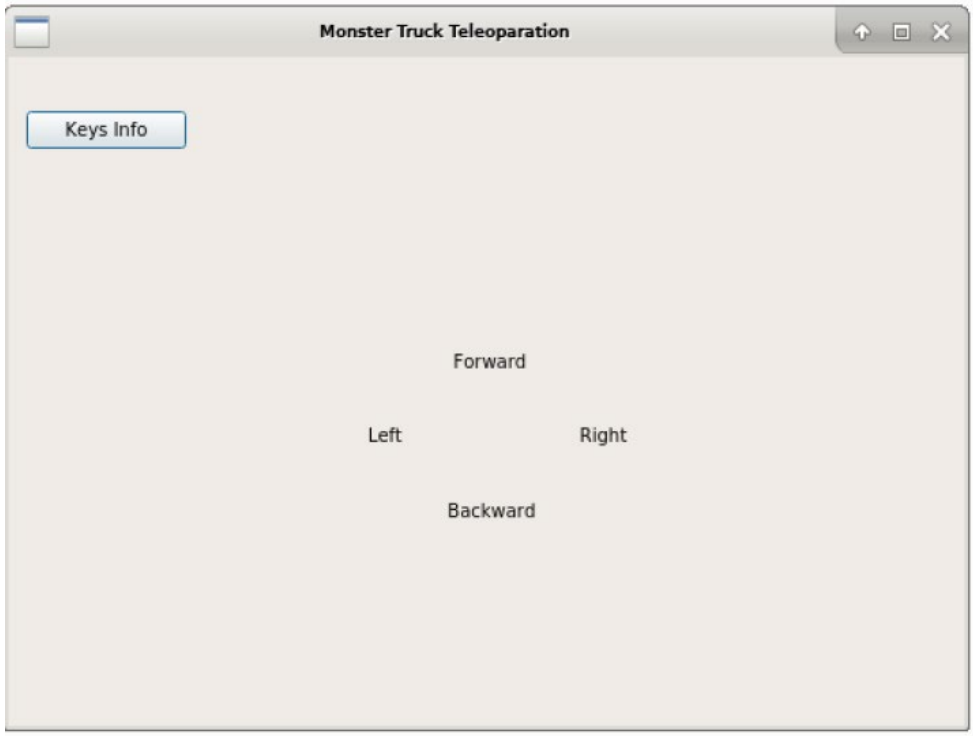
Εικόνα 35. Ακολουθία εντολών εκτέλεσης προγράμματος

4.5 Επεξήγηση ενός σεναρίου κίνησης

Έχοντας ακολουθήσει τα παραπάνω βήματα, αναδύονται τα παράθυρα του γραφικού περιβάλλοντος και ο χρήστης επιλέγει τον τρόπο πλοήγησης. Για την χειροκίνητη οδήγηση, τα κύρια κομμάτια κώδικα που συνδυάζονται είναι το αρχείο manual_window.cpp και navigation.ino. Ειδικά, το πρώτο αφορά το κομμάτι του γραφικού περιβάλλοντος. Οι εντολές που δίνονται εκεί αφορούν την δημοσίευση δεδομένων στο ROS με την μορφή cmd_vel_msg. Πατώντας, λοιπόν, ένα συγκεκριμένο κουμπί, το οποίο έχει κωδικοποιηθεί ως ένα (1), σημαίνεται η κίνηση προς τα εμπρός. Αντίστοιχα για να οπισθοδρομήσει, ο κωδικός για το κουμπί είναι -1. Σε αυτό το σημείο αξίζει να αποσαφηνιστεί πως δεν μπορούσε να δίνεται η ταχύτητα κατευθείαν από εκείνο το σημείο του κώδικα λόγω καθυστερήσεων, με αποτέλεσμα σε περίπτωση αλλαγής της να χάνεται χρόνος. Για αυτό επιλέχθηκε η μέθοδος που περιγράφηκε παραπάνω, με το Arduino να λαμβάνει την ταχύτητα σε PWM. Από την άλλη, στο αρχείο navigation.ino, στο κομμάτι που αφορά την χειροκίνητη οδήγηση, αξιοποιείται η δομή επιλογής if…else. Αν η ταχύτητα είναι μεγαλύτερη του μηδενός, το όχημα θα κινηθεί προς τα μπροστά, διαφορετικά προς τα πίσω. Αν δεν πατιέται κάποιο από τα κουμπιά, ακινητοποιείται. Όμοια για το servo, γίνονται οι απαραίτητες κωδικοποιήσεις για την στροφή δεξιά και αριστερά και για το LED πότε να ανάβει και να σβήνει. Η διαδικασία περιγράφεται αναλυτικά και με συνοδευόμενες εικόνες στην παρακάτω επέκταση.

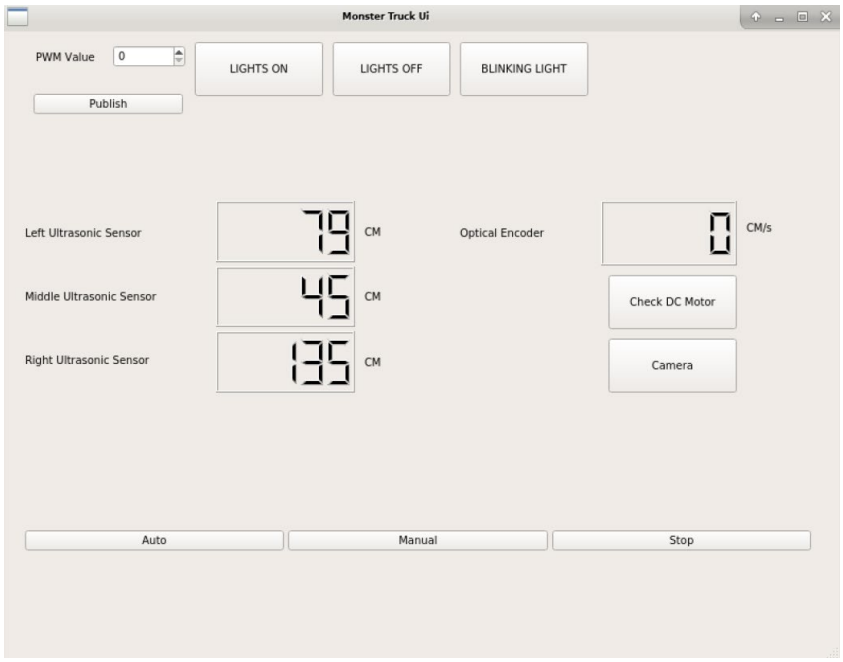
ΚΕΦΑΛΑΙΟ 5: Επεκτάσεις - Συμπεράσματα

Ως επέκταση, προστέθηκε κάμερα με την οποία μπορεί ο χρήστης να παρακολουθεί ανά πάσα στιγμή την πορεία του ρομπότ. Σε ένα ενδεικτικό βίντεο που παραδίδεται, παρουσιάζεται η χειροκίνητη πλοήγηση του οχήματος μέσα σε έναν εσωτερικό χώρο με εμπόδια. Επίσης, διακρίνεται και η χρήση του LED. Αρχικά, ενεργοποιούνται τα παράθυρα διάδρασης, σύμφωνα με την εικόνα 34, και στην συνέχεια με την επιλογή manual, αναδύεται το παράθυρο με τα πλήκτρα ελέγχου κατεύθυνσης (forward-backward, left-right). Πατώντας οποιοδήποτε από τα κουμπιά, αυτά φωτίζονται για να υποδείξουν την ενεργοποίησή τους. Βέβαια, αυτή η επέκταση δεν μπορεί να λειτουργήσει μέσω του συστήματος linux, γιατί με τα μέχρι στιγμής δεδομένα, η μεταφορά δεδομενων εκτελείται μόνο τοπικά μέσω του περιβάλλοντος qt. Για να είναι αυτό δυνατό, πρέπει να δημιουργηθεί ξεχωριστό ros topic για την κάμερα, προκειμένου να μεταδίδονται τα δεδομένα [48]. Επιπλέον, για την αξιοποίηση της κάμερας χρησιμοποιήθηκε η βιβλιοθήκη Qcamera. Η διασύνδεση πραγματοποιήθηκε με API του γραφικού περιβάλλοντος (qt) που διαχειρίζεται κάμερες. Το αρχείο κώδικα ονομάζεται cameradialog.cpp και πρόκειται για κλάσση που είναι υπεύθυνη για την λειτουργία του παραθύρου της κάμερας και να λαμβάνει τα δεδομένα από αυτήν.

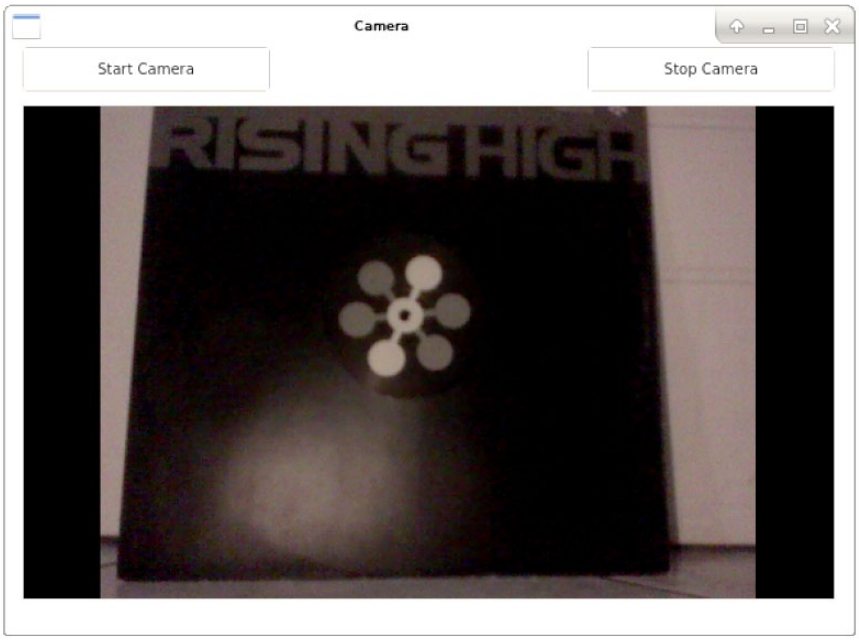


Εικόνα 36. Παράθυρο χειροκίνητης πλοήγησης

Παρακάτω, βρίσκεται ένα στιγμιότυπο κατά την κίνηση του ρομπότ, μέσα από το γραφικό περιβάλλον. Εφόσον δεν δίνεται κάποια τιμή στην ταχύτητα, το ρομπότ είναι ακίνητο και στο πλαίσιο των αποστασιόμετρων εμφανίζονται οι αποστάσεις σε εκατοστά από οτιδήποτε εντοπίζεται στο χώρο την δεδομένη στιγμή.



Εικόνα 37. Παράθυρο διαχείρισης κίνησης σε λειτουργία



Εικόνα 38. Φωτογραφία από την κάμερα του ρομπότ

Η παραλλαγή αυτή είναι δυνατό να προσφέρει εναλλακτικές χρήσεις του ρομπότ, πέραν της πλοήγησης και έρευνας σε άγνωστα μέρη. Παράλληλα, προκύπτουν δυνατότητες αναγνώρισης εικόνων (image recognition) και δράσης ανάλογα με τα ερεθίσματα που αναγνωρίζονται.

Σχετικά με τα επιδιωκόμενα αποτελέσματα της εργασίας, μέσα από τις δοκιμές που πραγματοποιήθηκαν, διαπιστώθηκε πως η διαχείριση του οχήματος με την επιλογή auto, αυτή της αυτόνομης οδήγησης, δεν λειτούργησε ιδανικά μέσα από την χρήση του γραφικού περιβάλλοντος. Παρατηρήθηκαν μεγάλες καθυστερήσεις πραγματικού χρόνου, επομένως εκτελούνταν εντολές μόνο μέσω του εγκιβωτισμένου κώδικα για το Arduino, όπως για παράδειγμα η ενεργοποίηση των κινητήρων. Επιπρόσθετα, δεν κατάφεραν να αξιοποιηθούν τα αποστασιόμετρα, ακριβώς λόγω της καθυστέρησης μεταφοράς δεδομένων. Από την άλλη, με την χειροκίνητη εναλλακτική τα αποτελέσματα ήταν ενθαρρυντικά, χωρίς όμως να απουσιάζουν οι καθυστερήσεις. Ο χαρακτήρας της εργασίας ήταν κυρίως διερευνητικός και εκπαιδευτικός, αφού αποτέλεσε μοναδική εμπειρία εις βάθους εξοικείωσης προγραμματισμού ενός τροχήλατου ρομπότ μέσω ROS.

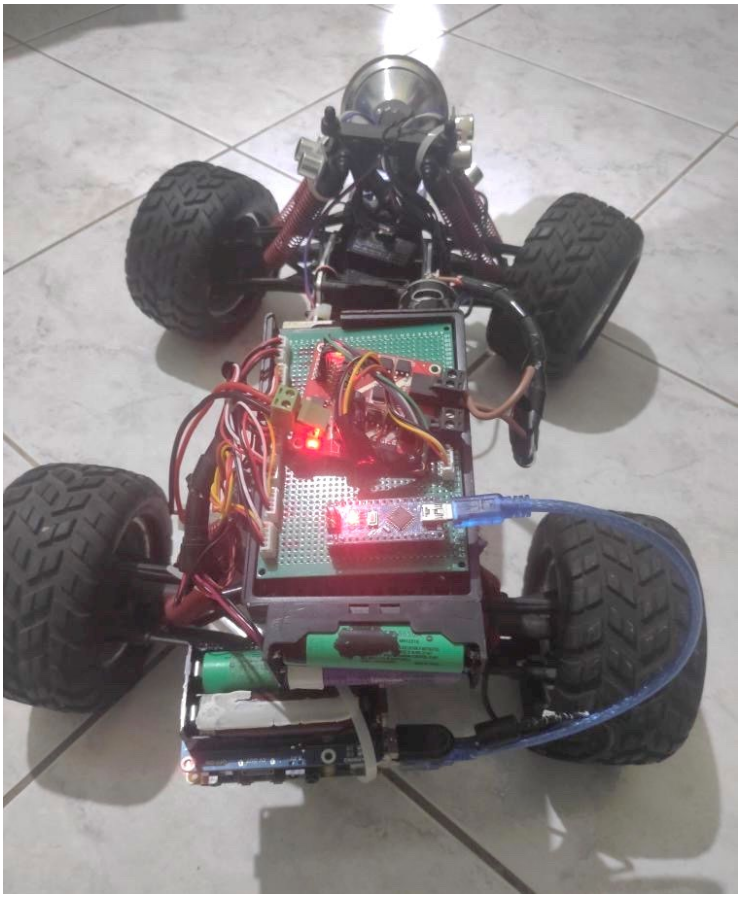
Αναζητώντας αντίστοιχες εφαρμογές στην εκπαίδευση, εντοπίζεται η χρήση ενός τροχήλατου ρομπότ για την εκμάθηση νόμων φυσικής σε σχολικό περιβάλλον. Ειδικότερα, η έρευνα αναφέρεται στην κατασκευή και διαχείριση ρομπότ από τους μαθητές μέσω μιας εφαρμογής σε κινητό (application). Το βασικό στοιχείο είναι ο Arduino και το επιδιωκόμενο γνωστικό αντικείμενο ο τρίτος νόμος του Νεύτωνα. Οι μαθητές θα χειρίζονται μέσα από το περιβάλλον της εφαρμογής την κίνηση αυτού, αλλάζοντας κάθε φορά το βάρος που θα πρέπει να τραβήξει το ρομπότ και παρατηρώντας τις αντιδράσεις [49]. Σε αυτή την διερευνητική διαδικασία, η ρομποτική εμφανίζεται ως αντικείμενο μελέτης, αλλά και ως μέσο μάθησης. Στην τριτοβάθμια εκπαίδευση, φοιτητές μηχανικής του North Central College, με προτροπή του αναπληρωτή καθηγητή Hector Rico-Aniles, ανέλαβαν την δημιουργία ενός τροχήλατου ρομπότ που θα μπορεί να κινείται και να αναζητεί δεδομένα σε ξένο περιβάλλον, αποσκοπώντας στις δυνατότητες του Mars Rover. Ως στόχοι τίθενται ο σχεδιασμός διαδρομής, η αποφυγή εμποδίων, η σάρωση του περιβάλλοντος και ταυτόχρονα ο εντοπισμός θέσης. Η μορφολογία θα προσομοιάζει μια πιο εξελιγμένη δομή αυτής των τηλε-κατευθυνόμενων παιχνιδιών και υβριδικών αυτοκινήτων. Βασικά στοιχεία είναι το Raspberry pi και μια FPGA SoC πλακέτα [50].

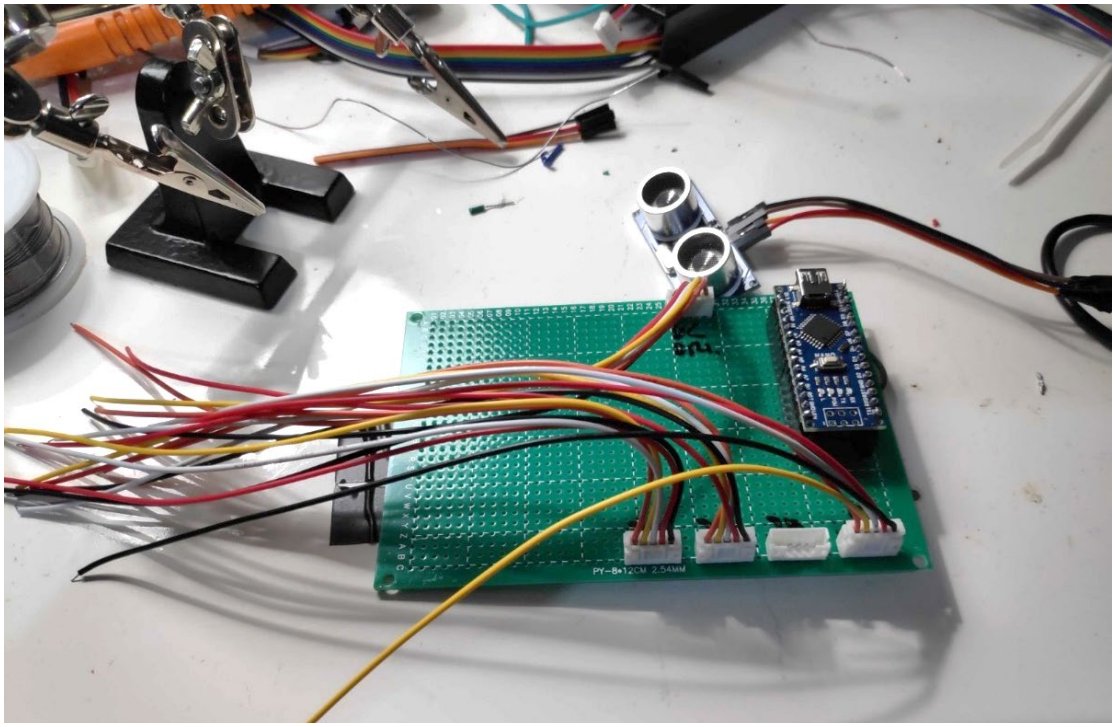
Σε κάθε περίπτωση, γίνεται φανερό πως η παρούσα μελέτη μπορεί να αποτελέσει αφετηρία για πολλές εκπαιδευτικές εφαρμογές, προσαρμόζοντας κάθε φορά τα επιμέρους εξαρτήματα και τους στόχους. Συγκεκριμένα, ο βαθμός δυσκολίας καθορίζεται από το επίπεδο εμπλοκής στη

κατασκευή καθαυτή, στην επιλογή των μέσων και αισθητήρων, την γλώσσα προγραμματισμού και το πλαίσιο που δημιουργείται από τους μαθησιακούς στόχους. Σε ένα αρχικό στάδιο, ανάλογα την εκπαιδευτική βαθμίδα για την οποία προορίζεται, θα μπορούσε να χρησιμοποιηθεί μόνο ο Arduino, ηχείο και δύο αντί για τέσσερις τροχούς. Σε ένα ακαδημαϊκό πλαίσιο, το ROS θα ήταν μια ιδανική προσθήκη, καθώς και η επέκταση των στόχων, με τον σχεδιασμό διαδρομής (path planning), τον εντοπισμό θέσης με σκοπό την αυτονομία.

Οι στόχοι που είχαν τεθεί, κατά την έναρξη της εργασίας, ικανοποιήθηκαν σε μεγάλο βαθμό. Οι βελτιώσεις που επιδέχεται η κατασκευή για την ιδανική πλοήγηση είναι προγραμματιστικής και συντονιστικής φύσεως. Ως προτεραιότητα, λοιπόν, τίθεται η περαιτέρω μελλοντική διεξαγωγή μελέτης για την επίλυση του ζητήματος της απόκρισης. Ακόμη, μια ενδιαφέρουσα προσθήκη που θα μπορούσε να βελτιστοποιήσει την κίνηση του οχήματος, θα ήταν η διεξοδική αναθεώρηση της κατασκευής ως προς την μηχανολογία, σύμφωνα με την έρευνα της Fung [23] για το Ackermann. Με αυτά τα νέα δεδομένα, θα ανακύψουν και νέες παραμετροποιήσεις στον εώς τώρα προγραμματισμό, σημαίνοντας μια εις βάθος αναζήτηση, ενδελεχή σχεδιασμό και πειραματισμό.

ΠΑΡΑΡΤΗΜΑ: ΕΠΙΠΡΟΣΘΕΤΟ ΦΩΤΟΓΡΑΦΙΚΟ ΥΛΙΚΟ & ΚΩΔΙΚΑΣ





ΚΩΔΙΚΑΣ

Ακολουθεί ο παραδοτέος κώδικας της ερευνητικής εργασίας. Διακρίνεται σε κομμάτια, ξεχωρίζοντας έτσι τις διαφορετικές λειτουργίες του κάθε script. Μετά από το χαρακτηριστικό σύμβολο // και με πράσινο χρώμα, είναι κάποια σύντομα, επεξηγηματικά σχόλια.

Παρατίθονται με την σειρά οι κώδικες για την λειτουργία των αποστασιόμετρων, των κινητήρων και σερβοκινητήρων, του οπτικού αποκωδικοποιητή, της αυτόνομης πλοήγησης και οι συναρτήσεις για το περιβάλλον αλληλεπίδρασης (A). Αντίστοιχα, παρακάτω διακρίνονται και οι κώδικες που συνθέτουν την δομή και την λειτουργία του γραφικού περιβάλλοντος (B).

A. ARDUINO LOW LEVEL CONTROL

```
//Εισαγωγή απαραίτητων βιβλιοθηκών
#include "ros.h"
#include <std_msgs/String.h>
#include <geometry_msgs/Twist.h>
#include <std_msgs/Int32.h>
#include <std_msgs/UInt16.h>
#include <sensor_msgs/Range.h>
#include <Servo.h>
#include <HardwareSerial.h>
#include <geometry_msgs/Vector3.h>

//--ROS
ros::NodeHandle* nh = new ros::NodeHandle();

//Μεταβλητές για κάθε μήνυμα
geometry_msgs::Vector3 ultrasonic_sensors;
typedef enum { STOP = 0, MANUAL = 1, AUTONAV = 2 , CHECKMOTORS = 4 }
commands;
typedef enum  { ON = 0, OFF = 1, BLINK = 3} ledCommands;
typedef enum  { RIGHT = 1, LEFT = 2 } servoCommands;
```

```
std_msgs::Int32 rpm_msg;
int led_msg = 0;
int  qt_msg = 0;
int servo_command = 0;
int counter = 0;

//LED
const int led_pin = 10;
//Ultrasonic Sensors
long duration; // Μεταβλητή που αποθηκεύει την διάρκεια του ηχητικού κύματος
int distance; // Μεταβλητή για την μέτρηση απόστασης
//--U1 ως το 1ο αποστασιόμετρο. Ακολουθούν αντίστοιχα τα άλλα δύο
#define u1_echo A2
#define u1_trig A3
//--U2
#define u2_echo A0
#define u2_trig A1
//--U3
#define u3_echo 6
#define u3_trig 7
//--Servo
Servo myservo;  // Δημιουργία αντικειμένου servo για τον έλεγχό του
int servo_pos = 0;
//--Motor Driver
const int en_a = 5;
const int in_1 = 4;
const int in_2 = 12;
int motor_speed = 0;
int sp = 0;
```

```
//--Optical Encoder
#define enc_count 200
#define enc  2
volatile long encoder_value = 0;
int interval = 1000;
long previus_millis = 0;
long current_millis = 0;
int rpm = 0;
float cm = 0;
//Speed limiter
float des_cm = 0.15;
unsigned int safety_speed = 50;
void balance_speed(float current_cm, float des_cm, int& pwm_);
//--Function prototyping
//--ROS Callbacks and functions
void callback_motors(const geometry_msgs::Twist& cmd_vel);
void pwm_callback(const geometry_msgs::Twist& pwm_msg);
void callback_servo(const std_msgs::UInt16& servo_msg);
void callback_qt(const std_msgs::UInt16& qt_msg_f);
void callback_led(const std_msgs::UInt16& led_msg_f);
void rpm_publisher();
void ultrasonic_sensors_publisher_call();
//--Ultrasonic
int u1();
int u2();
int u3();
//--Motors
void move_for(int sp);
void stop_stop();
void move_backwards(int sp);
```

```
void check_motors();
//--Servo
void turn_right();
void turn_left();
void manual_turn_right();
void manual_turn_left();
void check_servo();
//--Optical Encoder
void update_encoder();
float calculate_rpm();
//--Navigation
void auto_navigation(int& sp);
//--GUI
void auto_GUI();
void manual_GUI();
void stop_GUI();
void check_GUI();
void led_on_off();
```

A1. ROS Publishers & Subscribers

```
//--Publishers
ros::Publisher  optical_encoder_publisher("optical_encoder", &rpm_msg);
ros::Publisher  ultrasonic_sensors_publisher("ultrasonic_sensors", &ultrasonic_sensors);


//--Subscribers
//Motor Suscriber
ros::Subscriber <geometry_msgs::Twist> motor("/cmd_vel", &callback_motors); // Subscribe from topic cmd_vel, twisted messages
ros::Subscriber <geometry_msgs::Twist> pwm_sub("/pwm_value", &pwm_callback); // Subscribe from topic cmd_vel, twisted messages
```

```
//--Qt command Subscriber
ros::Subscriber <std_msgs::UInt16> qt_sub("/qt", &callback_qt);
//--Servo Subscriber
ros::Subscriber <std_msgs::UInt16> servo("/servo", &callback_servo);
//--LED Subscriber
ros::Subscriber <std_msgs::UInt16> led("/led", &callback_led);
```

A2. VOID SETUP

```
void setup() {
  //Led setup
  pinMode(led_pin, OUTPUT);
  //Ultrasonic Sensors Setup
  pinMode(u1_trig, OUTPUT); // Sets the trigPin as an OUTPUT
  pinMode(u1_echo, INPUT); // Sets the echoPin as an INPUT
  pinMode(u2_trig, OUTPUT); // Sets the trigPin as an OUTPUT
  pinMode(u2_echo, INPUT); // Sets the echoPin as an INPUT
  pinMode(u3_trig, OUTPUT); // Sets the trigPin as an OUTPUT
  pinMode(u3_echo, INPUT); // Sets the echoPin as an INPUT
  //Servo Setup
  myservo.attach(A4);
  myservo.write(90);
  //Motor Drivers Setup
  pinMode(en_a, OUTPUT);
  pinMode(in_1, OUTPUT);
  pinMode(in_2, OUTPUT);
  //Optical Encoder
  pinMode(enc, INPUT_PULLUP);
  attachInterrupt(digitalPinToInterrupt(enc), update_encoder, RISING);
  //Motors Check
  //check_servo();
```

```
  //check_motors();
//turn_right();
  //Communication rate
  nh->getHardware()->setBaud(115200);
  Serial.begin(115200);
  //Nodes initialization
  nh->initNode();
  //ROS--Publishers-Subscribers
  nh->subscribe(motor);
  nh->subscribe(pwm_sub);
  nh->subscribe(qt_sub);
  nh->subscribe(servo);
  nh->subscribe(led);
  nh->advertise(optical_encoder_publisher);
  nh->advertise(ultrasonic_sensors_publisher);   }
```

A3. VOID LOOP

```
void loop()
{
  auto_GUI();
  manual_GUI();
  stop_GUI();
  check_GUI();
  led_on_off();
  nh->spinOnce();
  delay(2);
}
```

A4. CALLBACKS & FUNCTIONS

```
//--Callback function for motors
void callback_led(const std_msgs::UInt16& led_msg_f)
{
  led_msg = led_msg_f.data;
}
void callback_motors(const geometry_msgs::Twist& cmd_vel)
{
  sp = cmd_vel.linear.x;
}
void pwm_callback(const geometry_msgs::Twist& pwm_msg)
{
  motor_speed = pwm_msg.linear.x;
}
void callback_qt(const std_msgs::UInt16& qt_msg_f)
{
  qt_msg = qt_msg_f.data;
}
void callback_servo  (const std_msgs::UInt16& servo_msg)
{
  servo_command = servo_msg.data;
}
//--Publisher for rpm
void rpm_publisher()
{
  rpm_msg.data = int (calculate_rpm());
  optical_encoder_publisher.publish(&rpm_msg);
}
```

```
//--Publisher for U1
void ultrasonic_sensors_publisher_call()
{
  ultrasonic_sensors.x = u1();
  ultrasonic_sensors.y = u2();
  ultrasonic_sensors.z = u3();
  ultrasonic_sensors_publisher.publish(&ultrasonic_sensors);
}
```

A5. ULTRASONIC

```
int u1(){        //Middle Sensor
   // Ξεκινάμε με εντολή απενεργοποίησης πριν δοθεί οποιοδήποτε σήμα
  digitalWrite(u1_trig, LOW);
  delayMicroseconds(2);
  // Δίνουμε trigPin HIGH (ACTIVE) για 10 microseconds
  digitalWrite(u1_trig, HIGH);
  delayMicroseconds(10);
  digitalWrite(u1_trig, LOW);
  // Λαμβάνει το echoPin, επιστρέφει τον ήχο κύματος σε microseconds
  duration = pulseIn(u1_echo, HIGH);
  // Για τον υπολογισμό της απόστασης
  distance = duration * 0.034 / 2; // Speed of sound wave divided by 2 (go and back)
//Η απόσταση εμφανίζεται στην οθόνη
//Serial.print("U1: Distance: ");Serial.print(distance);Serial.println(" cm/n");
  ultrasonic_sensors.x = distance;
  ultrasonic_sensors_publisher.publish(&ultrasonic_sensors);
  return distance;
  }
```

```
int u2(){       //Right Sensor
  // Ξεκινάμε με εντολή απενεργοποίησης πριν δοθεί οποιοδήποτε σήμα
  digitalWrite(u2_trig, LOW);
  delayMicroseconds(2);
  // Δίνουμε trigPin HIGH (ACTIVE) για 10 microseconds
  digitalWrite(u2_trig, HIGH);
  delayMicroseconds(10);
  digitalWrite(u2_trig, LOW);
  // Λαμβάνει το echoPin, επιστρέφει τον ήχο κύματος σε microseconds
  duration = pulseIn(u2_echo, HIGH);
  // Για τον υπολογισμό της απόστασης
  distance = duration * 0.034 / 2; // Speed of sound wave divided by 2 (go and back)
//  Η απόσταση εμφανίζεται στην οθόνη
// Serial.print("U2: Distance: ");Serial.print(distance);Serial.println(" cm");
  ultrasonic_sensors.y = distance;
  ultrasonic_sensors_publisher.publish(&ultrasonic_sensors);
  return distance;   }
int u3(){         //Left Sensor
  // Ξεκινάμε με εντολή απενεργοποίησης πριν δοθεί οποιοδήποτε σήμα
  digitalWrite(u3_trig, LOW);
  delayMicroseconds(2);
  // Δίνουμε trigPin HIGH (ACTIVE) για 10 microseconds
  digitalWrite(u3_trig, HIGH);
  delayMicroseconds(10);
  digitalWrite(u3_trig, LOW);
  // Λαμβάνει το echoPin, επιστρέφει τον ήχο κύματος σε microseconds
  duration = pulseIn(u3_echo, HIGH);
  // Για τον υπολογισμό της απόστασης
  distance = duration * 0.034 / 2; // Speed of sound wave divided by 2 (go and back)
// Η απόσταση εμφανίζεται στην οθόνη
```

```
//Serial.print("U3 Distance: ");Serial.print(distance);Serial.println(" cm/n");
  ultrasonic_sensors.z = distance;
  ultrasonic_sensors_publisher.publish(&ultrasonic_sensors);
  return distance;
  }
```

A6. MOTORS

```
void move_for(int sp_){

    digitalWrite(in_1, LOW);
    digitalWrite(in_2, HIGH);
    analogWrite(en_a, sp_);
}
void stop_stop(){

    digitalWrite(in_1, LOW);
    digitalWrite(in_2, LOW);
    analogWrite(en_a, sp);     }
void move_backwards(int sp){

   digitalWrite(in_1, HIGH);
   digitalWrite(in_2, LOW);
   analogWrite(en_a, abs(sp));
}
void check_motors(){


   digitalWrite(in_1, LOW);
   digitalWrite(in_2, HIGH);
   for(int i{0}; i < 20; i ++){
     analogWrite(en_a, i);
     delay(40);
   }
```

```
  delay(2000);
  digitalWrite(in_1, HIGH);
  digitalWrite(in_2, LOW);
  for(int i{0}; i < 25; i ++){
    analogWrite(en_a, i);
    delay(40);
  }
  delay(2000);
  digitalWrite(in_1, HIGH);
  digitalWrite(in_2, LOW);
  analogWrite(en_a, 0);
}
```

A7. SERVO

```
void move_for(int sp_){

  digitalWrite(in_1, LOW);
  digitalWrite(in_2, HIGH);
  analogWrite(en_a, sp_);
}
void stop_stop(){

  digitalWrite(in_1, LOW);
  digitalWrite(in_2, LOW);
  analogWrite(en_a, sp);
}
void move_backwards(int sp){

  digitalWrite(in_1, HIGH);
  digitalWrite(in_2, LOW);
```

```
  analogWrite(en_a, abs(sp));
}
void check_motors(){

  digitalWrite(in_1, LOW);
  digitalWrite(in_2, HIGH);
  for(int i{0}; i < 20; i ++){
    analogWrite(en_a, i);
    delay(40);
  }

   delay(2000);
   digitalWrite(in_1, HIGH);
   digitalWrite(in_2, LOW);
   for(int i{0}; i < 25; i ++){
    analogWrite(en_a, i);
    delay(40);
   }
   delay(2000);
   digitalWrite(in_1, HIGH);
   digitalWrite(in_2, LOW);
   analogWrite(en_a, 0);
}
```

A8. ENCODER

```
void update_encoder(){encoder_value ++;}

float calculate_rpm(){
 current_millis = millis();
 if (current_millis - previus_millis > interval){
```

```
    previus_millis = current_millis;
    rpm = (float)(encoder_value *60 / enc_count);
    cm = (0.11*2*3.14 *(rpm)/60);
    encoder_value = 0;
  }
  rpm_msg.data = int(cm*10);
  optical_encoder_publisher.publish(&rpm_msg);
  return cm;
}

void balance_speed(float current_cm, float des_c, int& pwm_){

   if (sp > 0){
     while (current_cm < des_c){
      sp ++;
      delay(1);
      if(sp >= safety_speed){sp -= 5;}
      break;
     }
   }
   while (current_cm >= des_c){
        sp --;
        delay(1);
        if(sp >= safety_speed){sp -= 5;}
        break;
    }
}
```

A9. VOID NAVIGATION

```
void auto_navigation(int& sp){

 int obstacle_distance = 25;
 // Αν δεν υπάρχει εμπόδιο:
 if(u1() > obstacle_distance && u2() > obstacle_distance && u3() > obstacle_distance)
 {
  move_for(sp);
  u1();u2();u3();
  calculate_rpm();
  // Αν το ρομπότ κολλήσει αλλά τα αποστασιόμετρα δεν βλέπουν εμπόδια:
  if(cm < 0.10){
   counter ++;
  }

  if (counter > 6){
     move_backwards(sp);
     u1();u2();u3();
     calculate_rpm();
     turn_right();
     delay(1000);

     u1();u2();u3();
     calculate_rpm();
     move_backwards(sp);
     turn_left();
     delay(1000);
     u1();u2();u3();
     calculate_rpm();
     counter = 0;
```

```
   }
 }
  else if(u3() < obstacle_distance/2){

     move_for(sp);
     turn_right();
     u1();u2();u3();
     calculate_rpm();
}
else if(u2() < obstacle_distance/2){
     move_for(sp);
     turn_left();
     u1();u2();u3();
  }
 else if(u1() < obstacle_distance){
   stop_stop();
   u1();u2();u3();
   calculate_rpm();
   delay(500);
   move_backwards(sp);
   u1();u2();u3();
   delay(800);
   stop_stop();
   u1();u2();u3();
   calculate_rpm();
   delay(200);
   if(u3() > u2()){
       move_for(sp);
       turn_right();
       u1();u2();u3();
```

```
      calculate_rpm();
    }
   else if(u2() > u3()){
     move_for(sp);
     turn_left();
     u1();u2();u3();
     calculate_rpm();
   }
  }

}
```

A10. GUI

```
void auto_GUI()
{
   if(qt_msg == AUTONAV){
   if (sp > safety_speed) {sp = safety_speed;}
   balance_speed(cm, des_cm, sp);
   auto_navigation(motor_speed);
  }
}

void manual_GUI(){
  if(qt_msg == MANUAL)
  {
     if (sp > 0)
     {
      move_for(motor_speed);
     }
     else if(sp<0)
```

```
      {
       move_backwards(motor_speed);
      }
      else
      {
       stop_stop();
      }
      //servo command is a command sent from Qt
      if (servo_command == RIGHT)
      {
        manual_turn_right();
      }
      else if (servo_command == LEFT)
      {
         manual_turn_left();
      }
      else
      {
       myservo.write(90);
      }
  }
}
void stop_GUI(){
  if(qt_msg == STOP)
  {

    stop_stop();
    u1();u2();u3();
  }
}
```

```
void check_GUI(){
  if(qt_msg == CHECKMOTORS)
  {
  move_for(sp);
  u1();u2();u3();
  calculate_rpm();
  }
}

void led_on_off()
{
    if(led_msg == ON)
    {
     digitalWrite(led_pin, HIGH);
    }
    else if(led_msg == OFF)
    {
     digitalWrite(led_pin, LOW);
    }
    else if(led_msg == BLINK)
    {

     digitalWrite(led_pin, HIGH);
     delay(50);
     digitalWrite(led_pin, LOW);
     delay(50);
    }
}
```

B. MONSTER_TRUCK_UI

B1. CAMERA WINDOW

//Cameradialog.cpp

```
#include "Cameradialog.h"
#include "ui_Cameradialog.h"
#include <QCamera>
#include <QtMultimediaWidgets/QCameraViewfinder>
#include <QCameraImageCapture>
#include <QVBoxLayout>
#include <QMenu>
#include <QAction>
#include <QCameraInfo>
#include <QScrollArea>

CameraDialog::CameraDialog(QWidget *parent) :
    QWidget(parent),
    ui(new Ui::CameraDialog)
{
    ui->setupUi(this);
    // Camera init
    m_Camera = new QCamera(this);
    m_CameraViewFinder = new QCameraViewfinder(this);
    m_CameraImapgeCapture = new QCameraImageCapture(m_Camera, this);
    m_Layout = new QVBoxLayout;
// Camera setup
    m_Camera->setViewfinder(m_CameraViewFinder);
    m_Layout->addWidget(m_CameraViewFinder);
    m_Layout->setMargin(0);
    ui->frame->setLayout(m_Layout);
```

```
CameraDialog::~CameraDialog()
{
   delete ui;
   delete m_Camera; delete m_CameraViewFinder;
   delete m_CameraImapgeCapture; delete m_Layout;
}
void CameraDialog::on_m_CameraStartButon_clicked()
{
   m_Camera->start();
}
void CameraDialog::on_m_CameraStopButton_clicked()
{
   m_Camera->stop();
}
```

//Cameradialog.h

```
#ifndef CAMERADIALOG_H
#define CAMERADIALOG_H

#include <QWidget>

namespace Ui {
class CameraDialog;
}

class QCamera;
class QCameraViewfinder;
class QCameraImageCapture;
class QVBoxLayout;
class QMenu;
```

```
class QAction;
class QScrollArea;
class CameraDialog : public QWidget
{
    Q_OBJECT
public:
    explicit CameraDialog(QWidget *parent = nullptr);
    ~CameraDialog();
private:
    Ui::CameraDialog *ui;

    QCamera* m_Camera;
    QCameraViewfinder* m_CameraViewFinder;
    QCameraImageCapture* m_CameraImapgeCapture;
    QVBoxLayout* m_Layout;

    QMenu* m_Options;
    QAction* m_StartCamera;
    QAction* m_StopCamera;
    QScrollArea* m_ScrollArea;
};
#endif // CAMERADIALOG_H
```

B3. MANUAL WINDOW

// manual_window.cpp

```
#include "manual_window.h"
#include "ui_manual_window.h"
#include <QMessageBox>
#include <QDebug>
#include <QWidget>

Manual_Window::Manual_Window(QWidget *parent) :
    QDialog(parent),
    ui(new Ui::Manual_Window)
{
    ui->setupUi(this);
    this->setWindowTitle("Monster Truck Teleoparation ");
    key = new  Keyboard_Events();
    ros_f = new _Ros();
}

Manual_Window::~Manual_Window()
{
    delete ui;delete key;delete timer; delete ros_f;
    ros_f->qt_command_publisher(0);
}
void Manual_Window::on_info_button_clicked()
{
    QMessageBox::about(this,"Control Info ",info_message);
}
void Manual_Window::keyPressEvent(QKeyEvent *event){
    switch(event->key()) {
        case Qt::Key_8:
```

```
        ros_f->set_cmd_vel_msg(FORWARD);
        ros_f->cmd_vel_publisher();
        ui->forward_label->setStyleSheet("QLabel { background-color : white; color : black; }");
        break;
    case Qt::Key_2:
        ros_f->set_cmd_vel_msg(BACKWARDS);
        ros_f->cmd_vel_publisher();
        ui->back_label->setStyleSheet("QLabel { background-color : white; color : black; }");
        break;
    case Qt::Key_6:
        ros_f->servo_command_publisher(RIGHT);
        ui->right_label->setStyleSheet("QLabel { background-color : white; color : black; }");
        break;
    case Qt::Key_4:
        ros_f->servo_command_publisher(LEFT);
        ui->left_label->setStyleSheet("QLabel { background-color : white; color : black; }");
        break;
    case Qt::Key_A:
        ros_f->set_cmd_vel_msg(motorCommands::STOP);
        ros_f->cmd_vel_publisher();
        break;
    case Qt::Key_Z:
        ros_f->led_command_publisher(OFF);
        break;
    case Qt::Key_X:
        ros_f->led_command_publisher(ON);
        break;
    }
}
```

```
void Manual_Window::keyReleaseEvent(QKeyEvent *event){
   switch(event->key()) {
      case Qt::Key_8:
         ui->forward_label->setStyleSheet("");
         ros_f->set_cmd_vel_msg(STOP);
         ros_f->cmd_vel_publisher();
         break;
      case Qt::Key_2:
         ui->back_label->setStyleSheet("");
         ros_f->set_cmd_vel_msg(STOP);
         ros_f->cmd_vel_publisher();
         break;
      case Qt::Key_6:
         ros_f->servo_command_publisher(STOP);
         ui->right_label->setStyleSheet("");
         break;
      case Qt::Key_4:
         //left
         ros_f->servo_command_publisher(STOP);
         ui->left_label->setStyleSheet("");
         break;
      case Qt::Key_Z:
         break;
      }
}
```

```
// manual_window.h
#ifndef MANUAL_WINDOW_H
#define MANUAL_WINDOW_H
#include <QDialog>
#include <string>
#include "../Events/keyboard_events.h"
#include <QTimer>
#include <QEvent>
#include <QKeyEvent>
#include "../ROS_src/_ros.h"
#include "../ROS_src/rviz.h"

namespace Ui {
class Manual_Window;
}

class Manual_Window : public QDialog
{
    Q_OBJECT
public:
    explicit Manual_Window(QWidget *parent = nullptr);
    virtual ~Manual_Window() override;
    virtual void keyPressEvent(QKeyEvent *event) override;
    virtual void keyReleaseEvent(QKeyEvent *event) override;
    enum servoCommands {RIGHT = 1, LEFT = 2};
    enum motorCommands { BACKWARDS = -1 , FORWARD = 1, STOP = 0};
    enum ledCommands { OFF = 0 , ON = 1};
public Q_SLOTS:
    void on_info_button_clicked();
```

```
private:
    Ui::Manual_Window *ui;
    Keyboard_Events* key;
    QTimer* timer;
    QEvent* event;
    _Ros* ros_f;
    rviz* rviz_obj;
    int x = 0;

    const char* info_message = "Keyboard Teleoparation Info\n\n"

            "-Move Forward : Num 8\n\n"

            "-Move Backward: Num 2\n\n"

            "-Turn Left :  Num 4\n\n"

            "-Turn Rigth : Num 6\n\n"

            "-Stop : Z \n\n"
            "**************************************************\n"
            "For better control press num lock and use the numpad ";
};

#endif // MANUAL_WINDOW_H
```

B4. Publishers-Subscribers

//Publishers.h

```
#ifndef PUBLISHERS_H
#define PUBLISHERS_H
#include "ros/ros.h"
#include "geometry_msgs/Twist.h"
struct Publishers
{
public:
Publishers(){}
~Publishers(){}

template<typename T>
void init(ros::Publisher& publisher_name, const char* topic_name, ros::NodeHandle& n)
{
   publisher_name = n.advertise<T>(topic_name, 25);
}
template<typename T>
void publish(ros::Publisher& publisher_name, T& msg)
{
   publisher_name.publish(msg);
}
};
#endif
```

//Subscribers

```
#ifndef SUBSCIBERS_H
#define SUBSCIBERS_H
#include "ros/ros.h"
#include <string>
```

```
#include <std_msgs/UInt16.h>
struct Subscribers
{
   Subscribers() {}
   ~Subscribers(){}
template<typename T>
void subscribe(ros::Subscriber& subscriber, const char* topic_name, ros::NodeHandle& n,  T callback)
{
  subscriber = n.subscribe(topic_name, 25, callback);
}
};
#endif
```

//_ros.cpp

```
#include "_ros.h"
#include "std_msgs/UInt16.h"
_Ros::_Ros()
{
   n = new ros::NodeHandle();
   // Create Publisher/Subscribers
   cmd_vel_pub = n->advertise<geometry_msgs::Twist>("cmd_vel", 200);
   pwm_value = n->advertise<geometry_msgs::Twist>("pwm_value", 200);
   qt_command = n->advertise<std_msgs::UInt16>("qt", 200);
   servo_command = n->advertise<std_msgs::UInt16>("servo",200);
   led_command = n->advertise<std_msgs::UInt16>("led",200);
}
_Ros::~_Ros()
{
   ros::spinOnce();
```

```
    delete n;
}
void _Ros::cmd_vel_publisher()
{
    cmd_vel_pub.publish(cmd_vel_msg);
}
void _Ros::pwm_value_publisher()
{
    pwm_value.publish(pwm_value_msg);
}
void _Ros::qt_command_publisher(unsigned short f_command)
{
    qt_msg.data = f_command;
    qt_command.publish(qt_msg);
}

void _Ros::servo_command_publisher(unsigned short f_servo_command)
{
    servo_msg.data = f_servo_command;
    servo_command.publish(servo_msg);
}
void _Ros::led_command_publisher(unsigned short f_command)
{
    led_msg.data = f_command;
    led_command.publish(led_msg);
}
```

```
//_ros.h
#ifndef _ROS_H
#define _ROS_H
#include "ros/ros.h"
#include "std_msgs/String.h"
#include "std_msgs/UInt16.h"
#include "geometry_msgs/Twist.h"
#include "geometry_msgs/Vector3.h"
#include "tf2_msgs/TFMessage.h"
#include "QDebug"
#include "geometry_msgs/Transform.h"
#include "geometry_msgs/TransformStamped.h"
#include "sensor_msgs/JointState.h"
#include "std_msgs/Header.h"
#include "std_msgs/Int32.h"

static geometry_msgs::Vector3 ultrasonic_msg;
static std_msgs::Int32 optical_encoder_msg;

class _Ros
{
public:
    _Ros();
    ~_Ros();

    //cmd_vel topic
    //Publish to cmd_vel pwm values
    void cmd_vel_publisher();
    //Set specific pwm value in linear.x
    void set_cmd_vel_msg(int pwm_f) {cmd_vel_msg.linear.x = pwm_f;}
```

```
//pwm_value topic
void pwm_value_publisher();
//Set specific pwm value in linear.x
void set_pwm_value(int pwm_f) {pwm_value_msg.linear.x = pwm_f;}
//qt_command topic
//Publish qt commands
void qt_command_publisher(unsigned short  f_command);
//Servo topic
//Publish servo commands
void servo_command_publisher(unsigned short f_servo_command);
//Set specific servo value
void set_servo_command(int servo_command_f);
//Led topic
//Publish LED commands
void led_command_publisher(unsigned short  f_command);
//ultrasonic_sensors topic
static void ultrasonic_callback(const geometry_msgs::Vector3::ConstPtr& us_msg)
{
   ultrasonic_msg = *us_msg;
}
void ultrasonic_subscriber()
{
   ros::Rate loop_rate(5);
   ultrasonic_sub = n->subscribe("ultrasonic_sensors",100,ultrasonic_callback);
   loop_rate.sleep();
   ros::spinOnce();
}
```

```
    //Optical encoder topic
    static void oe_callback(const std_msgs::Int32 ::ConstPtr& us_msg)
    {
       optical_encoder_msg = *us_msg;
    }
    void oe_subscriber()
    {
       ros::Rate loop_rate(5);
       optical_encoder_sub = n->subscribe("optical_encoder",100,oe_callback);
       loop_rate.sleep();
       ros::spinOnce();
    }
private:
    ros::NodeHandle *n;
    //cmd_vel topic
    ros::Publisher cmd_vel_pub ;
    geometry_msgs::Twist cmd_vel_msg;
    //Pwm_value topic
    ros::Publisher pwm_value ;
    geometry_msgs::Twist pwm_value_msg;
    //Qt topic
    ros::Publisher qt_command ;
    std_msgs::UInt16 qt_msg;
    //Servo topic
    ros::Publisher servo_command ;
    std_msgs::UInt16 servo_msg;
    //ultrasonic_sensors topic
    ros::Subscriber ultrasonic_sub;
    //Optical encoder topic
    ros::Subscriber optical_encoder_sub;
```

```
    //Led topic
    ros::Publisher led_command ;
    std_msgs::UInt16 led_msg;

};
#endif // _ROS_H
```

//Main.cpp

```
#include <QtGui>
#include <QApplication>
#include "ros/ros.h"
#include "mainwindow.h"
#include "ROS_src/_ros.h"

int main(int argc, char **argv)
{
    QApplication a(argc, argv);
    ros::init(argc, argv, "Monster_Truck_UI");
    _Ros ros_obj;
    MainWindow w;
    w.show();
    return a.exec();
}
```

//Mainwindow.cpp

```
#include "mainwindow.h"
#include "ui_mainwindow.h"
//QT
#include "QMessageBox"
#include "QDebug"
```

```
#include "QProcess"
#include "Events/keyboard_events.h"

//ROS MSGS
MainWindow::MainWindow(QWidget *parent) :

    QMainWindow(parent),
    ui (new Ui::MainWindow)
{
    ui->setupUi(this);
    this->setWindowTitle("Monster Truck Ui ");
    //Timers for ultrasonig sensors data
    timer = new QTimer(this);
    connect(timer, SIGNAL(timeout()),this,SLOT(ultrasonic_measurements()));
    connect(timer, SIGNAL(timeout()),this,SLOT(optical_encoder_measurements()));
    timer->start(500);
    //manual window
    manual_window = new Manual_Window();
    m_CameraWindow = new CameraDialog();
    //ROS
    ros_obj = new _Ros();
    ros_obj->ultrasonic_subscriber();
}
//Destructor
MainWindow::~MainWindow()
{
    ros_obj->set_pwm_value(0);
    ros_obj->pwm_value_publisher();
    ros_obj->qt_command_publisher(STOP);
    delete ui; delete ros_obj; delete m_CameraWindow;
```

```
}
//Auto navigation from arduino
void MainWindow::on_auto_button_clicked()
{
   ros_obj->qt_command_publisher(AUTONAV);
}
//Open dialog for manual teleop
void MainWindow::on_Manual_clicked()
{
   ros_obj->qt_command_publisher(MANUAL);
   // Open Dialog to control robot from keyboard
   manual_window->show();
   manual_window->exec();
   // Stop the motors
   ros_obj->qt_command_publisher(STOP);
}
//Stop Robot
void MainWindow::on_Stop_clicked()
{
   ros_obj->qt_command_publisher(STOP);
}
//Publisher for pwm values
void MainWindow::on_Publish_PWM_main_window_clicked()
{
   int pwm_value = ui->PWM_spinBox->text().toInt();
   ros_obj->set_pwm_value(pwm_value*2);
   ros_obj->pwm_value_publisher();
}
//Data from ultrasonic sensors
void MainWindow::ultrasonic_measurements()
```

```
{
    ui->left_ultrasonic_lcd->display(ultrasonic_msg.y );
    ui->middle_ultrasonic_lcd->display(ultrasonic_msg.x );
    ui->right_ultrasonic_lcd->display(ultrasonic_msg.z);
    ros_obj->ultrasonic_subscriber();
}
//Data from optical encoder
void MainWindow::optical_encoder_measurements()
{
    ui->optical_encoder_lcd->display(optical_encoder_msg.data);
    ros_obj->oe_subscriber();
}
//Check if motors work fine
void MainWindow::on_check_dc_motor_clicked()
{
    ros_obj->qt_command_publisher(CHECKMOTORS);
}
//Turn on LED
void MainWindow::on_lights_on_button_clicked()
{
    ros_obj->led_command_publisher(ON);
}
//Turn off LED
void MainWindow::on_lights_off_button_clicked()
{
    ros_obj->led_command_publisher(OFF);
}
//Blink LED
void MainWindow::on_lights_blinking_button_clicked()
{
```

```
    ros_obj->led_command_publisher(BLINK);
}
void MainWindow::keyPressEvent(QKeyEvent *event)
{
    switch(event->key())
    {
        case Qt::Key_Z:
        ros_obj->led_command_publisher(ON);
        break;
        case Qt::Key_X:
        ros_obj->led_command_publisher(OFF);
        break;
    }
}
void MainWindow::on_m_CameraDlgBtn_clicked()
{
    m_CameraWindow->show();
}
```

//Mainwindow.h

```
#ifndef MAINWINDOW_H
#define MAINWINDOW_H
#include <QMainWindow>
#include <QTimer>
#include "ros/ros.h"
#include "../src/ROS_src/_ros.h"
#include "Events/keyboard_events.h"
#include "Manual_Window/manual_window.h"
#include "Camera_Window/Cameradialog.h"
```

```
namespace Ui {
class MainWindow;
}
class MainWindow : public QMainWindow
{
 Q_OBJECT
public:
    explicit MainWindow(QWidget *parent = nullptr);
    virtual ~MainWindow() override;
    virtual void keyPressEvent(QKeyEvent *event) override;
    enum qtCommandToArduino {STOP = 0, MANUAL = 1, AUTONAV = 2,
CHECKMOTORS = 4};
    enum ledCommands {ON = 0, OFF = 1 , BLINK = 3};

public Q_SLOTS:
    void on_auto_button_clicked();             //Auto navigation
    void on_Manual_clicked();                  //Manual dialog exec
    void on_Stop_clicked();                   //Stop robot
    void on_Publish_PWM_main_window_clicked();
    void ultrasonic_measurements();
    void optical_encoder_measurements();
    void on_check_dc_motor_clicked();
    void on_lights_on_button_clicked();
    void on_lights_off_button_clicked();
    void on_lights_blinking_button_clicked();
    void on_m_CameraDlgBtn_clicked();
    Manual_Window* get_manual_window(){return manual_window; }

private:
    Ui::MainWindow *ui;
```

```
    _Ros* ros_obj;
    QTimer* timer;
    Keyboard_Events key;
    Manual_Window* manual_window;
    CameraDialog* m_CameraWindow;
};
#endif // MAINWINDOW_H
```

ΒΙΒΛΙΟΓΡΑΦΙΑ

[1] Καραγιαννίδης Π. (2018). Υλοποίηση Οχήματος με Αυτόνομη Κίνηση και Μετρήσεις Περιβαλλοντικών Συνθηκών μέσω Αισθητήριων Οργάνων. Σχολή Θετικών Επιστημών, Τμήμα Φυσικής. Αριστοτέλειο Πανεπιστήμιο Θεσσαλονικής.

[2] Αθανασάκη, Δ. (2015) *Κατασκευή ρομποτικού οχήματος με δυνατότητα αποφυγής εμποδίων*. Τ.Ε.Ι Μηχανικών Πληροφορικής Κρήτης.

[3] https://collection.sciencemuseumgroup.org.uk/objects/co531614/cybernetic-tortoise-developed-to-help-with-studies-on-brain-function-photocell-cybernetic-tortoise-psychology-neurology [Attribution-NonCommercial-ShareAlike 4.0 International (CC BY-NC-SA 4.0)]

[4] https://medium.com/@ReachRobotics/the-7-robots-that-shaped_the-industry-and-the-engineers-who-created-them-4f5bd42b0681

[5] https://en.wikipedia.org/wiki/Shakey_the_robot

[6] Chavez, F., B. (Μάρτιος 2018). The Main Parts of a Robot. Sciencing.com. Ανακτήθηκε από: https://sciencing.com/main-parts-robot-7403157.html

[7] Κοντόκαλος, Α. *Κατασκευή Βραχίονα 3 Βαθμών Ελευθερίας για Ρομπότ – Θερμοκηπίου*. Τμήμα Μηχανολογίας. Τ.Ε.Ι Κρήτης.

[8] Minopu Asada (2003). Robotics. Ed.: Bidgoli Hossein, Encyclopedia of Information Systems, Elsevier Science, pp. 707-722. https://doi.org/10.1016/B0-12-227240-4/00150-7.

[9] Gailliard, T. What Robots Are Used Today? Sciencing.com. Ανακτήθηκε από: https://sciencing.com/what-robots-are-used-today-12745877.html

[10] https://wowwee.com/robosapien-x

[11] Shin'ya Kotosaka, Tomohiro Shibata, Stefan Schaal. (2001). Humanoid robot "DB". Eds: Eiji Arai, Tatsuo Arai, Masaharu Takano. Human Friendly Mechatronics, Elsevier Science, pp. 279-284. https://doi.org/10.1016/B978-044450649-8/50047-4.

[12] https://en.wikipedia.org/wiki/Telenoid_R1

[13] https://en.wikipedia.org/wiki/HRP-4C

[14] Sugiura, E. (Σεπτέμβριος 2021). Japan's "work-via-robot" café helps disabled workers shine. NIKKEI Asia. Ανακτήθηκε από: https://asia.nikkei.com/Business/Business-trends/Japan-s-work-via-robot-cafe-helps-disabled-workers-shine

[15] https://www.youtube.com/watch?v=FFCPKmLAZb4

[16] https://el.wikipedia.org/wiki/Lego_Mindstorms

[17] https://niryo.com/fr/product/niryo-one/

[18] https://www.nasa.gov/audience/forstudents/5-8/features/nasa-knows/what_is_robotics_58.html

[19] https://en.wikipedia.org/wiki/Robonaut

[20] https://en.wikibooks.org/wiki/Robotics/Types_of_Robots/Wheeled

[21] Κούρος, Γ. (2016). Ανάπτυξη Αυτόνομου Ρομποτικού Οχήματος Εδάφους με Κινηματικό Μοντέλο 4WS4WD και Υλοποίηση Συστήματος για Αυτόνομη Εξερεύνηση σε Άγνωστο Περιβάλλον. Πολυτεχνική Σχολή. Αριστοτέλειο Πανεπιστήμιο Θεσσαλονίκης.

[22] Λαδουκάκης, Ο. (2011). Εφαρμογή μεθόδων Σχεδιασμού Τροχιάς για το σχεδιασμό κίνησης ρομποτικής μονάδας. Τμήμα Μηχανικών Σχεδίασης Προϊόντων και Συστημάτων. Πανεπιστήμιο Αιγαίου.

[23] Fung, A., Parker, W. (2015). *Enhanced Ackermann Steering Platform.* Worcester Polytechnic Institute, MA.

[24] Hernadez Juan, S., Herrero Cotarelo, F. (2015). *Autonomous navigation framework for a car-like robot.* Institut de Robòtica i Informàtica Industrial (IRI).

[25] Michael Cullen et al (2014). Optimal driveline robot base. Worcester Polytechnic Institute, MA.

[26] Ηλιοπούλου, Ε. (2018). *Προσομοίωση του ρομποτικού οχήματος εξωτερικού χώρου reDevil*. Σχολή Μηχανικών Παραγωγής και Διοίκησης. Πολυτεχνείο Κρήτης.

[27] https://wiki.ros.org/

[28] Μαυρίδης, Π. (2019). *Ανάπτυξη Λογισμικού σε περιβάλλον ROS για την Κίνηση βραχιόνων ρομπότ διαστημικού εξομοιωτή και Σχεδιασμός Τροχιάς για την αυτόματη σύλληψη στόχων.* Σχολή

Θετικών Επιστημών, Τμήμα Πληροφορικής και Τηλεπικοινωνιών. Εθνικό και Καποδιστριστριακό Πανεπιστήμιο Αθηνών.

[29] Fairchild, C. & Harman, T. (2016). *ROS Robotics By Example.* Packt Publishing, UK.

[30] Mayachita,I. et al. (2013). Implementation of Entertaining Robot on ROS Framework at Procedia Technology, Volume 11, pp. 380-387. Ανακτήθηκε από: https://www.sciencedirect.com/science/article/pii/S2212017313003605

[31] Kaltenegger, E., Binder, B., Bader, M. (2017). *Controlling and Tracking an Unmanned Ground Vehicle with Ackermanndrive*. Institute of Computer Aided Automation. Vienna University of Technology, Austria.

[32] https://www.skroutz.gr/s/10727492/Nano-V3-ATmega328P-USB-cable.html

[33] https://www.skroutz.gr/s/19212716/Raspberry-Pi-4-Model-B-4GB.html

[34] https://let-elektronik.dk/shop/1440-afstand--bevaegelse/15569-ultrasonic-distance-sensor---hc-sr04/

[35] https://electropeak.com/learn/interfacing-lm393-infrared-speed-sensor-with-arduino/

[36] https://store.rc4wd.com/540-Crawler-Brushed-Motor-55T_p_823.html

[37] https://ardtech.webnode.gr/l/pos-leitoyrgei-enas-servo-kinitiras/

[38] https://www.gadgeta.gr/samsung-epanafortizomeni-mpataria-viomichanikoy-typoy-unprotected-18650-li-ion-3-7v-3500mah-13a-inr18650-35e/

[39] https://www.geekbuying.com/item/Lion-Power-BG712-7-4V-5200mAh-30C-LiPo-Battery-352297.html

[40] https://grobotronics.com/waveshare-ups-hat.html

[41] Ζερμπιώνη, Α. *Αναζήτηση και Παρακολούθηση Αυτοκινούμενων Ρομποτικών οχημάτων με μεθόδους μηχανικής όρασης*. Πανεπιστήμιο Πατρών.

[42] http://wiki.ros.org/rosserial_arduino/Tutorials/Arduino%20IDE%20Setup

[43] https://fritzing.org/

[44] https://answers.ros.org/question/337030/qt-5-for-ros-melodic/

[45] https://www.codeprj.com/blog/5eb2f11.html

[46] https://www.jetbrains.com/help/clion/quick-cmake-tutorial.html

[47] https://programming.vip/docs/ros-ros-spin-and-ros-spinonce-differences-and-use.html

[48] http://wiki.ros.org/Sensors/Cameras

[49] Matos, M. & Zannin, M. (2021). Educational Robotics: Building and Applying an App-controlled Car to Study Newton's Laws. Open Education Studies. 3. 49-55. doi: 10.1515/edu-2020-0139.

[50] Imm, J. (Ιούλιος 2021). Robotic car project keeps North Central engineering students working at the cutting edge. North Central College. Ανακτήθηκε από: https://www.northcentralcollege.edu/news/2021/07/30/robotic-car-project-keeps-north-central-engineering-students-working-cutting-edge