ΑΛΕΞΑΝΔΡΕΙΟ ΤΕΧΝΟΛΟΓΙΚΟ ΕΚΠΑΙΔΕΥΤΙΚΟ ΙΔΡΥΜΑ
ΘΕΣΣΑΛΟΝΙΚΗΣ (Α.Τ.Ε.Ι.Θ.)
ΣΧΟΛΗ ΤΕΧΝΟΛΟΓΙΚΩΝ ΕΦΑΡΜΟΓΩΝ (ΣΤΕΦ)
ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΑΥΤΟΜΑΤΙΣΜΟΥ Τ.Ε.

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Κατασκευή αυτόνομου οχήματος (AGV) ,για ευέλικτα συστήματα παραγωγής

ΒΛΑΧΟΓΙΑΝΝΗΣ ΓΕΩΡΓΙΟΣ

ΚΥΡΙΑΚΙΔΗΣ ΤΙΓΓΙΛΙΔΗΣ ΠΑΝΑΓΙΩΤΗΣ

ΕΠΙΒΛΕΠΩΝ ΚΑΘΗΓΗΤΗΣ : ΔΗΜΗΤΡΙΟΣ ΜΠΕΧΤΣΗΣ,
ΚΑΘΗΓΗΤΗΣ ΕΦΑΡΜΟΓΩΝ

ΠΡΟΛΟΓΟΣ

Από την έναρξη της βιομηχανικής επανάστασης, ο κόσμος γνώρισε τον εκβιομηχανισμό της παραγωγής σε πάρα πολλούς τομείς , και με την βοήθεια των μηχανών ,εισήλθε σε μία νέα εποχή, ραγδαίας τεχνολογικής και παραγωγικής ανάπτυξης .

Σήμερα, με την χρήση του Internet, την ανάπτυξη εφαρμογών και ρομποτικών συστημάτων , μπορούμε πλέον να μιλάμε για μία νέα εποχή στην βιομηχανία , όπου οι μηχανές έχουν εξελιχθεί τόσο , ώστε να λειτουργούν ημιαυτόνομα στον κύκλο παραγωγής με αποτέλεσμα την ελαχιστοποίηση της επίβλεψης από τον ανθρώπινο παράγοντα και αντίστοιχα, την βελτιστοποίηση στην ποιότητα των υπηρεσιών.

Την τελευταία δεκαετία , η εισαγωγή έξυπνων αυτόνομων οχημάτων (**Automated Guided Vehicles - AGV**) στις γραμμές παραγωγής, διευκολύνει ακόμη περισσότερο τις παραγωγικές διεργασίες , αφού αναλαμβάνουν δύσκολες , επικίνδυνες και μονότονες εργασίες για τον άνθρωπο, καθιστώντας τα απαραίτητα για το μέλλον της εξέλιξης του τομέα της παραγωγής .

Επίσης το χαμηλό συνολικό κόστος που απαιτούν για την λειτουργία τους , σε συνάρτηση με την ταχύτητα εκτέλεσης των διεργασιών , και κυρίως έχοντας την δυνατότητα για ασταμάτητη λειτουργία ταυτόχρονα με υψηλή ακρίβεια , τα καθιστά αξεπέραστα από τον ανθρώπινο παράγοντα , και άρα απαραίτητα .

Ευρύτατη χρήση ρομπότ γίνεται σε πάρα πολλούς παραγωγικούς τομείς όπως στην ιατρική, την αεροναυπηγική, την αεροδιαστημική και κυρίως στη βιομηχανία (βιομηχανική ρομποτική), πρόσφατα μέχρι και στην πολεμική βιομηχανία κ.ά.

Ευχαριστίες

Για τη διεκπεραίωση της πτυχιακής εργασίας ευχαριστούμε θερμά:

- Τον κύριο Δημήτριο Μπεχτσή, καθηγητή Εφαρμογών του τμήματος Μηχανικών Αυτοματισμού για τη συνεχή επιστημονική επίβλεψη, καθοδήγηση και υποστήριξή του, που ήταν κομβική για την υλοποίηση της παρούσας πτυχιακής εργασίας.

- Τους καθηγητές του τμήματος Αυτοματισμού, που μας έμαθαν να σκεπτόμαστε, καθώς επίσης και να εκφράσουμε την ευγνωμοσύνη μας για όλες τις γνώσεις που απλόχερα μας χάρισαν, σε διδακτικές ώρες και μη.

- Τις οικογένειες και τους φίλους μας για την συμπαράσταση, την υπομονή και την ανεκτικότητά τους στη μέχρι τώρα φοιτητική σταδιοδρομία μας.

ABSTRACT

An Intelligent Autonomous Vehicle (IAV) prototype for industrial facilities

One of the most evolving state-of-the-art technological solutions is the development of Intelligent Autonomous Vehicles (IAVs) that navigate on industrial facilities taking real time decisions with minimal or zero human assistance. IAVs rely on microelectronics and sensor technologies for monitoring, communicating, coordinating with the facility's equipment or with other vehicles.Our custom built IAV is a prototype for industrial facilities that can monitor industrial processes in a facility and enhance the decision making process in the shop floor.

For our prototype we used a handmade chassis with two separate geared motor-driven wheels along with encoders to control their revolutions and a single caster wheel. The main data processing unit is a Raspberry Pi board, communicating through a standard Wifi shield and a PiFace digital 2 extension board with relays.

Additionally, we included an L298N dual H-bridge to drive the motors, a camera for object identification and a LiDAR laser sensor for obstacle avoidance, area mapping and identification of regions of interest. Moreover, we added sufficient cooling with a 12mm fan, an adjustable switching regulator to power the electronics, and finally, a Lithium Polymer (Li-Po) battery to power everything.

The PiFace is a H.A.T. (Hardware Attached on Top) board, which is used with the Raspberry Pi and consequently cuts off access to the original pins. In our case, we created a custom board, and attached the PiFace on top of it, releasing the majority of the pins for further use, as will be shown afterwards.
Furthermore, we developed a custom operating system (OS) image for the Raspberry Pi based on Ubuntu Linux and the Robot Operating System (ROS). ROS is an open-source, meta-operating system that provides hardware abstraction, low-level device control, data fusion and manipulation as well as libraries for building and executing operations across multiple entities. Our vehicle can be remotely controlled with the camera node and a custom ROS interface while on the same time identify the surroundings of the facility layout using the LiDAR laser sensor.

The vehicle could autonomously navigate in an industrial facility and monitor the real time processes in order to provide decisions maker with valuable feedback from the shop floor.

ΠΕΡΙΛΗΨΗ

Ένα πρωτότυπο ευφυούς αυτόνομου οχήματος (IAV) για βιομηχανικές εγκαταστάσεις

Μία από τις πλέον εξελισσόμενες τεχνολογικές λύσεις είναι η ανάπτυξη ευφυών αυτόνομων οχημάτων (IAV) που κατευθύνονται στις βιομηχανικές εγκαταστάσεις λαμβάνοντας αποφάσεις σε πραγματικό χρόνο με ελάχιστη ή μηδενική ανθρώπινη βοήθεια. Τα IAV βασίζονται σε τεχνολογίες μικροηλεκτρονικής και αισθητήρων για την παρακολούθηση, την επικοινωνία και τον συντονισμό με τον εξοπλισμό της εγκατάστασης ή με άλλα οχήματα. Το IAV μας είναι ένα πρωτότυπο για βιομηχανικές εγκαταστάσεις που μπορεί να παρακολουθήσει τις βιομηχανικές διαδικασίες σε μια εγκατάσταση και να ενισχύσει τη διαδικασία λήψης αποφάσεων.

Για το πρωτότυπο μας χρησιμοποιήσαμε ένα χειροποίητο πλαίσιο με δύο ξεχωριστούς τροχούς, με κινητήρες μαζί με κωδικοποιητές για τον έλεγχο των στροφών τους, και μία ελεύθερη ρόδα. Η κύρια μονάδα επεξεργασίας δεδομένων είναι μια πλακέτα Raspberry Pi, η οποία επικοινωνεί μέσω μιας κάρτας ασύρματου δικτύου και μιας πλακέτας επέκτασης PiFace Digital 2 με ρελέ. Επιπροσθέτως, συμπεριλάβαμε μια πλακέτα ελέγχου τον κινητήρων με διπλή γέφυρα H-bridge, την L298N για τον έλεγχο των κινητήρων, μια κάμερα για την αναγνώριση αντικειμένων παράλληλα με έναν LiDAR laser αισθητήρα για την αποφυγή εμποδίων, την χαρτογράφηση του χώρου και τον εντοπισμό των περιοχών ενδιαφέροντος. Επιπλέον, προσθέσαμε επαρκή ψύξη με ανεμιστήρα 12mm, σταθεροποιητή τάσης για να τροφοδοτήσουμε τα ηλεκτρονικά, και τέλος μια μπαταρία λιθίου (Li-Po) για την τροφοδοσία. Το PiFace είναι ένα επί κορυφής προσαρτώμενο εξάρτημα το οποίο χρησιμοποιείται με το Raspberry Pi και κατά συνέπεια διακόπτει την πρόσβαση στα αρχικά pins. Στην περίπτωσή μας, δημιουργήσαμε μία προσαρμοσμένη πλακέτα και προσαρτήσαμε το PiFace επάνω της, αφήνοντας ελεύθερα τα pins για περαιτέρω χρήση.

Επιπλέον, αναπτύξαμε ένα προσαρμοσμένο λειτουργικό σύστημα για το Raspberry Pi με το Ubuntu Linux για βάση και το λειτουργικό σύστημα του ρομπότ (ROS). Το ROS είναι ένα μετα-λειτουργικό σύστημα ανοιχτού κώδικα που παρέχει στρώμα υλικού, έλεγχο συσκευών χαμηλού-επιπέδου, σύντηξη και χειρισμό δεδομένων, καθώς και βιβλιοθήκες για την κατασκευή και την εκτέλεση εργασιών σε πολλαπλές οντότητες. Το όχημά μας μπορεί να ελεγχθεί εξ αποστάσεως μέσω του κόμβου της κάμερας και του χειροποίητου ROS περιβάλλοντος, ενώ ταυτόχρονα χαρτογραφεί τον περιβάλλοντα χώρο των εγκαταστάσεων χρησιμοποιώντας τον laser αισθητήρα LiDAR. Το όχημα θα μπορούσε να πλοηγηθεί αυτόνομα σε μια βιομηχανική εγκατάσταση και να παρακολουθεί τις διαδικασίες σε πραγματικό χρόνο, προκειμένου να παράσχει στον υπεύθυνο πολύτιμη ανατροφοδότηση δεδομένων.

Σκοπός της παρούσας πτυχιακής είναι η υλοποίηση και ο προγραμματισμός ενός αυτόνομου ρομποτικού οχήματος , που να μπορεί να χαρτογραφεί το περιβάλλον στο οποίο βρίσκεται , και αφού ο χρήστης του σηματοδοτεί τα σημεία "κλειδιά" στον χώρο , αυτό στην συνέχεια ,να μπορεί να πλοηγηθεί αυτόνομα , χωρίς περαιτέρω διορθώσεις ή προγραμματισμό, με τελικό σκοπό να επιτελεί κάποια εργασία (μεταφορά κάποιου αντικειμένου από ένα σημείο σε κάποιο άλλο κ.α.) .

Επιμέρους στόχοι που τέθηκαν είναι το ρομπότ να μπορεί να αποφεύγει με έξυπνο τρόπο τυχόν εμπόδια που του παρουσιάζονται στο δρόμο του , καθώς και την επιλογή βελτιστοποιημένης διαδρομής μεταξύ δύο επιθυμητών σημείων . Επίσης, δυνατότητα αναγνώρισης προϊόντων και αντικειμένων μέσω QR Codes από κάμερα.

Η μορφή του ρομπότ είναι σφαιρική . Η επιλογή έγινε ώστε να είναι εύκολη η πλοήγηση του στον χώρο ,αποφεύγοντας τυχόν συγκρούσεις με άλλα αντικείμενα στο στρίψιμο .

Η αξία της πτυχιακής εργασίας βρίσκεται κυρίως στην ευελιξία που προσφέρει ,στα συστήματα παραγωγής.
Αναβαθμίζει τον τρόπο με τον οποίο εκτελούνται διεργασίες και μεταφορές σε κάποιον χώρο. Αυξάνει την ικανότητα του συστήματος (παραγωγική μονάδα , εργοστάσιο κτλ) να μπορεί να αλλάζει ευκολότερα την διάταξη παραγωγής , όταν θέλει να παράγει νέους τύπους προϊόντων ,ή και σειρά εργασιών που εκτελούνται σε ένα μέρος.

Τα πλεονεκτήματα είναι πολλά όπως :

- Μειωμένο κόστος κατασκευής,
- Χαμηλότερο κόστος ανά μονάδα παραγωγής,
- Μεγαλύτερη παραγωγικότητα της εργασίας,
- Μεγαλύτερη απόδοση και αποτελεσματικότητα,
- Αυξημένη αξιοπιστία του συστήματος,
- Καλύτερα αποθέματα εξαρτημάτων, μέσω καλύτερης διαχείρησης αποθήκης .
- Προσαρμοστικότητα στις λειτουργίες CAD / CAM,
- Μικρότεροι χρόνοι παράδοσης.

Κάποια από τα μειονεκτήματα είναι :

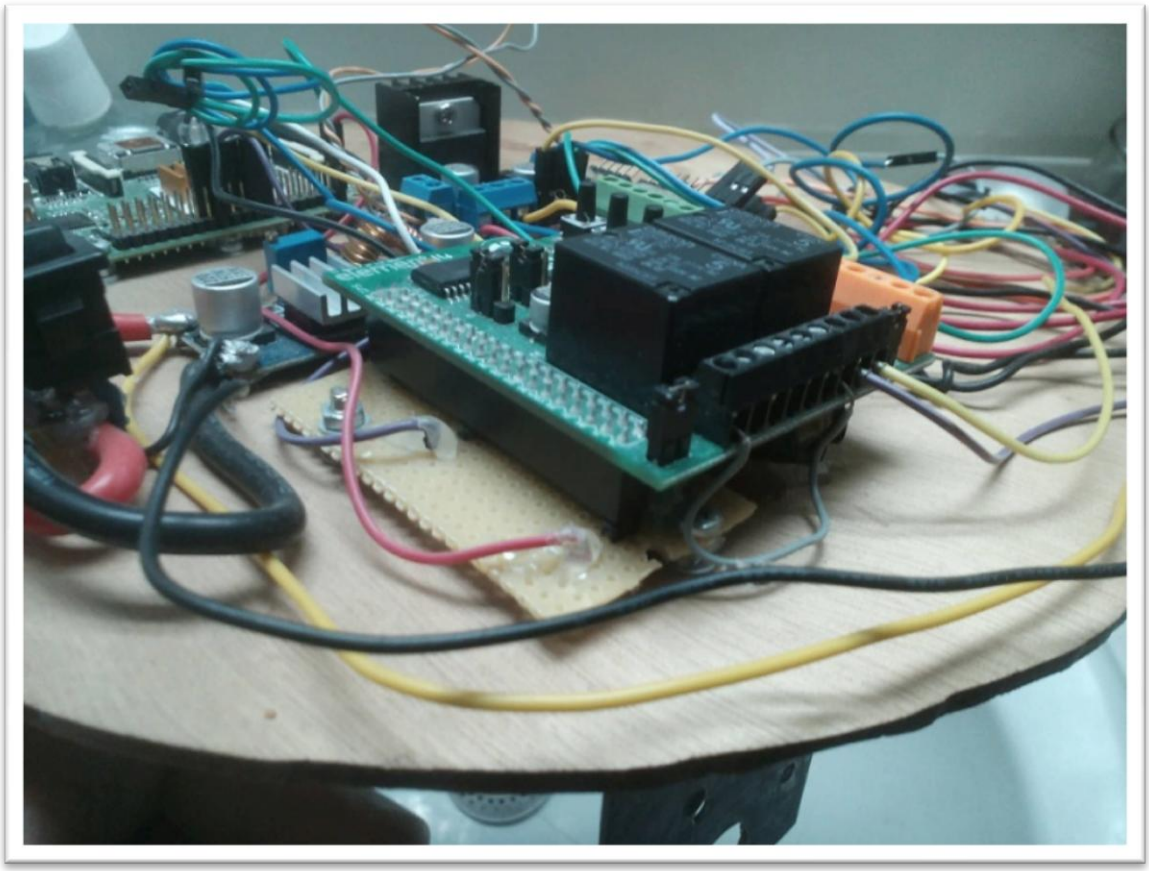
- Το αρχικό κόστος ρύθμισης
- Οι απαιτήσεις εξειδικευμένου εργατικού δυναμικού
- Η γενικότερη πολυπλοκότητα του συστήματος.

Η πρωτοτυπία της πτυχιακής βρίσκεται στο ότι υλοποιήθηκε ένα ρομποτικό όχημα με εξαρτήματα ξένα μεταξύ τους , που είχαν ως αποτέλεσμα να ανεβάσουν κατά πολύ τον πήχη της δυσκολίας , μιας και εξαρτήματα από διαφορετικούς κατασκευαστές που προορίζονται και για άλλες εφαρμογές , χρειάζονται ειδικές ρυθμίσεις , πατέντες και γενικά μελέτη για την σωστή λειτουργία και συνεργασία μεταξύ τους .

Για παράδειγμα ο τρόπος τροφοδοσίας του μικροεπεξεργαστή είναι πρωτότυπος. Ο διαχωρισμός της πλακέτας επέκτασης PiFace Digital 2 από την συνένωση με τον επεξεργαστή επίσης .

Συγκεκριμένα, η πλακέτα αυτή ,προσφέρει 8 ψηφιακές εισόδους και 8 ψηφιακές εξόδους, 2 ρελέ και 4 κουμπιά ελέγχου για την κάλυψη διαφόρων αναγκών ,παρόλα αυτά όμως , εφαρμόζει πάνω και στα 40 GPIO του Raspberry Pi με αποτέλεσμα να τα αποκλείει από τη χρήση ενώ ουσιαστικά χρησιμοποιεί μόνο 7.

Με σκοπό λοιπόν την άμεση υποστήριξη περισσότερων αισθητήρων από τις εισόδους - εξόδους του επεξεργαστή , δημιουργήθηκε μία χειροποίητη πλακέτα στην οποία το PiFace εφαρμόζει και δέχεται τις 7 εισόδους που χρειάζεται για να λειτουργεί και απελευθερώνει όλες τις υπόλοιπες προς χρήση.



Ο διαχωρισμός της πλακέτας PiFace από τον επεξεργαστή , με χειροποίητη πλακέτα.

ΠΕΡΙΕΧΟΜΕΝΑ

1. ΕΙΣΑΓΩΓΗ

1.1. Τι είναι τα ρομπότ

Ρομπότ είναι μηχανικές κατασκευές που ελέγχονται είτε απευθείας (μέσω χειρισμού) είτε λειτουργούν αυτόνομα - αυτόματα, υποκαθιστώντας τον ανθρώπινο παράγοντα σε διάφορες εργασίες σε διάφορους τομείς . Σε όλες τις περιπτώσεις , καθοδηγούνται από το εγκατεστημένο ηλεκτρονικό σύστημα τους , και φυσικά από κάποιο πρόγραμμα ηλεκτρονικού υπολογιστή .

Η λέξη ρομπότ προέρχεται από την τσέχικη λέξη robota ,που σημαίνει εργασία. Καθιερώθηκε ως όρος με την σημερινή του έννοια το 1920 από τον Τσέχο θεατρικό συγγραφέα Κάρελ Τσάπεκ.

1.2.Κατηγοριοποιήσεις

Υπάρχουν διάφορα κριτήρια διάκρισης και αντίστοιχες κατηγοριοποιήσεις των ρομπότ. Μία από αυτές είναι η διάκρισή τους σε τρεις, επί του παρόντος, γενιές.

- Στην πρώτη γενιά κατατάσσονται τα ρομπότ που διευθύνονται άμεσα από κάποιον χρήστη .Έχουν περιορισμένη ευελιξία και θα μπορούσαμε να τα θεωρήσουμε έμμεση προέκταση του χειρισμού των ανθρώπων.

  Ένα παράδειγμα είναι τα περονοφόρα ανυψωτικά, αλλιώς forklift ή Clark, τα οποία είναι ουσιαστικά εξειδικευμένα εργαλεία στην υπηρεσία του ανθρώπου, που επιτρέπουν τη μετακίνηση επικίνδυνων ή βαρέων αντικειμένων .

περονοφόρο ανυψωτικό

- Στη δεύτερη γενιά κατατάσσονται τα ρομπότ που έχουν την δυνατότητα προγραμματισμού , ώστε να λειτουργούν υπό κάποια ρουτίνα επαναλαμβανόμενη , και άρα ημιαυτόνομα .Επίσης στην ίδια κατηγορία είναι και τα ρομπότ που λαμβάνουν εντολές από κάποιο σύστημα αριθμητικού ελέγχου.
  (Ρομπότ στην βιομηχανία και στις αλυσίδες παραγωγής)

ρομποτικοί βραχίονες σε αλυσίδα παραγωγής

- Στην τρίτη γενιά κατατάσσονται τα ρομπότ που είναι επιπλέον εφοδιασμένα και με διάφορους αισθητήρες , που μέσω της επεξεργασίας των πληροφοριών που λαμβάνουν ,μπορούν και αλληλεπιδρούν με το περιβάλλον τους .Άρα είναι σύνθετα συστήματα, με δυνατότητες ανάδρασης και άμεσης αλληλεπίδρασης .

AGV σε χώρο παραγωγής , εν ώρα εργασίας

1.3. Ιστορική αναφορά

1.3.1. Αρχαίες αναφορές και κατασκευές

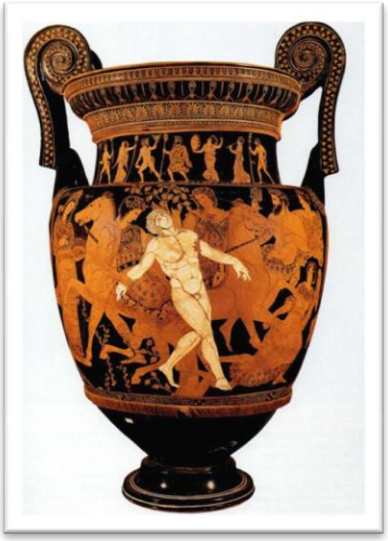
Αναφορές σε αυτόματες μηχανές και ανθρωποειδή υπάρχουν από αρχαιοτάτων χρόνων σε πολλές μυθολογίες και πολιτισμούς με εξέχων τον Ελληνικό .

Μερικά παραδείγματα είναι :

- Μηχανικές υπηρέτριες που φτιάχτηκαν από τον θεό Ήφαιστο για να τον βοηθούν στο εργαστήριο του.

- Αυτόματοι τρίποδες που σέρβιραν τους θεούς στα συμπόσια τους

- Κατασκευή του χάλκινου γίγαντα Τάλω , που με εντολή του Δία , ο Ήφαιστος φιλοτέχνησε για χάρη του Μίνωα , Βασιλιά της Κρήτης , ώστε να φυλάει το νησί από εισβολείς εν καιρώ πολέμου , και εν καιρώ ειρήνης να μεταφέρει τους βασιλικούς νόμους απ άκρη σε άκρη του νησιού .



Ο κρατήρας του Ζωγράφου του Τάλω. Εθνικό Αρχαιολογικό Μουσείο Νάπολης . Ιταλία.

- Φύλακος και Αυτόνοος . Δύο χάλκινοι γίγαντες , που υπερασπίστηκαν με επιτυχία το Ιερό των Δελφών από την επιδρομή 5000 περσών με σκοπό την λεηλάτηση του , μετά την Ηρωική πτώση των Ελλήνων στις Θερμοπύλες, όπως μας πληροφορεί ο Ηρόδοτος .

1.3.2 Τα ρομπότ στον 20ο αιώνα.

Το ρομπότ Elektro

Στα τέλη της δεκαετίας του 1920, ένα από τα πρώτα ανθρωποειδή ρομπότ παρουσιάστηκε στην έκθεση του Model Engineers Society του Λονδίνου το οποίο εφευρέθηκε από τον *WH Richards*.

Το ρομπότ Έρικ, ήταν φτιαγμένο από αλουμίνιο .Μέσα του υπήρχαν ηλεκτρομαγνήτες και ένας κινητήρας που τροφοδοτούνταν από μία μπαταρία . Το ρομπότ αν και ήταν αδύνατο να περπατήσει , μπορούσε να σηκωθεί όρθιο , να κινήσει τα χέρια του , να υποκλιθεί , ακόμα και να μιλήσει μέσω φωνητικού ελέγχου.

Το 1939, παρουσιάσθηκε στην Διεθνή Έκθεση της Νέας Υόρκης το ανθρωποειδές ρομπότ Elektro. Είχε την δυνατότητα να περπατάει με φωνητική εντολή, να μιλάει προφέροντας περίπου 700 λέξεις (χρησιμοποιώντας ένα 78-rpm πικάπ), να σκάει μπαλόνια, και να μετακινεί το κεφάλι και τους βραχίονες του. Ο εσωτερικός του μηχανισμός αποτελείτο από ένα γρανάζι χάλυβα με έκκεντρο κινητήρα.

Το 1948 ο Grey William Walter κατασκευάζει τα πρώτα αυτόνομα ρομπότ στο Νευρολογικό Ινστιτούτο Burden στο Μπρίστολ της Αγγλίας .

Στις έρευνές του θέλησε να αποδείξει ότι οι περίπλοκες συμπεριφορές του ατόμου ,ευθύνονται στον τρόπο που είναι συνδεδεμένα τα νευρονικά δίκτυα του εγκεφάλου .Οπώς πολύ σωστά υποστήριξε , η πολύπλοκη λειτουργία του εγκεφάλου οφείλεται στο πως τα εγκεφαλικά και νευρονικά δίκτυα του εγκεφάλου επικοινωνούν μεταξύ τους . Αργότερα , θα αναπτυχθεί και η γνωστή στον κλάδο επιστήμη των νευρονικών δικτύων , που θα σημάνει και την έναρξη της εποχής της τεχνητής νοημοσύνης (A.I.) .

Τα πρώτα ρομπότ που δημιούργησε ονομάζονταν *Elmer* και *Elsie*, συχνά αποκαλούμενα και "χελώνες" , λόγω του σχήματος τους και του αργού ρυθμού που κινούνταν.

Τα ρομπότ ήταν σε θέση να κάνουν φωτοταξία. Με τον τρόπο αυτό μπορούσαν να προσανατολιστούν με τον σταθμό ανεφοδιασμού , όταν η μπαταρία τους τελείωνε .
Η λέξη ρομπότ , και κατ επέκταση ρομποτική , που χρησιμοποιείται για να περιγράψει αυτές τις μηχανές, οφείλεται στον συγγραφέα επιστημονικής φαντασίας Isaac Asimov.

Ο Asimov δημιούργησε τους " Τρεις Νόμους της Ρομποτικής " γνωστοί και ως " Οι νόμοι του Asimov ", ένα σημαντικό και πρωτοπόρο θέμα θα έλεγε κανείς για την εποχή του ,που σκοπό είχε να εισάγει για πρώτη φορά έναν κώδικα συμπεριφοράς στα μελλοντικά αυτόνομα και εξελιγμένα ως προς την νοημοσύνη ρομπότ.

Έως και σήμερα χρησιμοποιούνται από πολλούς συγγραφείς , σκηνοθέτες ακόμη και εφευρέτες ,για να καθορίσουν τους νόμους που θα διέπουν τις εξελιγμένες μηχανές .
Οι τρεις νόμοι, όπως αναφέρονται στο "Εγχειρίδιο της Ρομποτικής, 56η έκδοση, 2058 AD», είναι:

1. Το ρομπότ δε θα κάνει κακό σε άνθρωπο, ούτε με την αδράνειά του θα επιτρέψει να συμβεί αυτό .
2. Το ρομπότ πρέπει να υπακούει τις διαταγές που του δίνουν οι άνθρωποι, εκτός και αν αυτές οι διαταγές έρχονται σε σύγκρουση με τον πρώτο νόμο.3. Το ρομπότ οφείλει να προστατεύει την ύπαρξή του, εφόσον αυτό δεν συγκρούεται με τον πρώτο και τον δεύτερο νόμο.

Στους παραπάνω νόμους, στηρίχθηκαν τα διηγήματα για τα ρομπότ του, όπως και πολλών άλλων συγγραφέων επιστημονικής φαντασίας μετέπειτα.

1.3.3. Τα ρομπότ στην σύγχρονη εποχή

Κινούμενα ρομπότ (mobile robots)

Τα αυτοκινούμενα ρομπότ, έχουν την ικανότητα να κινούνται αυτόνομα στο χώρο και να αλληλεπιδρούν με το περιβάλλον.

Ένα βασικό παράδειγμα ενός αυτοκινούμενου ρομπότ είναι το αυτόνομο καθοδηγούμενο όχημα ή αλλιώς *Automated Guided Vehicle* (**AGV**).

Το AGV είναι ένα αυτοκινούμενο ρομπότ που για να κινηθεί αυτόνομα , ακολουθεί χρωματικούς δείκτες στο πάτωμα.

Για την επίτευξη αυτού του στόχου , μπορεί επίσης να χρησιμοποιεί τεχνητή όραση μέσω κάμερας , ή και τεχνολογία αισθητήρων λέιζερ (LiDaR κτλ .) .

Η παρούσα πτυχιακή βασίζεται πάνω σε αυτήν την μορφή , στην τρίτη περίπτωση με την τεχνολογία αισθητήρα λέιζερ.

AGV σε βιομηχανικό περιβάλλον

Βιομηχανικά ρομπότ (Industrial robots)

Τα αποκαλούμενα βιομηχανικά ρομπότ, παραπέμπουν συνήθως σε ρομποτικούς βραχίονες πολλών βαθμών ελευθερίας, όπου το ένα άκρο τους είναι συνδεδεμένο σε σταθερή επιφάνεια και τα υπόλοιπα μέλη κινούνται και περιστρέφονται ελεύθερα προς επίτευξη της εργασίας που έχουν προγραμματιστεί να εκτελούν .

Ρομποτικός βιομηχανικός βραχίονας

Άλλοι τύποι είναι :

- **Αρθρωτά ή σπονδυλωτά ρομπότ.**
- **Συνεργατικά ή συλλογικά ρομπότ (Collaborative robots).**
- **Στρατιωτικά Robot.**
- **Factory robots**

2. ΠΕΡΙΓΡΑΦΗ ΤΗΣ ΠΤΥΧΙΑΚΗΣ ΕΡΓΑΣΙΑΣ

2.1.Ανάλυση Υλοποίησης

Συνολικά **κύρια** εξαρτήματα (Hardware) που χρησιμοποιήθηκαν για την υλοποίηση:

- Σασί - σώμα του ρομπότ κυκλικής μορφής από πλαστικό υλικό.

Σώμα

- Κεντρική πλακέτα επεξεργασίας: Raspberry Pi 2 Model B+

  Το Raspberry Pi είναι ένας πλήρης υπολογιστής ενσωματωμένος σε έναν ενιαίο πίνακα κυκλωμάτων , με μικροεπεξεργαστή , μνήμη , είσοδο / έξοδο (GPIO) και άλλα χαρακτηριστικά που απαιτούνται από έναν λειτουργικό υπολογιστή.

Raspberry Pi 2 Model B+

- Κινητήρες πλοήγησης, ηλεκτρικοί (12V), με κιβώτιο ταχύτητας εγκατεστημένο αναλογίας 1 προς 28, για μεγαλύτερη μετάδοση ροπής και χαμηλότερες στροφές, και εγκατεστημένους rotary encoder για την δυνατότητα μέτρησης πλήθους περιστροφών.

Κινητήρες 12 V

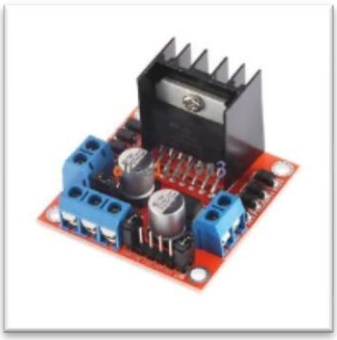
- Πλακέτα PiFace Digital 2.
  Το PiFace Digital 2 έχει σχεδιαστεί για να συνδέεται με το GPIO του Raspberry Pi B +. Με το PiFace Digital 2 ανιχνεύουμε την κατάσταση των αισθητήρων και μόλις εντοπιστεί αυτή η κατάσταση, αλληλεπιδρά με το Raspberry Pi, το οποίο καθορίζει τον τρόπο ανταπόκρισης σε αυτήν την κατάσταση. Κύρια λειτουργία του είναι η οδήγηση των εξόδων των ηλεκτροκινητήρων με έλεγχο στροφών.

PiFace Digital 2

- RP LiDAR A1M8. Αισθητήρας laser, συνεχόμενης περιστροφής 360 μοιρών για την δυνατότητα χαρτογράφησης του περιβάλλοντος.

RP LiDAR A1 M8

- L298N Dual H-Bridge Motor driver. Ο motor driver είναι μία πλακέτα μετάφρασης και ενίσχυσης ρεύματος. Αρχικά λαμβάνει κάποιο σήμα ελέγχου χαμηλού ρεύματος από τον επεξεργαστή, και στην συνέχεια δίνει σήμα υψηλότερου ρεύματος προς κάποιον κινητήρα ώστε να λειτουργήσει επιθυμητά.



L298N Motor Driver

- Η τροφοδοσία γίνεται μέσω μπαταρίας Li-Po τριών κελιών (τάση πλήρης φόρτισης 12.6V), χωρητικότητας 5000 mAh. Οι κινητήρες λειτουργούν με ρεύμα απευθείας από την μπαταρία, με δυνατότητα πλήρης λειτουργίας χαρτογράφησης και κίνησης για 4 ώρες.

LiPo μπαταρία

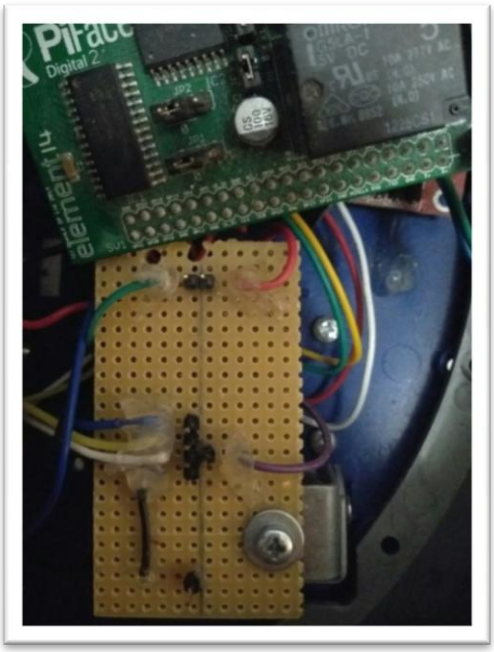
- Μετατροπέας DC-DC (stepdown) για τις ανάγκες σταθερής τροφοδοσίας της πλακέτας του επεξεργαστή ,όπως και των υπόλοιπων ηλεκτρονικών εξαρτημάτων που απαιτούν σταθερή τάση της τάξεως των 5V.

πλακέτα DC-DC stepdown

Δευτερεύοντα εξαρτήματα που υλοποιήθηκαν και χρησιμοποιήθηκαν:

- Χειροποίητη πλακέτα στήριξης και λειτουργίας του PiFace αποσπασμένη από τον επεξεργαστή Pi .
  Το PiFace σαν πλακέτα, εγκαθίσταται (από την εταιρεία παραγωγής) επάνω από τις θέσεις GPIO του Raspberry Pi, με αποτέλεσμα να απομονώνει όλα τα GPIO του ελεγκτή τα οποία όμως δεν χρειάζεται για να λειτουργήσει.

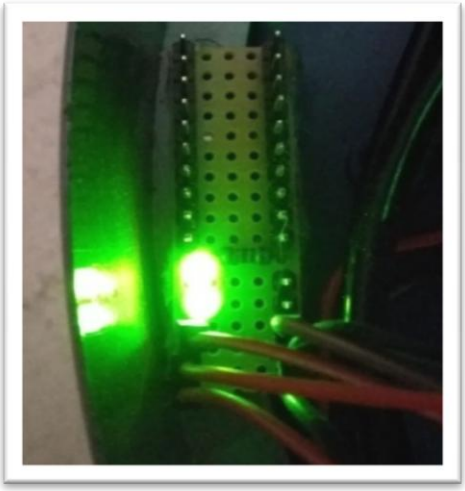
  Αυτό έχει ως συνέπεια, την αδυναμία χρήσης αρκετών εισόδων - εξόδων του ελεγκτή, οι οποίες είναι απαραίτητες για χρήση διαφόρων εφαρμογών, όπως, σύνδεση και λειτουργία διαφόρων αισθητήρων που επιθυμεί ο χρήστης .



Χειροποίητη πλακέτα στήριξης

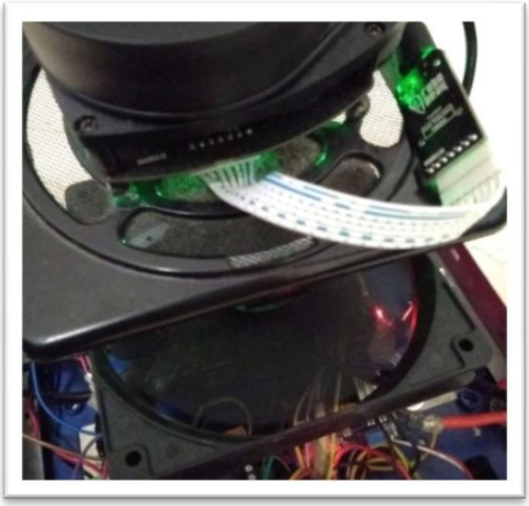
  Κατόπιν μελέτης και έρευνας, διαπιστώσαμε ποιες ήταν οι απαραίτητες θύρες (GPIO) που χρειάζεται για να λειτουργήσει το Pi Face Digital 2, και έτσι δημιουργήσαμε μία ξεχωριστή πλακέτα η οποία μέσω καλωδίων, συνδέεται μόνον στα απαραίτητα GPIO του ελεγκτή, και στην συνέχεια τροφοδοτεί το Pi Face. Με αυτόν τον τρόπο «ελευθερώθηκαν» όλες οι υπόλοιπες θύρες, για περαιτέρω χρήση .

- Δημιουργία χειροποίητης πλακέτας πολλαπλών εξόδων τάσης 5V/3A για τις ανάγκες ταυτόχρονης τροφοδοσίας πολλαπλών αισθητήρων και περιφερειακών ηλεκτρονικών εξαρτημάτων .
(Εγκατεστημένα led για την διευκόλυνση αναγνώρισης ύπαρξης τροφοδοσίας) .



Πλακέτα πολλαπλών εξόδων 5V/3A

- Ανεμιστήρας ψύξης ελεγκτή και περιφερειακών συστημάτων προσαρμοσμένος σε ειδική βάση που τον σταθεροποιεί . Επάνω από την βάση στηρίζεται ο αισθητήρας 360° laser.



Ανεμιστήρας ψύξης

- Ελεύθερος τροχός οριζόντιας κύλισης προσαρμοσμένος σε βάση για την δυνατότητα ισορροπίας του ρομπότ .

τροχός κύλισης

- Ρόδες από λάστιχο, για την αποφυγή ολίσθησης των τροχών στο έδαφος.

ρόδες μετάδοσης κίνησης

- Θήκη για την μπαταρία κάτω από το σκελετό για εξοικονόμηση χώρου και αποφυγή περαιτέρω θερμικής συμφόρησης στον χώρο των ηλεκτρονικών .

θήκη μπαταρίας στο κάτω μέρος του οχήματος

- Διακόπτης παροχής ενέργειας σε όλα τα συστήματα .

Κεντρικός διακόπτης ενέργειας με φωτεινή ένδειξη λειτουργίας

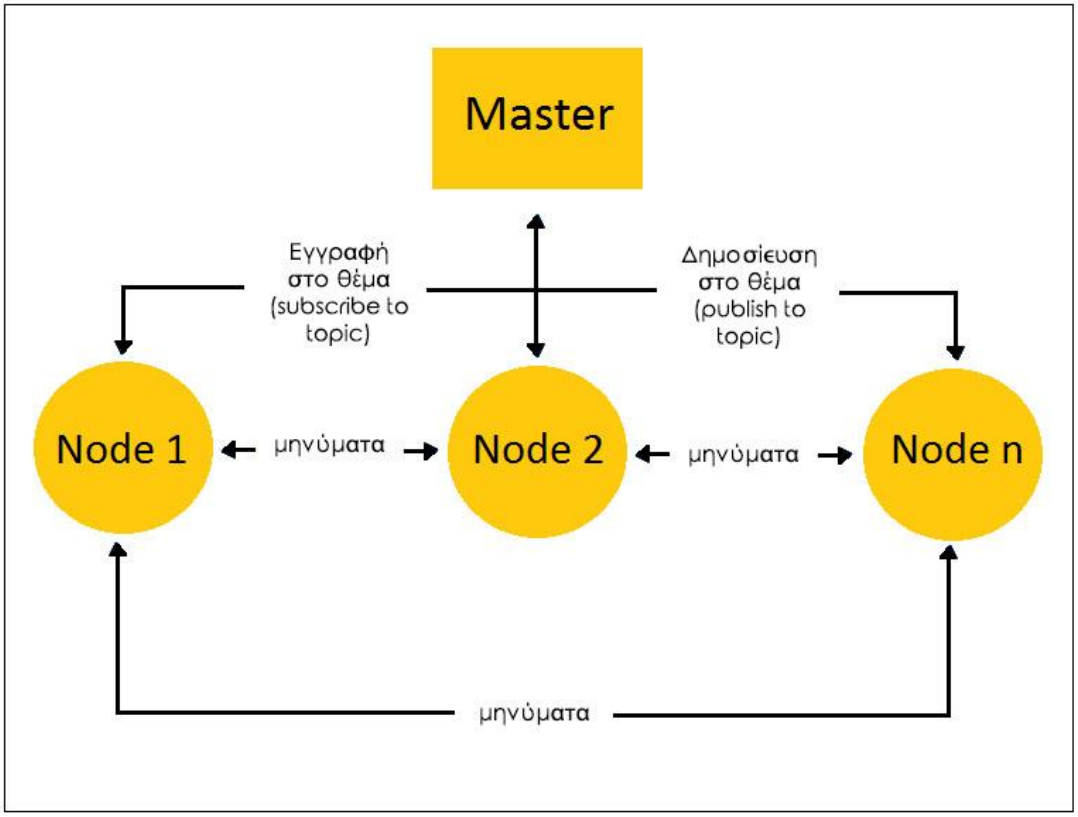
2.2.Λογισμικό

2.2.1. ROS (Robot Operating System)

Ένα από τα πλέον διαδεδομένα λειτουργικά συστήματα ανάπτυξης λογισμικού αυτόνομων οχημάτων είναι το Robot Operating System(ROS). Το ROS είναι ένα ενδιάμεσο λογισμικό ρομποτικής (δηλαδή συλλογή πακέτων λογισμικού για ανάπτυξη λογισμικού ρομπότ) και είναι το βασικό λειτουργικό σύστημα που χρησιμοποιήθηκε στην παρούσα διπλωματική εργασία ως το βασικό μέσο επικοινωνίας μεταξύ του AGV και του ηλεκτρονικού υπολογιστή. Η έκδοση του ROS που χρησιμοποιήθηκε στην παρούσα υλοποίηση είναι η Kinetic.

Παρόλο που το ROS δεν είναι λειτουργικό σύστημα, παρέχει υπηρεσίες σχεδιασμένες για ένα ετερογενές σύμπλεγμα υπολογιστών, όπως η αφαίρεση υλικού, ο έλεγχος συσκευών χαμηλού επιπέδου, η εφαρμογή κοινώς χρησιμοποιούμενων λειτουργιών, η μετάδοση μηνυμάτων μεταξύ διαδικασιών και η διαχείριση πακέτων. Τα σύνολα των διαδικασιών που βασίζονται στο ROS αναπαριστώνται σε μια αρχιτεκτονική γραφημάτων, όπου η επεξεργασία και η λειτουργία λαμβάνει χώρα σε κόμβους που μπορούν να επικοινωνήσουν και να ανταλλάσουν μηνύματα μεταξύ τους αλλά και με τα εξαρτήματα του οχήματος, τα αισθητήρια, τους κινητήρες, άλλα κυκλώματα και να ελέγχουν και να επηρεάζουν την κατάστασή τους.
Το ROS αναπτύχθηκε στο εργαστήριο τεχνητής νοημοσύνης του Stanford το 2007, και την τελευταία δεκαετία έχει γίνει το κυρίαρχο middle-ware frameworkγια ρομποτικές εφαρμογές. Οι βιβλιοθήκες και τα εργαλεία του είναι ανοιχτού κώδικα, γραμμένα σε C++ και Python για συστήματα Unix . Συγκεκριμένα το μόνο λειτουργικό σύστημα για το οποίο το ROS χαρακτηρίζεται πλήρως υποστηριζόμενο είναι τα Ubuntu Linux, ενώ σε συστήματα όπως Fedora Linux, macOS και Microsoft Windows βρίσκεται ακόμα σε πειραματικό στάδιο και υποστηρίζεται μόνο από την ανοιχτή κοινότητα χρηστών.
Η λογική πίσω από τον τρόπο λειτουργίας του ROS είναι ότι κάθε μέρος ενός ρομπότ (ή κάθε ρομπότ σε ένα σύστημα πολλών ρομπότ) αποτελεί ένα ξεχωριστό κόμβο (node) στο σύστημα προγραμματισμένος να εκτελεί το δικό του κομμάτι εργασίας, ανεξαρτήτως αλλά και σε επικοινωνία με τους υπόλοιπους. Η επικοινωνία γίνεται μέσω των μηνυμάτων που μεταδίδονται ανάμεσα στους κόμβους μέσω μίας απλής TCP διασύνδεσης. Αν ένας κόμβος είναι εγγεγραμένος (subscriber) σε ένα θέμα (topic) μπορεί να λαμβάνει όλα τα μηνύματα που δημοσιεύονται (published) σε αυτό από οποιονδήποτε άλλο κόμβο. Για παράδειγμα, ένας κόμβος που εκτελεί τη λειτουργία της σάρωσης του χώρου με δέσμεςlaser (LiDAR), συνδεδεμένος με έναν ανάλογο κόμβο κίνησης, μπορεί να ενημερώσει το σύστημα για τη θέση ενός αντικειμένου και

ο κόμβος που εκτελεί τη λειτουργία της κίνησης, λαμβάνοντας την πληροφορία αυτή να εκτελέσει την ανάλογη κίνηση αποφυγής ή σχεδιασμού πορείας.



Παρόμοια λειτουργία εκτελούν και οι υπηρεσίες του ROS (services), όπου ένας κόμβος είναι σχεδιασμένος να παρέχει μία υπηρεσία (server) και οποιοσδήποτε άλλος μπορεί να κάνει αίτηση (request) και να λάβει την αντίστοιχη απάντηση (reply).Για να μπορέσει να λειτουργήσει σωστά η υλοποίηση της πτυχιακής, θα πρέπει να έχουμε έναν υπολογιστή ελέγχου που θα έχει εγκατεστημένο το λειτουργικό σύστημα Ubuntu μαζί με το ROS Kinetic.

Έτσι ο υπολογιστής θα μας μπορεί να τρέχει τους αλγόριθμους που απαιτούν μεγάλη υπολογιστική ισχύ, καθώς παίρνει δεδομένα απο το Raspberry Pi και τους αισθητήρες του. Επομένως και το Raspberry Pi θα πρέπει να εξοπλιστεί με το αντίστοιχο λειτουργικό σύστημα Ubuntu, συγκεκριμένα το Ubuntu Mate για ARM. Έτσι, το Raspberry Pi χρειάζεται μόνο τις βιβλιοθήκες και τα πακέτα που χρειάζονται για να λειτουργήσουν οι αισθητήρες του και να "ακούει" τις εντολές του κύριου υπολογιστή ελέγχου, έτσι ώστε να εκτελεί τις λειτουργίες του. Το ROS Kinetic υποστηρίζεται απο την έκδοση Xenial (Ubuntu 16.04).

2.2.2 Πακέτα ανοικτού λογισμικού του ROS

Το Robot Operating System προσφέρει την επιλογή ενσωμάτωσης διαφόρων πακέτων ανοιχτού λογισμικού στην εφαρμογή μας. Το λογισμικό αυτό που εμπεριέχεται στο ROS, είναι χωρισμένο σε πακέτα. Με τον όρο πακέτα εννοούμε ένα σύνολο από βιβλιοθήκες, αρχεία κώδικα σε διάφορες γλώσσες προγραμματισμού, ROS nodes, αρχεία ρυθμίσεων, αρχεία εκκίνησης, μοντέλα κ.α. τα οποία αποτελούν ή μια ολοκληρωμένη εφαρμογή ή κάποιο βοήθημα.
Τα πακέτα αυτά είναι προσβάσιμα σε διαδικτυακούς αποθηκευτικούς χώρους που ονομάζονται repositories, όπως είναι το GitHub. Μέσω αυτών των repositories, ο χρήστης κατεβάζει το πακέτο που τον διευκολύνει με βάση την έκδοση του ROS και τον τύπο του υλικού της εφαρμογής του και το ενσωματώνει σε αυτήν έπειτα απο κάποιες ρυθμίσεις.

Ο σκοπός ύπαρξης των πακέτων αυτών είναι η επαναχρησιμοποίηση χρήσιμων εργαλείων για την ανάπτυξη εφαρμογών, αποφεύγοντας την δημιουργία αυτών από την αρχή. Βέβαια, όταν μια εφαρμογή έχει σκοπό να γίνει εμπορική, ο χρήστης είναι υποχρεωμένος να πάρει την έγκριση του δημιουργού του κάθε πακέτου, μαζί με άλλες διαδικασίες έτσι ώστε να αποφύγει διάφορες κατηγορίες, πάντα με βάση τον τύπο της άδειας του κάθε πακέτου λογισμικού, και τα δικαιώματα του δημιουργού.
Κάποιες από αυτές τις άδειες περιέχουν περιοριστικούς όρους όπως η υποχρεωτική αναφορά στο όνομα του δημιουργού ή κατόχου των πνευματικών δικαιωμάτων, καθώς αυτοί οι όροι δεν περιορίζουν τις προηγούμενες ελευθερίες τροποποίησης και διακίνησης, υπάρχουν όμως και άδειες που δεν επιτρέπουν την τροποποίηση μιας βιβλιοθήκης ή ενός πακέτου.

2.2.3 Εγκατάσταση Ubuntu στον υπολογιστή ελέγχου

Ο πιο γρήγορος τρόπος για να κατεβάσουμε τη LTS έκδοση του Ubuntu είναι από τη σελίδα τους. Οι εκδόσεις LTS (Long Term Support, Υποστήριξη Μακράς Διαρκείας) είναι οι πιο σταθερές εκδόσεις του Ubuntu. Κυκλοφορούν κάθε δύο και υποστηρίζονται με updates για πέντε χρόνια. Η έκδοση amd64 θα εγκαταστήσει την 64-bit έκδοση του Ubuntu. Το όνομα προέρχεται από τους δημιουργούς της αρχιτεκτονικής 64-bit, την AMD, δεν είναι περιορισμός για τον επεξεργαστή οπότε λειτουργεί πλήρως και σε επεξεργαστές Intel.
Αφού τα κατεβάσουμε λοιπόν, θα πρέπει να δημιουργήσουμε ένα μέσο εγκατάστασης, δίσκο DVD ή USB δίσκο.

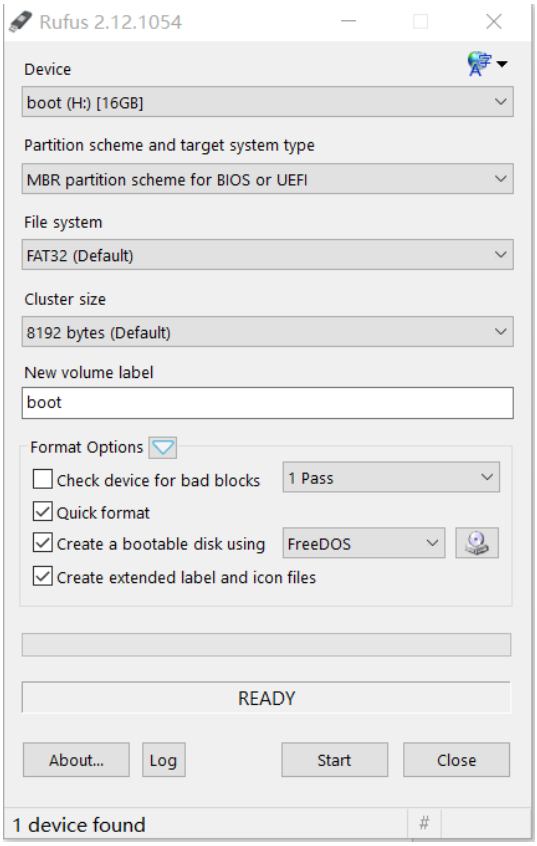
Η εγγραφή σε DVD γίνεται απευθείας από τα Windows, με ένα απλό δεξί κλικ. Μπορούμε όμως να χρησιμοποιήσουμε και κάποιο πρόγραμμα εγγραφής CD/DVD.

Για τη δημιουργία USB για την εγκατάσταση Ubuntu, προτείνεται η χρήση των προγραμμάτων Rufus, ή Unetbootin. Το πρώτο βήμα για την εγκατάσταση Ubuntu είναι να κάνουμε εκκίνηση από το μέσο εγκατάστασης, DVD ή USB. Γενικά το Ubuntu έχει εξαιρετική υποστήριξη για τα βασικά υποσυστήματα του υπολογιστή (κάρτα γραφικών, κάρτα ήχου, κάρτα δικτύου).

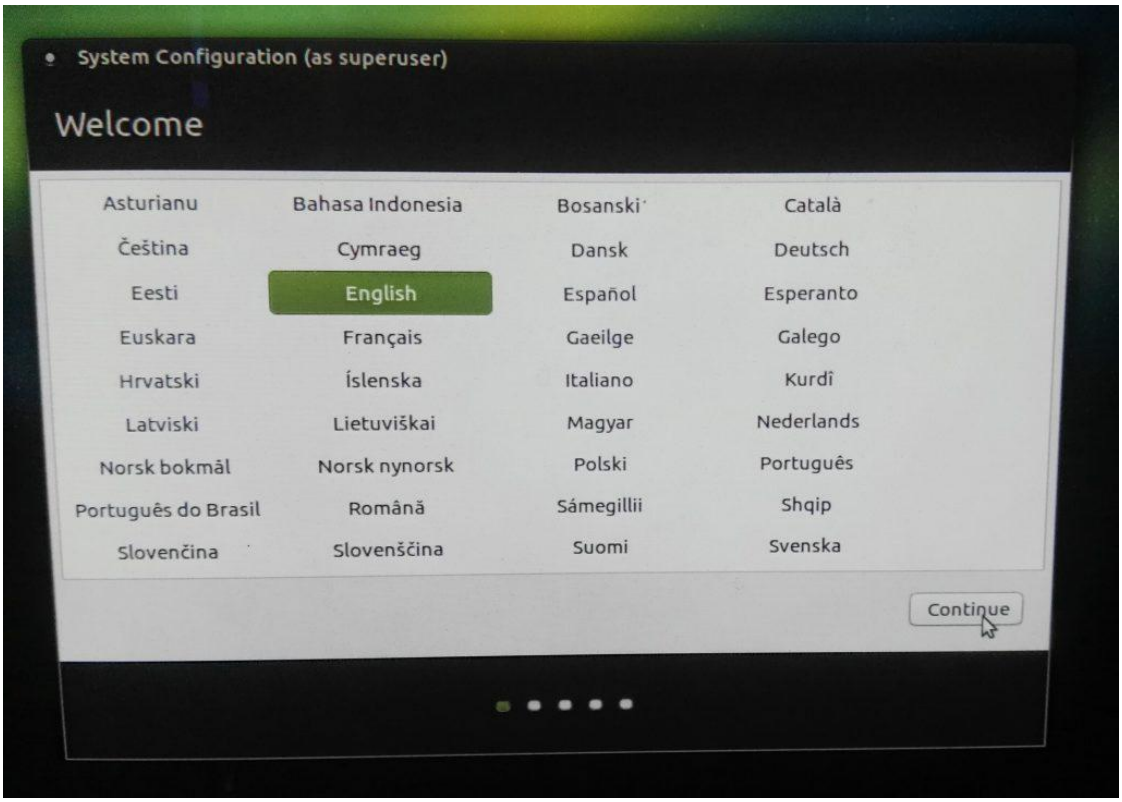
Μόλις κάνουμε εκκίνηση απο το μέσο εγκατάστασης, επιλέγουμε την γλώσσα που επιθυμούμε κι ακολουθούμε τις οδηγίες για τον χώρο στο δίσκο, την ώρα, το όνομα χρήστη και τον κωδικό. Από εδώ, η εγκατάσταση Ubuntu συνεχίζει 100% αυτόματα. Σε λίγα λεπτά, η εγκατάσταση Ubuntu έχει ολοκληρωθεί, και το σύστημα θα μας ζητήσει να κάνουμε επανεκκίνηση.

2.2.4 Εγκατάσταση Ubuntu Mate (ARM) στο Raspberry Pi

Αρχικά κατεβάζουμε τα Ubuntu Mate 16.04 (armv7) για το Raspberry Pi απο την σελίδα του Ubuntu Mate. Αφού τα κατεβάσουμε, θα πρέπει να φορτώσουμε τα Ubuntu Mate στην κάρτα μνήμης του Raspberry Pi. Για να το πετύχουμε αυτό θα πρέπει να έχουμε έναν μετατροπέα καρτών microSD σε USB ώστε να συνδεθεί στον υπολογιστή μας, ή μια συσκευή ανάγνωσης καρτών. Σε αυτό το σημείο πρέπει να προτείνω μια καλή κάρτα τουλάχιστον 16GB ταχύτητας τουλάχιστον Class 10.
Αυτό είναι πολύ σημαντικό, όχι μόνο για τον χώρο και την ταχύτητα του Raspberry Pi, αλλά και την αντοχή του στον χρόνο. Μια κάρτα που δεν πληρεί τις προϋποθέσεις θα δημιουργήσει πολλά προβλήματα, με το βασικότερο να είναι η αλλοίωση των δεδομένων της κάρτας. Αυτό μπορεί να αποβεί πραγματικά καταστροφικό για την εφαρμογή, ειδικά σε συνθήκες απότομης διακοπής τροφοδοσίας ή μη κανονικής διακοπής λειτουργίας. Για την διαμόρφωση της κάρτας και την φόρτωση του λειτουργικού συστήματος σε αυτήν προτείνεται η χρήση του προγράμματος Rufus.



Για την αποφυγή προβλημάτων καλό θα ήταν να ακολουθήσουμε αυτό το μοντέλο φόρτωσης των Ubuntu Mate στην κάρτα μνήμης.
Αφού τελειώσουμε με την εγγραφή, εισάγουμε την κάρτα στο Raspberry Pi, συνδέουμε τα περιφερειακά μας και ανοίγουμε την τροφοδοσία.
Αν τα έχουμε κάνει όλα σωστά θα δούμε αυτή την οθόνη καλωσορίσματος και εγκατάστασης. Επιλέγουμε την γλώσσα, την τοποθεσία και το πληκτρολόγιο που επιθυμούμε.

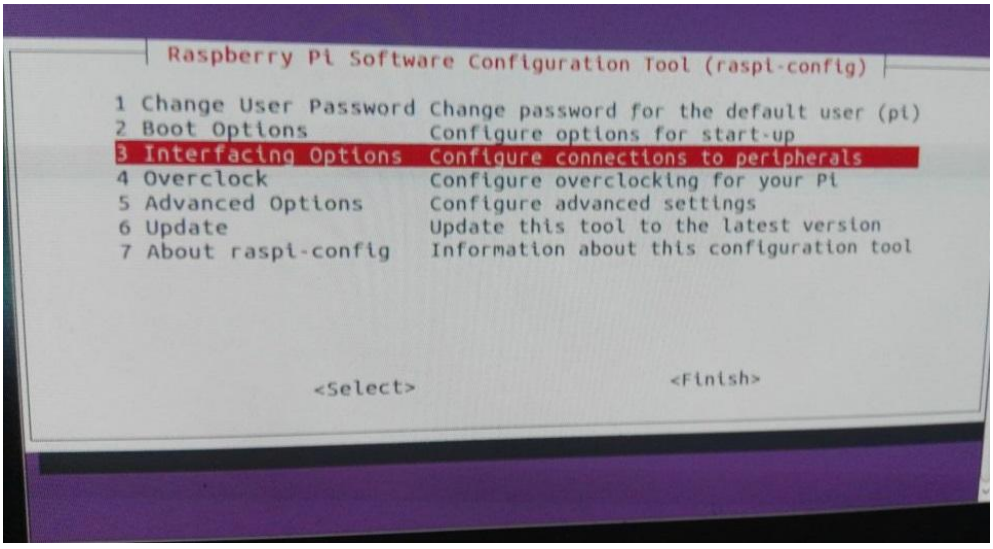


Έπειτα θέτουμε το όνομα χρήστη, το όνομα του υπολογιστή και τον κωδικό μας και σε λίγα λεπτά τα Ubuntu Mate έχουν εγκατασταθεί. Συνδεόμαστε στο Wifi μας πατώντας στο μενού πάνω δεξιά.

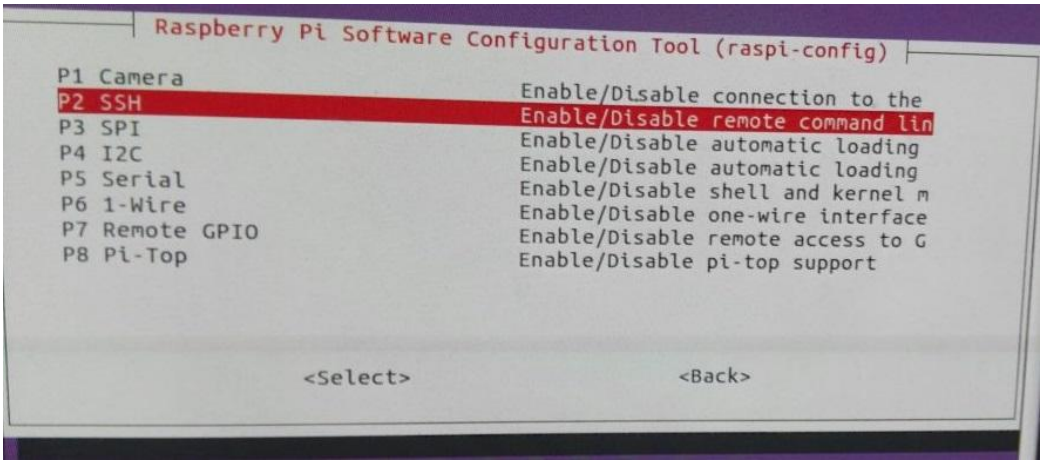
Στη συνέχεια πρέπει να ενεργοποιήσουμε την υπηρεσία SSH (Secure Socket Shell) για την επικοινωνία με το Raspberry Pi δίχως οθόνη. Το SSH είναι ένα κρυπτογραφικό πρωτόκολλο δικτύου για την ασφαλή λειτουργία των υπηρεσιών δικτύου σε ένα μη ασφαλές δίκτυο. Οι τυπικές εφαρμογές περιλαμβάνουν την απομακρυσμένη εντολή σύνδεσης γραμμής εντολών και την εκτέλεση απομακρυσμένης εντολής, αλλά οποιαδήποτε υπηρεσία δικτύου μπορεί να εξασφαλιστεί με SSH. Παρέχει δηλαδή ένα ασφαλές κανάλι μέσω ενός μη ασφαλούς δικτύου.
Ανοίγουμε ένα τερματικό είτε αναζητώντας τα προγράμματα, είτε πατώντας Ctrl+T και πληκτρολογούμε:

sudo raspi-config

Θα μας εμφανιστεί η παρακάτω οθόνη:



Επιλέγουμε **Interfacing Options** και βλέπουμε αυτή την οθόνη:



Επιλέγουμε SSH και το κάνουμε Enable.
Αντίστοιχα, ενεργοποιούμε και το **SPI** και το **I2C**.

Η σειριακή περιφερειακή διασύνδεση (SPI) είναι ένας δίαυλος διεπαφής που χρησιμοποιείται συνήθως για την αποστολή δεδομένων μεταξύ των μικροελεγκτών και των μικρών περιφερειακών όπως οι καταχωρητές, οι αισθητήρες και κάρτες SD. Χρησιμοποιεί ξεχωριστές γραμμές ρολογιών (SCLK) που καθορίζει την ταχύτητα μεταφοράς των δεδομένων, μαζί με μια γραμμή επιλογής για να επιλέξετε τη συσκευή με την οποία θέλετε να μιλήσετε. Είναι απαραίτητο για την λειτουργία του Pi Face και οποιασδήποτε πλακέτα επέκτασης.

Αντίστοιχα, ενεργοποιούμε και το I2C. Το Πρωτόκολλο Διασυνδεδεμένου Κυκλώματος (I2C) είναι ένα πρωτόκολλο που επιτρέπει πολλαπλά ψηφιακά ολοκληρωμένα κυκλώματα "σκλάβων" να επικοινωνούν με ένα η περισσότερα ολοκληρωμένα κυκλώματα ελέγχου. Όπως και η σειριακή περιφερειακή διασύνδεση (SPI), προορίζεται μόνο για επικοινωνίες μικρών αποστάσεων εντός μιας μόνο συσκευής και είναι ιδανικό για κάθε έργο που απαιτεί μεγάλο αριθμό εξόδων. Μια τυπική χρήση είναι για οθόνες LCD, οι οποίες απαιτούν έως και 16 ακίδες για χρήση. Το I2C για την ίδια δουλειά χρειάζεται μόνο τέσσερις. Δύο για επικοινωνία και δύο για τάση και γείωση.

Μόλις ενεργοποιήσουμε τις διεπαφές κάνουμε επανεκκίνηση.

Αφού ξεκινήσουμε πάλι, πρέπει να βεβαιωθούμε οτι στον δρομολογητή μας έχουμε "δέσει" την τοπική IP που παίρνει το Raspberry Pi με την διεύθυνση MAC της κάρτας δικτύου του. Για παράδειγμα η τοπική IP 192.168.1.5 να είναι πάντα η τοπική IP της κάρτας δικτύου του Raspberry Pi με MAC address [28:ff:3e:02:ac:16]. Καθώς υπάρχουν απεριόριστα είδη δρομολογητών με τις δικές τους διεπαφές, δεν θα αναλύσουμε την διαδικασία. Αφού λοιπόν ολοκληρώσουμε, πρέπει να επεξεργαστούμε το αρχείο hosts. Το αρχείο hosts είναι ένα αρχείο απλού κειμένου βασισμένο στο λειτουργικό σύστημα που χαρτογραφεί ονόματα ονομάτων σε διευθύνσεις IP. Περιέχει γραμμές διευθύνσεων IP στο πρώτο πεδίο και ονόματα κεντρικών υπολογιστών στο δεύτερο πεδίο. Κάθε πεδίο διαχωρίζεται από το χώρο των καρτελών απο ένα κενό. Μπορείτε επίσης να προσθέσετε μια γραμμή σχολίων προσθέτοντας μια δίεση (#) μπροστά.

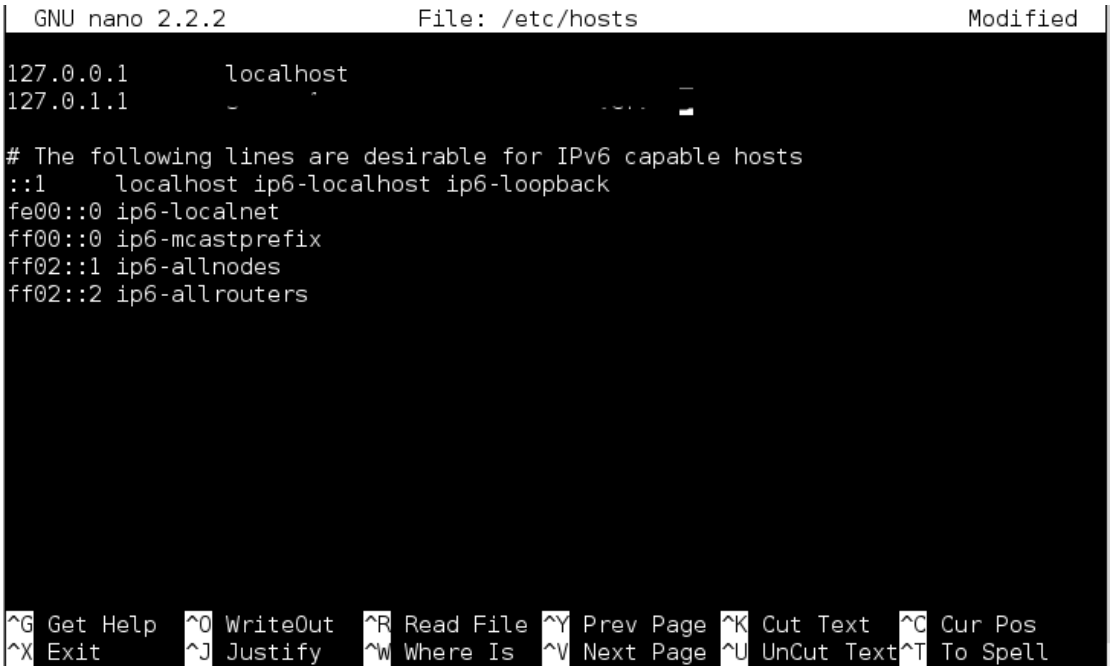
Για να επεξεργαστούμε το αρχείο πρέπει να εγκαταστήσουμε έναν κειμενογράφο. Προσωπική μου προτίμηση είναι ο Nano, οπότε για να τον εγκαταστήσουμε πληκτρολογούμε στο τερματικό:

sudo apt-get install nano

Μόλις ο Nano εγκατασταθεί, για να επεξεργαστούμε το αρχείο hosts πληκτρολογούμε στο τερματικό:

sudo nano /etc/hosts

και πρέπει να βλέπουμε αυτό το αρχείο.



Προσθέτουμε την τοπική IP του Raspberry Pi και το όνομα που του έχουμε θέσει δίπλα, αλλά και την τοπική IP και το όνομα που έχουμε θέσει στον υπολογιστή ελέγχου. Για παράδειγμα :

192.168.1.5 rospi
192.168.1.6 control

Για να αποθηκεύσουμε το έγγραφο, πατάμε Ctrl+X , έπειτα "Y" για επιβεβαίωση, και Enter.

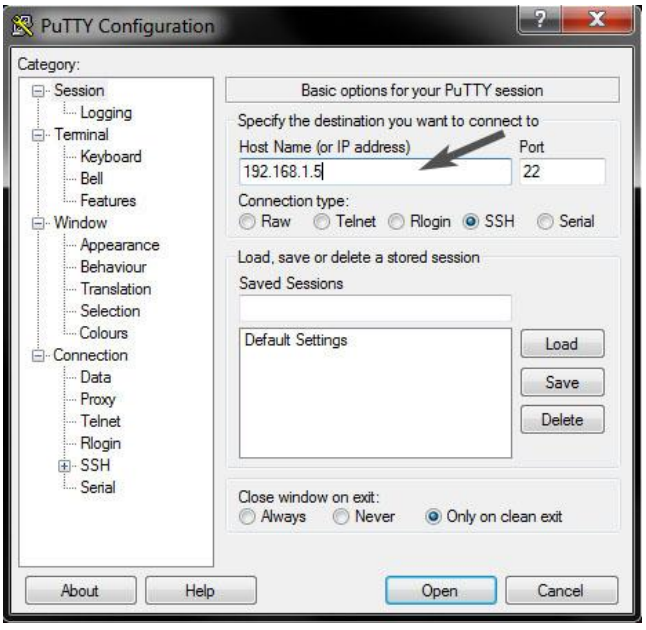
Αντίστοιχα, κάνουμε ακριβώς το ίδιο και στον υπολογιστή ελέγχου για να μπορούν να "βλέπουν" η μία συσκευή την άλλη.
Επειδή ο πυρήνας του ROS θα λειτουργεί απο τον υπολογιστή ελέγχου, για να το πούμε αυτό στο Raspberry Pi , ανοίγουμε ένα τερματικό και πληκτρολογούμε:

echo "ROS_MASTER_URI=http://control:11311" >> ~/.bashrc

και έτσι το Raspberry Pi θα ξέρει κάθε φορά που θα ανοίγει ένα νέο τερματικό, που να ψάξει τον πυρήνα του ROS. Για να αποφύγουμε προβλήματα συνδεσιμότητας, και επειδή δημιουργούμε έργο εκπαιδευτικού περιεχομένου, θα απενεργοποιήσουμε το τείχος προστασίας και του Raspberry Pi καθώς και του υπολογιστή ελέγχου, όχι μόνο για την συνδεσιμότητα του SSH αλλά και αργότερα για το ROS.Εναλλακτικά θα μπορούσαμε απλά να ανοίξουμε τις κατάλληλες πόρτες, αργότερα όμως θα είναι πάρα πολλές. Για να απενεργοποιήσουμε το τείχος προστασίας των Ubuntu και των Ubuntu Mate του Raspberry Pi, ανοίγουμε ένα τερματικό και πληκτρολογούμε:

sudo ufw disable

Αντίστοιχα κάνουμε το ίδιο και για τον υπολογιστή ελέγχου. Αφού ολοκληρώσαμε, μπορούμε πια να συνδεθούμε στο Raspberry Pi και δίχως να συνδέουμε τα περιφερειακά του κάθε φορά. Για να συνδεθούμε στο Raspberry Pi μέσω Windows κατεβάζουμε το PuTTY. Το PuTTY είναι λογισμικό ανοιχτού κώδικα για την σύνδεση στο Raspberry Pi δίχως οθόνη. Πληκτρολογούμε την IP του Raspberry Pi στο τοπικό δίκτυο, στο πεδίο "Host name" και την πόρτα (απο προεπιλογή είναι η 22, μπορείτε να την αλλάξετε) όπως βλέπουμε στην εικόνα και πατάμε το Open.



Έπειτα, θα μας ζητηθεί το όνομα χρήστη και μετά ο κωδικός πρόσβασης στο Raspberry Pi. Με την πληκτρολόγησή τους, εισερχόμαστε στο σύστημα.

Θα δούμε μπροστά μας την γραμμή εντολών όπως και στην εικόνα, και κάναμε επιτυχή σύνδεση.

Έτσι μπορούμε εύκολα να διαχειριστούμε βασικές λειτουργίες του Raspberry Pi καθώς βρίσκεται πάνω σε κάποιο όχημα, δίχως οθόνη, πληκτρολόγιο και ποντίκι.
Πριν προχωρήσουμε στην εγκατάσταση του ROS, θα πρέπει να ενημερώσουμε τα λειτουργικά συστήματα και του Raspberry Pi και του υπολογιστή ελέγχου ώστε να έχουμε την πιο ενημερωμένη έκδοση.
Ανοίγουμε ένα τερματικό και πληκτρολογούμε:

sudo apt-get update -y

sudo apt-get upgrade -y

και στο Raspberry Pi και στον υπολογιστή ελέγχου και περιμένουμε να τελειώσουν. Έπειτα για να ενημερώσουμε και το Kernel του Raspberry Pi, τρέχουμε την εντολή μόνο στο Raspberry Pi:

sudo rpi-update

και έχουμε τελειώσει με τις ενημερώσεις. Δεν θα κάνουμε ξανά ενημέρωση μετά την εγκατάσταση του ROS και των πακέτων του, καθώς αυτό είναι σχεδόν σίγουρο οτι θα προκαλέσει προβλήματα στο έργο και στις ρυθμίσεις μας.

2.2.5 Εγκατάσταση ROS Kinetic στο Raspberry Pi και στον υπολογιστή ελέγχου.

Η διαδικασία εγκατάστασης του ROS στα Ubuntu Mate είναι διαφορετική απο τις εκδόσεις λειτουργικών συστημάτων που ο κατασκευαστής προτείνει να λειτουργεί το Raspberry Pi (Raspbian).

Ξεκινώντας πρέπει να εγκαταστήσουμε τα αποθετήρια του ROS στο αρχείο sources.list. Ανοίγουμε ένα τερματικό και πληκτρολογούμε:

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

Με αυτόν τον τρόπο ο υπολογιστής μας θα δέχεται ενημερώσεις και λογισμικό από την τοποθεσία package.ros.org. Έπειτα θα εγκαταστήσουμε το προσωπικό μας κλειδί. Έτσι εξασφαλίζουμε ότι ο πηγαίος κώδικας που λαμβάνουμε έχει επαληθευτεί. Πληκτρολογούμε στο ίδιο τερματικό την εντολή:

sudo apt-key adv --keyserver hkp://ha.pool.sks-keyservers.net:80 --recv-key 421C365BD9FF1F717815A3895523BAEEB01FA116

και εάν όλα πήγαν καλά θα λάβουμε το μήνυμα " gpg: Total number processed: 1".

Η προτεινόμενη και πιο ολοκληρωμένη επιλογή εγκατάστασης του ROS Kinetic είναι το ROS Kinetic Desktop Full και για την εγκατάσταση του πληκτρολογούμε στο τερματικό την παρακάτω εντολή:

sudo apt-get install ros-kinetic-desktop-full

και όταν ερωτηθούμε αν θέλουμε όντως να το εγκαταστήσουμε, πληκτρολογούμε "y".
Επίσης εγκαθίστανται κάποια βασικά πακέτα του ROS που είναι απαραίτητα για την λειτουργία του καθώς και την συνεργασία του με άλλα προγράμματα. Παρόλα αυτά, εάν χρειαστούμε κάτι παραπάνω, το μόνο που χρειάζεται να κάνουμε είναι να ψάξουμε στην βιβλιοθήκη του με την εντολή:

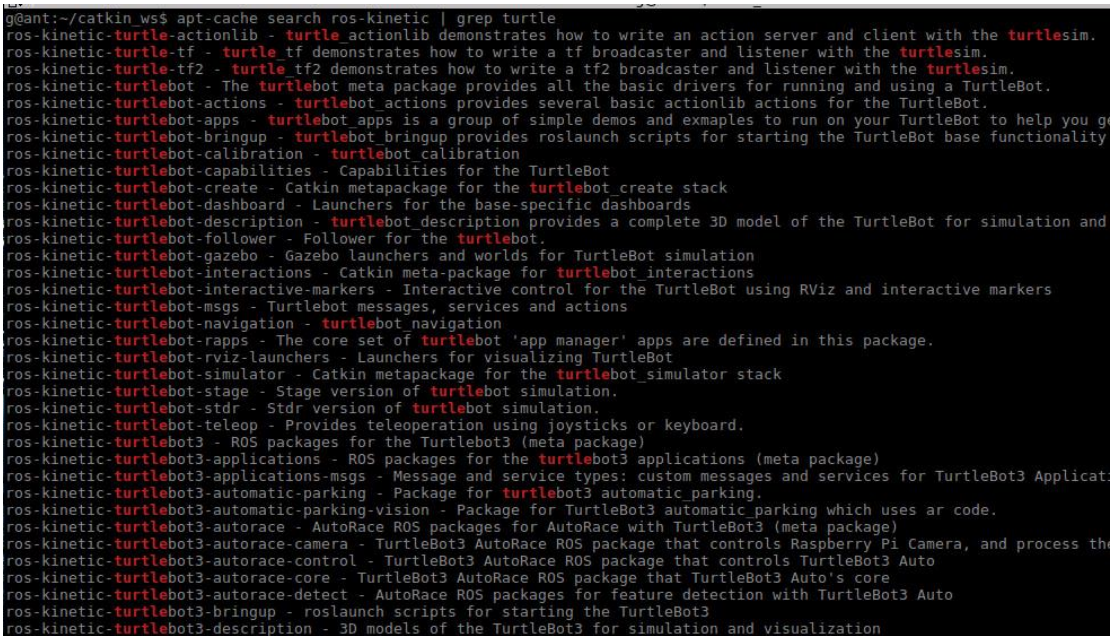
apt-cache search λέξη κλειδί

και πιο συγκεκριμένα για το ROS Kinetic η λέξη κλειδί μας ξεκινάει με την λέξη ros-kinetic και συνεχίζει με το πακέτο που επιθυμούμε.

Για πιο συγκεκριμμένη αναζήτηση, μπορούμε να φιλτράρουμε τα αποτελέσματα με μια λέξη που επιθυμούμε, χρησιμοποιώντας την εντολή grep παράλληλα, για παράδειγμα:

apt-cache search ros-kinetic | grep turtle

Το αποτέλεσμα επισημαίνεται με χρωματισμό με βάση τη λέξη που βάλαμε για φίλτρο, όπως βλέπουμε στην εικόνα:



Στη συνέχεια, για να χρησιμοποιήσουμε το ROS πρέπει πρώτα να έχουμε αρχικοποιήσει το rosdep. Το rosdep καθιστά δυνατή την εύκολη συναρμολόγηση πηγαίου κώδικα που χρειάζονται πυρηνικά στοιχεία του ROS για να εκτελεστούν. Ανοίγουμε ένα τερματικό και πληκτρολογούμε τις παρακάτω εντολές:

sudo rosdep init
rosdep update

Έπειτα, ρυθμίζουμε το περιβάλλον του ROS. Είναι βολικό αν οι μεταβλητές περιβάλλοντος ROS προστίθενται αυτόματα σε κάθε συνεδρία τερματικού που ανοίγουμε.
Ανοίγουμε ένα τερματικό και πληκτρολογούμε:

echo "source /opt/ros/kinetic/setup.bash" >> ~/.bashrc
source ~/.bashrc

Μέχρι τώρα έχετε εγκαταστήσει ό,τι χρειάζεστε για να εκτελέσετε τα βασικά πακέτα ROS. Για να δημιουργήσετε και να διαχειριστείτε τους δικούς σας χώρους εργασίας ROS, υπάρχουν διάφορα εργαλεία και απαιτήσεις που διανέμονται ξεχωριστά απο την αρχική εγκατάσταση. Για παράδειγμα, το

rosinstall είναι ένα συχνά χρησιμοποιούμενο εργαλείο γραμμής εντολών που σας επιτρέπει να κατεβάσετε εύκολα πολλαπλά πακέτα ROS με μία εντολή.

Για να εγκαταστήσουμε αυτό το εργαλείο και άλλες εξαρτήσεις που είναι απαραίτητες για την κατασκευή πακέτων ROS πληκτρολογούμε στο τερματικό:

sudo apt install python-rosinstall python-rosinstall-generator python-wstool build-essential

Τώρα, για να μπορούμε να δουλέψουμε πρέπει να δημιουργήσουμε ένα περιβάλλον εργασίας (workspace). Πληκτρολογούμε στο τερματικό:

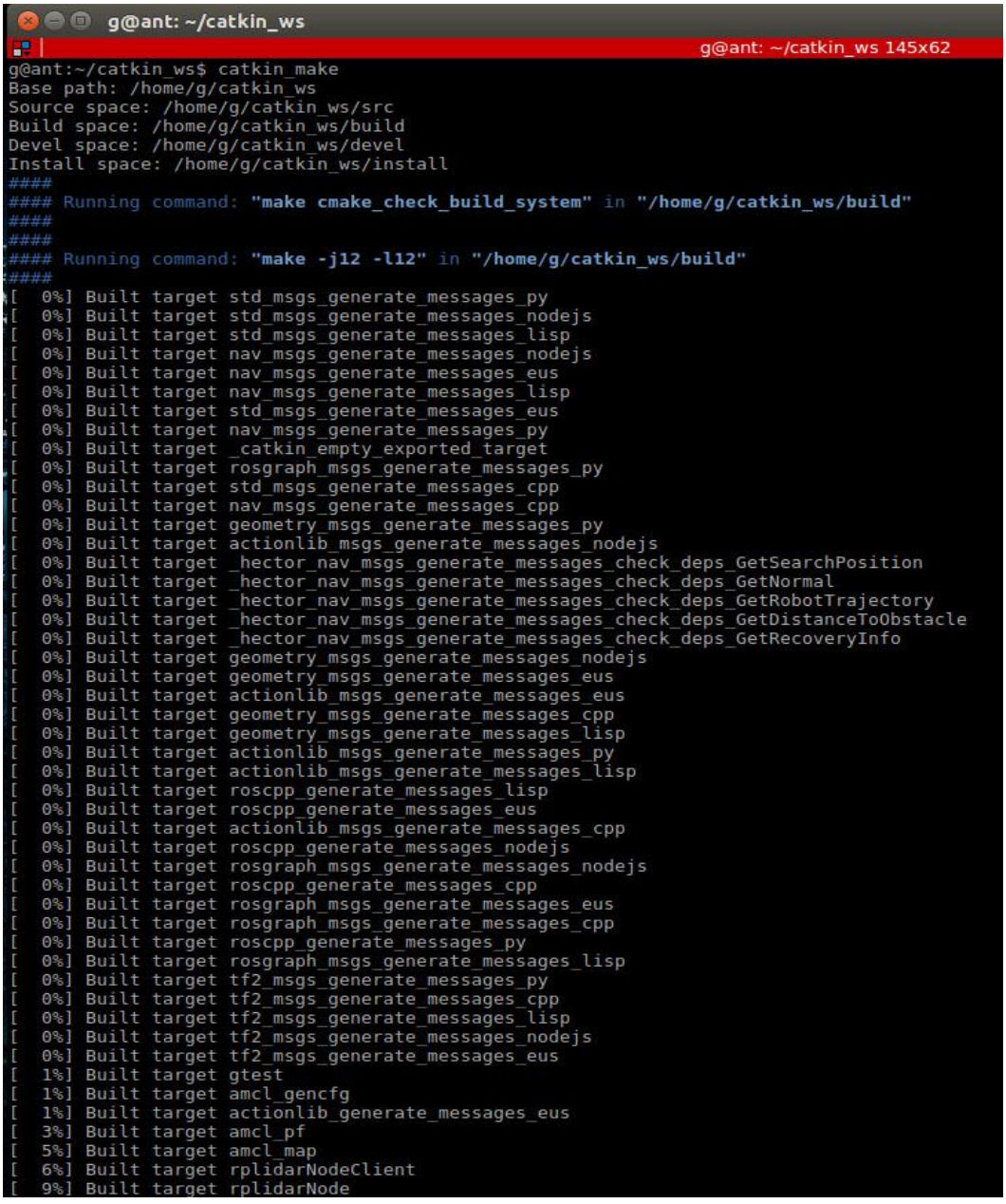
source /opt/ros/kinetic/setup.bash
mkdir -p ~/catkin_ws/src
cd ~/catkin_ws/src
catkin_init_workspace
cd ~/catkin_ws/
catkin_make

Έτσι, δηλώνουμε ποια κομμάτια κώδικα του πακέτου θέλουμε να μεταγλωττίσουμε (στο άδειο περιβάλλον εργασίας), όπως επίσης δηλώνουμε ποια αρχεία του πακέτου μας είναι εκτελέσιμα. Στην συνέχεια η εντολή "catkin_make" εκτελεί μεταγλώττιση.
Έτσι δημιουργούνται δύο φάκελοι μέσα στο workspace μας.Ο φάκελος build που περιέχει μια λίστα με τα εκτελέσιμα πακέτα του workspace, και ο φάκελος devel που περιέχει αρχεία εγκατάστασης. Πρέπει να δείξουμε στο ROS αυτόν τον φάκελο για το εκάστοτε τερματικό μας. Πληκτρολογούμε στο τερματικό:

source devel/setup.bash

Κάνοντας source σε αυτόν τον φάκελο δείχνουμε στο ROS την διαδρομή που πρέπει να εκτελεί τα αρχεία μέσω του περιβάλλοντος εργασίας.

Στην συνέχεια, για κάθε πακέτο που θα προσθέτουμε ή θα επεξεργαζόμαστε απο το περιβάλλον εργασίας θα πρέπει να εκτελούμε την μεταγλώττιση, όπως βλέπουμε στην εικόνα:



Η διαδικασία εγκατάστασης είναι ακριβώς ίδια και για τον υπολογιστή ελέγχου, στα Ubuntu 16.04. Οπότε θα εκτελέσουμε ακριβώς τις ίδιες ενέργειες για την εγκατάσταση του ROS Kinetic.

2.2.6 Εγκατάσταση πακέτων του ROS Kinetic που χρειαζόμαστε στον υπολογιστή ελέγχου

Αρχικά πρέπει να εγκαταστήσουμε το Git. Το Git είναι ένα σύστημα διαχείρισης εκδόσεων αρχείων. Παρακολουθεί τις αλλαγές σε ένα σύνολο αρχείων κρατώντας στιγμιότυπά τους. Το σύνολο στιγμιοτύπων ονομάζεται αποθετήριο (repository).

Στόχοι του Git είναι να υποστηρίξει τη διανεμημένη παραγωγικότητα και την ταχύτητα. Το Git βοηθά τον χρήστη να διατηρήσει την υπευθυνότητα και την πατρότητα για κάθε δράση του, να διατηρεί την ακεραιότητα του έργου και να βελτιώνει τη συνολική επικοινωνία μεταξύ των συνεργατών. Δεν θα επεκταθούμε παραπάνω σχετικά με την φιλοσοφία του, είναι ένα εργαλείο που μας επιτρέπει να προσθέτουμε και να κατεβάζουμε εύκολα πακέτα από την κοινότητα του ROS. Για να το εγκαταστήσουμε, ανοίγουμε ένα τερματικό και πληκτρολογούμε:

sudo apt-get install git-core

Έπειτα πρέπει να προσθέσουμε το πακέτο Navigation Stack, μια στοίβα πλοήγησης που λαμβάνει πληροφορίες 2 διαστάσεων από την οδομετρία, και τους αισθητήρες και φυσικά ένα στόχο και εξάγει εντολές ασφαλούς πλοήγησης που αποστέλλονται σε μια κινητή βάση (move base). Για να δημιουργήσουμε ένα τοπικό αντίγραφο του αποθετηρίου στον υπολογιστή μας και να αντιγράψουμε το project απο το github στον δίσκο μας πρέπει πρώτα να πλοηγηθούμε στον φάκελο src του περιβάλλοντος εργασίας μας. Έτσι, εκτελούμε στο τερματικό:

cd ~/catkin_ws/src
git clone https://github.com/ros-planning/navigation.git

και μόλις το πακέτο αντιγραφεί επιτυχώς στον τοπικό μας δίσκο, εκτελούμε μεταγλώττιση:

catkin_make

Έτσι, προσθέσαμε επιτυχώς το πακέτο της στοίβας πλοήγησης του ROS στο περιβάλλον εργασίας μας. Θα ακολουθήσουμε την ίδια διαδικασία και για τα επόμενα πακέτα που θα χρειαστεί να προσθέσουμε.

Με τον ίδιο τρόπο θα εγκαταστήσουμε και το πακέτο rplidar_ros για την λειτουργία του αισθητήρα LiDAR και τις δοκιμές που χρειάζονται στον υπολογιστή ελέγχου. Ανοίγουμε ένα τερματικό και εκτελούμε:

git clone https://github.com/Slamtec/rplidar_ros

Έτσι δημιουργήσαμε ένα τοπικό αντίγραφο του αποθετηρίου στον υπολογιστή μας, στο περιβάλλον εργασίας μας.
Αντίστοιχα, με τον ίδιο τρόπο θα εγκαταστήσουμε το πακέτο hector_slam για την χρήση του αισθητήρα LiDAR στην χαρτογράφηση του χώρου. Ανοίγουμε ένα τερματικό και εκτελούμε:

git clone https://github.com/tu-darmstadt-ros-pkg/hector_slam

Μόλις αντιγραφεί, θα εγκαταστήσουμε και το πακέτο find_object για την ανίχνευση αντικειμένων μέσω της κάμερας του Raspberry Pi.
Ανοίγουμε ένα τερματικό και εκτελούμε:

sudo apt-get install ros-kinetic-find-object-2d

Τέλος, όταν όλα τα πακέτα εγκατασταθούν, εκτελούμε ξανά τη μεταγλώττιση στο περιβάλλον εργασίας μας:

catkin_make

Έπειτα πρέπει να κάνουμε ρυθμίσεις για να μπορούν αυτά τα πακέτα να έχουν σωστή αλληλεπίδραση με το όχημά μας.

Για να μπορεί το όχημα να κινείται σωστά, έχουμε συνδέσει τους κινητήρες μας με τις εξόδους του L298N Motor driver και τον έλεγχο περιστροφής του κινητήρα και ταχύτητας στην πλακέτα PiFace Digital 2.
Για να μπορούμε να ελέγχουμε με τηλεχειρισμό το όχημά μας, έχουμε γράψει ένα απλό πρόγραμμα για την οδήγηση των κινητήρων όπως θα δείξουμε στο επόμενο κεφάλαιο. Το πρόγραμμα αυτό δέχεται χαρακτήρες απο διασύνδεση Secure Shell, ή μηνύματα τύπου Twist απο το πληκτρολόγιο του ROS.

Το πληκτρολόγιο αυτό στέλνει μηνύματα τύπου Twist του ROS τα οποία περιέχουν εντολές ταχύτητας, γωνιακής ταχύτητας και κατεύθυνσης.
Για να το εγκαταστήσουμε, ανοίγουμε ένα νέο τερματικό και πληκτρολογούμε:

sudo apt-get install ros-kinetic-teleop-twist-keyboard

και για να το τρέξουμε εκτελούμε:

rosrun teleop_twist_keyboard teleop_twist_keyboard.py

Όταν το τρέξουμε επιτυχώς θα δούμε στην οθόνη μας τις οδηγίες ελέγχου του πληκτρολογίου.

Reading from the keyboard and Publishing to Twist!

Moving around:

u i o

j k l

m , .

q/z : increase/decrease max speeds by 10%

w/x : increase/decrease only linear speed by 10%

e/c : increase/decrease only angular speed by 10%

anything else : stop

CTRL-C to quit

2.2.7 Ρυθμίσεις των πακέτων στον υπολογιστή ελέγχου

Μετά την εγκατάσταση των πακέτων πρέπει να κάνουμε κάποιες ρυθμίσεις για να μπορούν τα πακέτα να λειτουργούν με τους δικούς μας αισθητήρες και τις δυνατότητές τους, είτε να βελτιστοποιήσουμε τη λειτουργία τους. Αρχικά πλοηγούμαστε στον φάκελο **/src/hector_slam/hector_slam_launch/launch** του περιβάλλοντος εργασίας μας.
Ανοίγουμε ένα τερματικό και εκτελούμε:

cd ~/catkin_ws/src/hector_slam/hector_slam_launch/launch

και δημιουργούμε ένα αρχείο εκτέλεσης με το όνομα pislam.launch το οποίο περιέχει τις παραμέτρους εκκίνησης της χαρτογράφησης.
Για να το δημιουργήσουμε, εκτελούμε στο τερματικό:

nano pislam.launch

Οι παράμετροι εκκίνησης είναι οι εξής:

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

<launch>

  <arg name="geotiff_map_file_path" default="$(find hector_geotiff)/maps"/>

  <param name="/use_sim_time" value="false"/>

  <node pkg="rviz" type="rviz" name="rviz" args="-d $(find
hector_slam_launch)/rviz_cfg/mapping.rviz"/>

  <include file="$(find hector_mapping)/launch/mapping.launch"/>

  <include file="$(find hector_geotiff)/launch/geotiff_mapper.launch">
    <arg name="trajectory_source_frame_name"
value="scanmatcher_frame"/>
    <arg name="map_file_path" value="$(arg geotiff_map_file_path)"/>
  </include>

</launch>
```

Για να αποθηκεύσουμε, πατάμε "Ctrl+X" , έπειτα "y" για να αποδεχτούμε τις αλλαγές και μετά Enter.

Έπειτα πρέπει να δημιουργήσουμε το αρχείο mapping.rviz το οποίο περιέχει τις παραμέτρους απεικόνισης της χαρτογράφησης στο rviz.
Για να το δημιουργήσουμε, εκτελούμε στο τερματικό:

cd ~/catkin_ws/src/hector_slam/hector_slam_launch/ rviz_cfg
nano mapping.rviz

Οι παράμετροι απεικόνισης στο αρχείο είναι οι εξής:

```
Panels:
  - Class: rviz/Displays
    Help Height: 78
    Name: Displays
    Property Tree Widget:
      Expanded:
        - /Global Options1
        - /Status1
        - /Map1
        - /Path1
        - /Pose1
      Splitter Ratio: 0.5
    Tree Height: 540
  - Class: rviz/Selection
    Name: Selection
  - Class: rviz/Tool Properties
    Expanded:
      - /2D Pose Estimate1
      - /2D Nav Goal1
      - /Publish Point1
    Name: Tool Properties
    Splitter Ratio: 0.588679
  - Class: rviz/Views
    Expanded:
      - /Current View1
    Name: Views
    Splitter Ratio: 0.5
  - Class: rviz/Time
    Experimental: false
    Name: Time
    SyncMode: 0
    SyncSource: ""
Visualization Manager:
  Class: ""
  Displays:
    - Alpha: 0.5
      Cell Size: 1
      Class: rviz/Grid
      Color: 160; 160; 164
      Enabled: true
      Line Style:
```

```
      Line Width: 0.03
      Value: Lines
    Name: Grid
    Normal Cell Count: 0
    Offset:
      X: 0
      Y: 0
      Z: 0
    Plane: XY
    Plane Cell Count: 10
    Reference Frame: <Fixed Frame>
    Value: true
  - Alpha: 0.7
    Class: rviz/Map
    Color Scheme: map
    Draw Behind: false
    Enabled: true
    Name: Map
    Topic: /map
    Value: true
  - Alpha: 1
    Buffer Length: 1
    Class: rviz/Path
    Color: 25; 255; 0
    Enabled: true
    Name: Path
    Topic: /trajectory
    Value: true
  - Alpha: 1
    Axes Length: 1
    Axes Radius: 0.1
    Class: rviz/Pose
    Color: 255; 25; 0
    Enabled: true
    Head Length: 0.3
    Head Radius: 0.1
    Name: Pose
    Shaft Length: 1
    Shaft Radius: 0.05
    Shape: Axes
    Topic: /slam_out_pose
    Value: true
Enabled: true
Global Options:
  Background Color: 48; 48; 48
  Fixed Frame: map
  Frame Rate: 30
Name: root
Tools:
  - Class: rviz/Interact
```

```
    Hide Inactive Objects: true
  - Class: rviz/MoveCamera
  - Class: rviz/Select
  - Class: rviz/FocusCamera
  - Class: rviz/Measure
  - Class: rviz/SetInitialPose
    Topic: /initialpose
  - Class: rviz/SetGoal
    Topic: /move_base_simple/goal
  - Class: rviz/PublishPoint
    Single click: true
    Topic: /clicked_point
 Value: true
 Views:
  Current:
    Class: rviz/XYOrbit
    Distance: 15
    Focal Point:
      X: 0.5
      Y: 0.5
      Z: 1
    Name: Current View
    Near Clip Distance: 0.01
    Pitch: 1
    Target Frame: <Fixed Frame>
    Value: XYOrbit (rviz)
    Yaw: 0
  Saved: ~
Window Geometry:
 Displays:
  collapsed: false
 Height: 846
 Hide Left Dock: false
 Hide Right Dock: false
 Selection:
  collapsed: false
 Time:
  collapsed: false
 Tool Properties:
  collapsed: false
 Views:
  collapsed: false
 Width: 1200
 X: 53
 Y: 60
```

Για να αποθηκεύσουμε, πατάμε "Ctrl+X" , έπειτα "y" για να αποδεχτούμε τις αλλαγές και μετά Enter.

Τέλος, πρέπει να δημιουργήσουμε το αρχείο mapping.launch το οποίο περιέχει τις ρυθμίσεις λειτουργίας του αισθητήρα RPLidar A1 καθώς και τις βελτιστοποιήσεις για την καλύτερη λειτουργία του.

Για να το δημιουργήσουμε, εκτελούμε στο τερματικό:

cd ~/catkin_ws/src/ hector_slam/hector_mapping/launch
nano mapping. launch

Οι ρυθμίσεις εκκίνησης καλής λειτουργίας του αισθητήρα RPLidar A1 στο αρχείο είναι οι εξής:

```
<launch>

  <arg name="tf_map_scanmatch_transform_frame_name" default="scanmatcher_frame"/>
  <arg name="base_frame" default="base_link"/>
  <arg name="odom_frame" default="base_link"/>
  <arg name="pub_map_odom_transform" default="true"/>
  <arg name="scan_subscriber_queue_size" default="5"/>
  <arg name="scan_topic" default="scan"/>
  <arg name="map_size" default="250"/>  <!--metra-->

  <node pkg="hector_mapping" type="hector_mapping" name="hector_mapping" output="screen">


    <!-- Frame names -->
    <param name="map_frame" value="map" />
    <param name="base_frame" value="$(arg base_frame)" />
    <param name="odom_frame" value="$(arg odom_frame)" />
    <!-- Tf use -->
    <param name="use_tf_scan_transformation" value="true"/>
    <param name="use_tf_pose_start_estimate" value="false"/>
    <param name="pub_map_odom_transform" value="$(arg pub_map_odom_transform)"/>


    <!-- Map size / start point -->
    <param name="map_resolution" value="0.055"/> <!-- best so far -->
    <param name="map_size" value="$(arg map_size)"/>
    <param name="map_start_x" value="0.5"/>
    <param name="map_start_y" value="0.5" />
    <param name="map_multi_res_levels" value="2" />

    <!-- Map update parameters -->
```

```
    <param name="update_factor_free" value="0.4"/>   <!-- best so far -->
    <param name="update_factor_occupied" value="0.99" />
    <param  name="map_update_distance_thresh"  value="0.4"/>   <!-- best  so
far -->
    <param name="map_update_angle_thresh" value="0.06" />
    <param name="laser_z_min_value" value = "-1.0" />
    <param name="laser_z_max_value" value = "1.0" />


    <!-- Advertising config -->
    <param name="advertise_map_service" value="true"/>

    <param          name="scan_subscriber_queue_size"          value="$(arg
scan_subscriber_queue_size)"/>
    <param name="scan_topic" value="$(arg scan_topic)"/>

    <!-- Debug parameters -->

    <!--
      <param name="output_timing" value="false"/>
      <param name="pub_drawings" value="true"/>
      <param name="pub_debug_output" value="true"/>
    -->
    <param  name="tf_map_scanmatch_transform_frame_name"  value="$(arg
tf_map_scanmatch_transform_frame_name)" />
  </node>
  <node                pkg="tf"                type="static_transform_publisher"
name="base_to_laser_broadcaster" args="0 0 0 0 0 0 base_link laser 100" />

</launch>
```

Για να αποθηκεύσουμε, πατάμε "Ctrl+X" , έπειτα "y" για να αποδεχτούμε τις αλλαγές και μετά Enter.

2.2.8 Εγκατάσταση πακέτων του ROS Kinetic που χρειαζόμαστε στο Raspberry Pi του οχήματος

Αρχικά πρέπει κι εδώ να εγκαταστήσουμε το Git.

Για να το εγκαταστήσουμε, ανοίγουμε ένα τερματικό και πληκτρολογούμε:

sudo apt-get install git-core

Έπειτα πρέπει να πλοηγηθούμε στον φάκελο src του περιβάλλοντος εργασίας μας. Έτσι, εκτελούμε στο τερματικό:

cd ~/catkin_ws/src

Θα εγκαταστήσουμε και το πακέτο rplidar_ros για την λειτουργία του αισθητήρα LiDAR στο όχημά μας. Ανοίγουμε ένα τερματικό και εκτελούμε:

git clone https://github.com/Slamtec/rplidar_ros

Έτσι δημιουργήσαμε ένα τοπικό αντίγραφο του αποθετηρίου στον υπολογιστή μας, στο περιβάλλον εργασίας μας.

Για την λειτουργία της κάμερας και την ανίχνευση αντικειμένων μέσω αυτής, πρέπει να εγκαταστήσουμε το πακέτο cv_camera,το οποίο χρησιμοποιεί την βιβλιοθήκη OpenCV για να καταγράψει την εικόνα της κάμερας. Το OpenCV είναι μια βιβλιοθήκη λειτουργιών προγραμματισμού που στοχεύει κυρίως στην μετάδοση εικόνας στον υπολογιστή με σκοπό την όραση σε πραγματικό χρόνο.
Ανοίγουμε λοιπόν ένα τερματικό και εκτελούμε:

git clone https://github.com/OTL/cv_camera.git

Τώρα θα φτιάξουμε ένα δικό μας πακέτο για τον τηλεχειρισμό του οχήματος απο τον υπολογιστή ελέγχου. Εφόσον είμαστε ήδη στον φάκελο src του περιβάλλοντος εργασίας μας θα χρησιμοποιήσουμε το εργαλείο **catkin_create_pkg** για να δημιουργήσουμε ένα νέο πακέτο που θα λέγεται "**pi_teleop**" το οποίο θα εξαρτάται απο τις βιβλιοθήκες std_msgs , geometry_msgs και rospy. Για να το δημιουργήσουμε, πληκτρολογούμε στο τερματικό τα παρακάτω:

catkin_create_pkg pi_teleop std_msgs geometry_msgs rospy

Έτσι δημιουργήσαμε ένα φάκελο **pi_teleop** μέσα στον φάκελο **src** που περιέχει ένα αρχείο **package.xml** και ένα αρχείο **CMakeLists.txt**, τα οποία έχουν γεμίσει με όλες τις πληροφορίες του πακέτου που δώσαμε πριν στο εργαλείο catkin_create_pkg, και μπορούμε να τις επεξεργαστούμε μετά, όπως όνομα, έκδοση, email, εξαρτώμενες βιβλιοθήκες και άδεια.

Έπειτα θα φτιάξουμε ένα φάκελο scripts μέσα στο νέο μας πακέτο, αφού πρώτα πλοηγηθούμε σε αυτό, πληκτρολογούμε:

cd pi_teleop
mkdir scripts

Μετά θα πλοηγηθούμε στον φάκελο που δημιουργήσαμε και θα φτιάξουμε ένα αρχείο με το πρόγραμμα που χρειαζόμαστε για τον τηλεχειρισμό του οχήματος. Θα το ονομάσουμε **move_pi.py** (python). Πληκτρολογούμε στο τερματικό:

cd scripts
nano move_pi.py

και έτσι θα ανοίξει ο κειμενογράφος nano. Εκεί, θα γράψουμε το πρόγραμμά μας:

```
#!/usr/bin/env python
import sys, tty, termios, time
from time import sleep
import pifacedigitalio as pfio
import getch

from geometry_msgs.msg import Twist
import rospy
from std_msgs.msg import String

pfd = pfio.PiFaceDigital()
pfio.init()


def mrb():
  pfio.digital_write(2,1)
  return;
def mlb():
  pfio.digital_write(4,1)
  return;
def mrfr():
  pfio.digital_write(3,1)
  return;
def mlfr():
  pfio.digital_write(5,1)
  return;
def res():
  pfio.digital_write(2,0)
  pfio.digital_write(3,0)
  pfio.digital_write(4,0)
  pfio.digital_write(5,0)
  return;
```

```
move1=0
move2=0

dreamlin=0.028
dreamang=0.001

def callback(msg):
        move1 = msg.linear.x;
        move2 = msg.angular.z;


          if(move1 > 0 and move2 == 0):
                  mlfr()
                  mrfr()
                  sleep(dreamlin)
                  res()

          if(move1 < 0):
                  mlb()
                  mrb()
                  sleep(dreamlin)
                  res()

          if(move1 > 0 and move2 > 0 ):
                  mrfr()
              mlb()
                  sleep(dreamang)
                  res()

          if(move1 > 0 and move2 < 0):
                  mlfr()
              mrb()
                  sleep(dreamang)
                  res()

        if(move1 ==0 and move2 < 0):
                  print("\n Teleoperation Ended Successfully!")
                  pfio.init()
              rospy.signal_shutdown("Shutdown Complete")

def listener():

    rospy.init_node('move_pi')
    rospy.Subscriber('cmd_vel', Twist, callback)
    rospy.loginfo("Waiting for data...")
    # spin() simply keeps python from exiting until this node is stopped
     rospy.spin()

if __name__ == '__main__':
    listener()
```

Για να αποθηκεύσουμε, πατάμε "Ctrl+X" , έπειτα "y" για να αποδεχτούμε τις αλλαγές και μετά Enter.
Το πρόγραμμα χρησιμοποιεί την βιβλιοθήκη του PiFace Digital 2 καθώς αυτή είναι η πλακέτα σύνδεσης στην παρούσα υλοποίηση. Χρησιμοποιεί επίσης την βιβλιοθήκη getch που λαμβάνει χαρακτήρες (για τον χειρισμό μέσω Secure Shell), καθώς και τις αντίστοιχες βιβλιοθήκες geometry_msgs και std_msgs του ROS για να μπορεί να κινείται είτε με τα πλήκτρα WASD μέσω διασύνδεσης Secure Shell είτε μέσω του πληκτρολογίου Twist Teleop Keyboard του ROS. Το πληκτρολόγιο αυτό στέλνει μηνύματα τύπου Twist του ROS τα οποία περιέχουν εντολές ταχύτητας, γωνιακής ταχύτητας και κατεύθυνσης.

Η βιβλιοθήκη getch θα πρέπει να εγκατασταθεί για την αντίστοιχη έκδοση python του ROS Kinetic, δηλαδή την έκδοση 2.7.
Για να την εγκαταστήσουμε θα πρέπει πρώτα να έχουμε εγκαταστήσει το σύστημα διαχείρισης πακέτων της python 2.x, το PIP. Ανοίγουμε ένα νέο τερματικό και πληκτρολογούμε:

sudo apt-get install python-pip python-dev build-essential
sudo pip install --upgrade pip
pip install getch

Τέλος, όταν όλα τα πακέτα είναι έτοιμα, και έχουμε εγκαταστήσει τα απαραίτητα εργαλεία επιτυχώς, εκτελούμε τη μεταγλώττιση στο περιβάλλον εργασίας μας:

catkin_make

Μετά την εγκατάσταση των πακέτων θα πρέπει να κάνουμε κάποιες ρυθμίσεις για να μπορούν τα πακέτα να λειτουργούν με τους δικούς μας αισθητήρες και τις δυνατότητές τους σωστά, είτε να βελτιστοποιήσουμε τη λειτουργία τους. Αρχικά θα πρέπει να συνδέσουμε τον αισθητήρα LiDAR και να ανιχνεύσουμε που συνδέθηκε, πληκτρολογώντας την παρακάτω εντολή:

ls -l /dev|grep ttyUSB

Η παρακάτω εντολή θα μας επιστρέψει τις συσκευές USB που έχουν συνδεθεί, με τον αντίστοιχο αριθμό δίπλα τους, πχ **ttyUSB0.**

Μετά θα δώσουμε τα απαραίτητα δικαιώματα εκτέλεσης του αισθητήρα σε αυτή την θύρα, στον διαχειριστή συσκευών udev. Εκεί, θα προσθέσουμε τους κανόνες μας.

Για να το κάνουμε αυτό, θα πλοηγηθούμε στη διαδρομή **/etc/udev/rules.d** πληκτρολογώντας:

cd /etc/udev/rules.d

και θα δημιουργήσουμε ένα αρχείο **rplidar.rules:**

nano rplidar.rules

βάζοντας μέσα στον κειμενογράφο το παρακάτω**:**

KERNEL=="ttyUSB*", ATTRS{idVendor}=="10c4", ATTRS{idProduct}=="ea60", MODE:="0777", SYMLINK+="rplidar"

όπου στο πεδίο **KERNEL** ως θύρα θα βάλουμε την θύρα που μας επέστρεψε η εντολή **ls -l /dev|grep ttyUSB** προηγουμένως.

Για να αποθηκεύσουμε, πατάμε "Ctrl+X" , έπειτα "y" για να αποδεχτούμε τις αλλαγές και μετά Enter.

Μετά θα πρέπει να κάνουμε επανεκκίνηση στον διαχειριστή συσκευών για να φορτώσει τους νέους κανόνες. Πληκτρολογούμε στο τερματικό:

sudo service udev reload
sudo service udev restart

Έτσι η συσκευή έχει τα απαραίτητα δικαιώματα για να τρέξει στο λειτουργικό σύστημά μας. Τώρα θα πρέπει να ρυθμίσουμε την θύρα και στο πακέτο rplidar_ros, οπότε πλοηγούμαστε στο πακέτο στο περιβάλλον εργασίας μας:

cd ~/catkin_ws/src/rplidar_ros/launch
nano rplidar.launch

και στην οθόνη μας θα εμφανιστούν τα περιεχόμενά του:

```
<launch>
  <node  name="rplidarNode"          pkg="rplidar_ros"   type="rplidarNode" output="screen">
  <param name="serial_port"         type="string" value="/dev/ttyUSB0"/>
  <param name="serial_baudrate"     type="int"    value="115200"/>
  <param name="frame_id"            type="string" value="laser"/>
  <param name="inverted"            type="bool"   value="false"/>
  <param name="angle_compensate"    type="bool"   value="true"/>
  </node>
</launch>
```

Όπου στην παράμετρο serial_port θα πρέπει να βάλουμε την δική μας θύρα που είναι συνδεδεμένη η συσκευή, όπως είδαμε προηγουμένως.

Για να αποθηκεύσουμε, πατάμε "Ctrl+X" , έπειτα "y" για να αποδεχτούμε τις αλλαγές και μετά Enter.

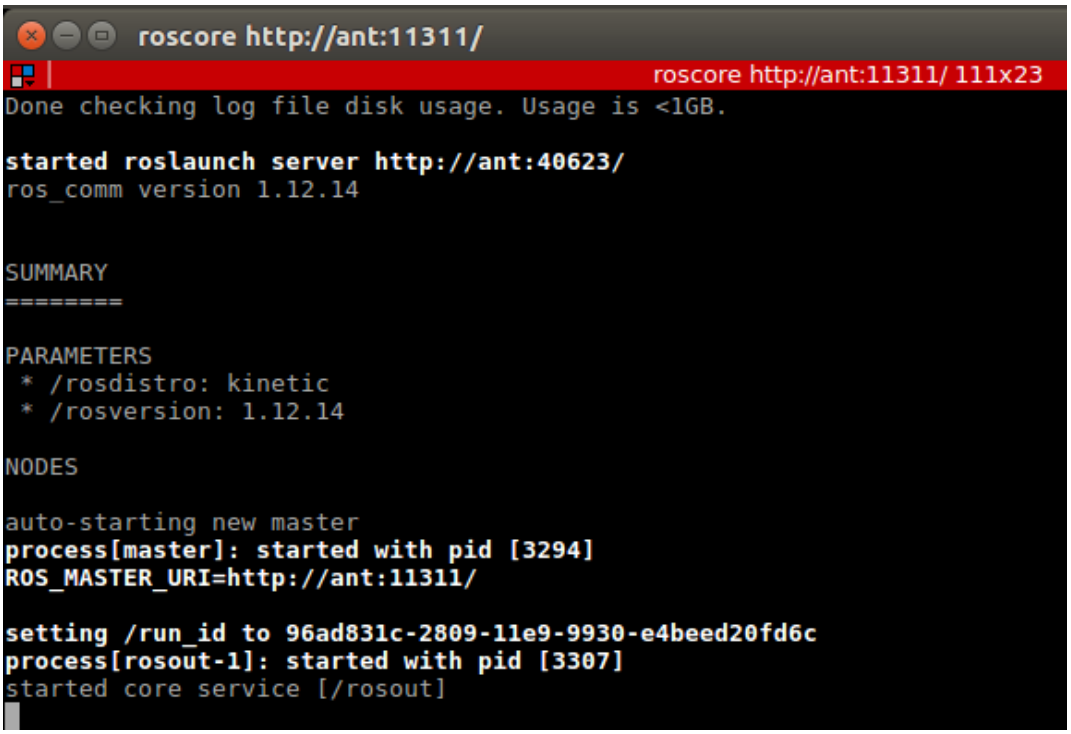
Το πακέτο **cv_camera** δεν χρειάζεται κάποια ρύθμιση στην περίπτωσή μας επειδή έχουμε μόνο μια κάμερα, και η συσκευή βρίσκεται πάντα στην θέση **dev/video0 .**

2.2.9 Εντολές σύνδεσης, λειτουργίας και αλληλεπίδρασης μεταξύ του οχήματος και του υπολογιστή ελέγχου.

Αρχικά πρέπει να ανοίξουμε το roscore στον υπολογιστή ελέγχου, μια συλλογή απο προγράμματα που χρειάζονται για να λειτουργήσει ένα σύστημα βασισμένο στο ROS για να μπορούν αυτά τα προγράμματα να επικοινωνήσουν μεταξύ τους. Ανοίγουμε ένα τερματικό στον υπολογιστή μας και πληκτρολογούμε την εντολή:

roscore

και θα δούμε τα παρακάτω μηνύματα αν άνοιξε σωστά.



Έπειτα θα πρέπει να ενεργοποιήσουμε τον αισθητήρα LiDAR στο Raspberry Pi για να ξεκινήσει την λειτουργία και την ανίχνευσή του.
Πρέπει αρχικά να συνδεθούμε με το Raspberry Pi.

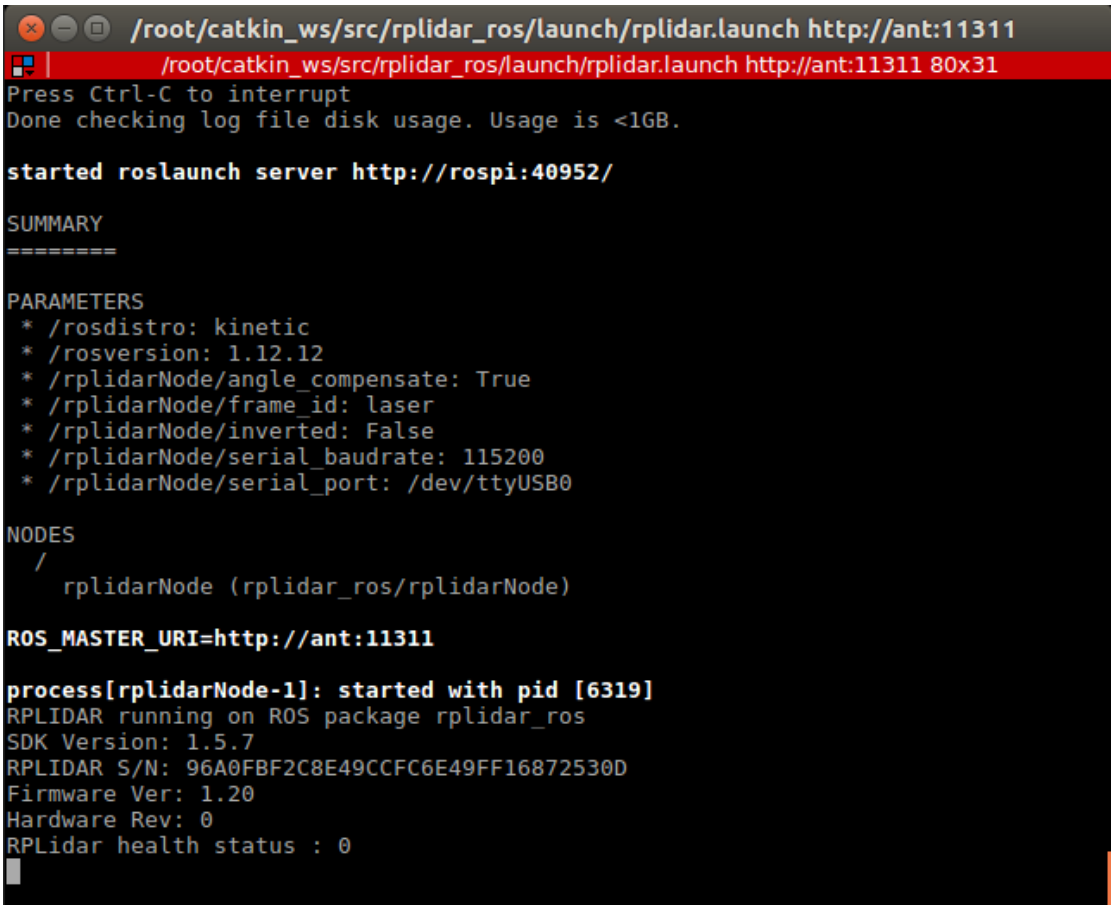
Ανοίγουμε ένα νέο τερματικό στον υπολογιστή και συνδεόμαστε στο Raspberry Pi, πληκτρολογώντας:

ssh username@hostname/IP

Όπου username το όνομα χρήστη του Ubuntu Mate στο Raspberry Pi, hostname ή IP την διεύθυνση του Raspberry Pi στο δίκτυο όπως την βάλαμε στο αρχείο **hosts** προηγουμένως. Θα μας ζητήσει τον κωδικό του χρήστη στο Raspberry Pi, τον πληκτρολογούμε σωστά και έχουμε πρόσβαση στο Raspberry Pi. Μόλις γίνει αυτό, για να βάλουμε τον αισθητήρα LiDAR σε λειτουργία, πληκτρολογούμε στο ίδιο τερματικό:

roslaunch rplidar_ros rplidar.launch

και εάν όλα λειτουργούν σωστά, ο αισθητήρας ανοίγει, βλέπουμε σαν απάντηση τα παρακάτω:

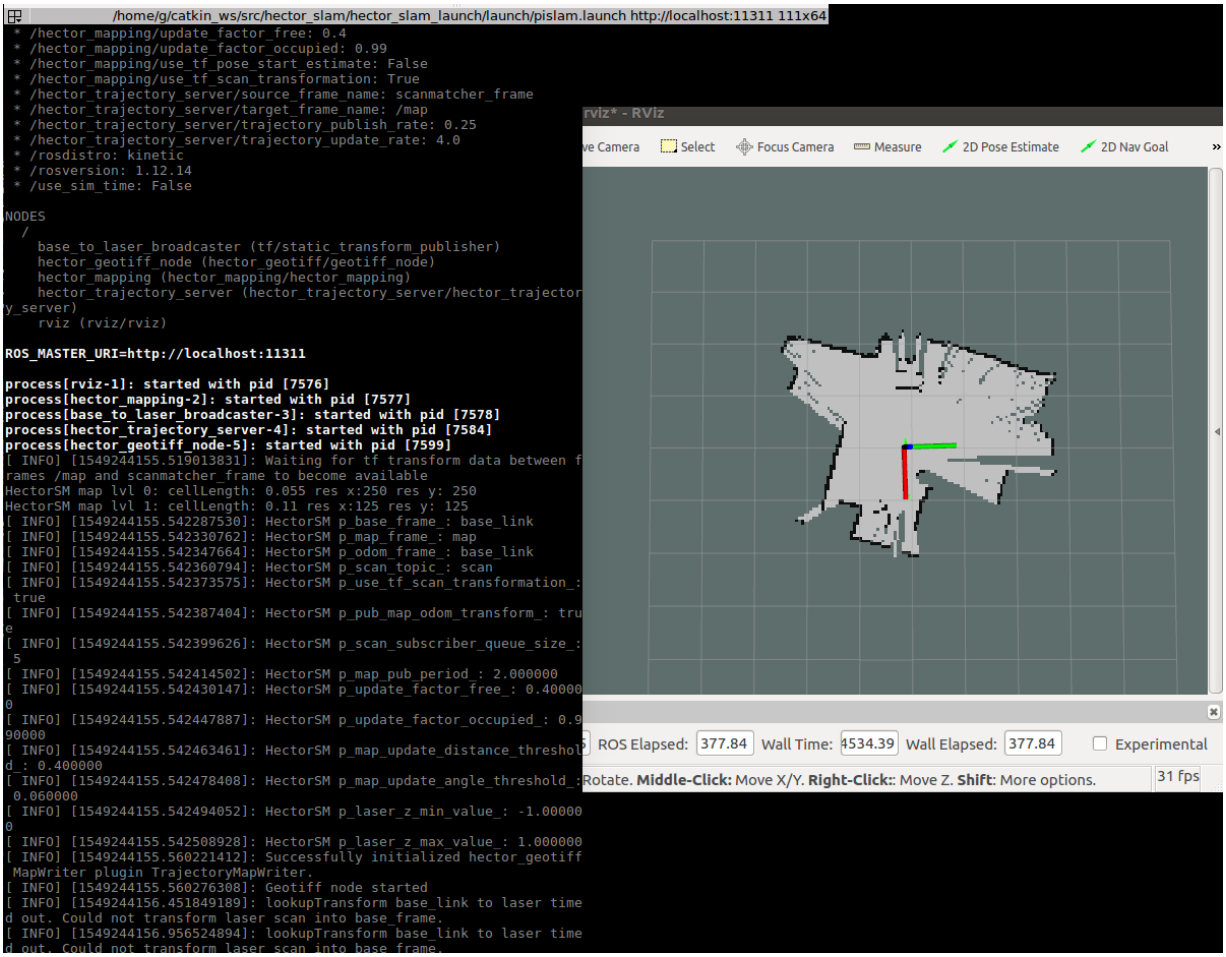


Συνεχίζοντας, θα πρέπει να ξεκινήσουμε να αξιοποιούμε τα δεδομένα του αισθητήρα στον υπολογιστή μας και πιο συγκεκριμμένα να τα δούμε αρχικά

στο Rviz. Για να το κάνουμε αυτό, ανοίγουμε ένα νέο τερματικό στον υπολογιστή και πληκτρολογούμε:

roslaunch hector_slam_launch pislam.launch

και εάν ο υπολογιστής λαμβάνει σωστά τα δεδομένα του αισθητήρα, θα δούμε αυτά τα μηνύματα, καθώς και το Rviz, που αρχίζει να εκμεταλλεύεται τα δεδομένα αυτά σχηματίζοντας ένα χάρτη.



Τώρα για να δημιουργήσουμε έναν χάρτη του περιβάλλοντος με ακρίβεια, θα πρέπει το όχημά μας να κινείται. Έτσι, θα ανοίξουμε ένα νέο τερματικό, και θα συνδεθούμε ξανά στο Raspberry Pi μέσω SSH με τον τρόπο που δείξαμε πριν.

Μόλις συνδεθούμε, θα πρέπει να ξεκινήσουμε το πρόγραμμα κίνησης του οχήματος που "ακούει" τις εντολές κίνησης του υπολογιστή. Πληκτρολογούμε λοιπόν:

rosrun pi_teleop move_pi.py

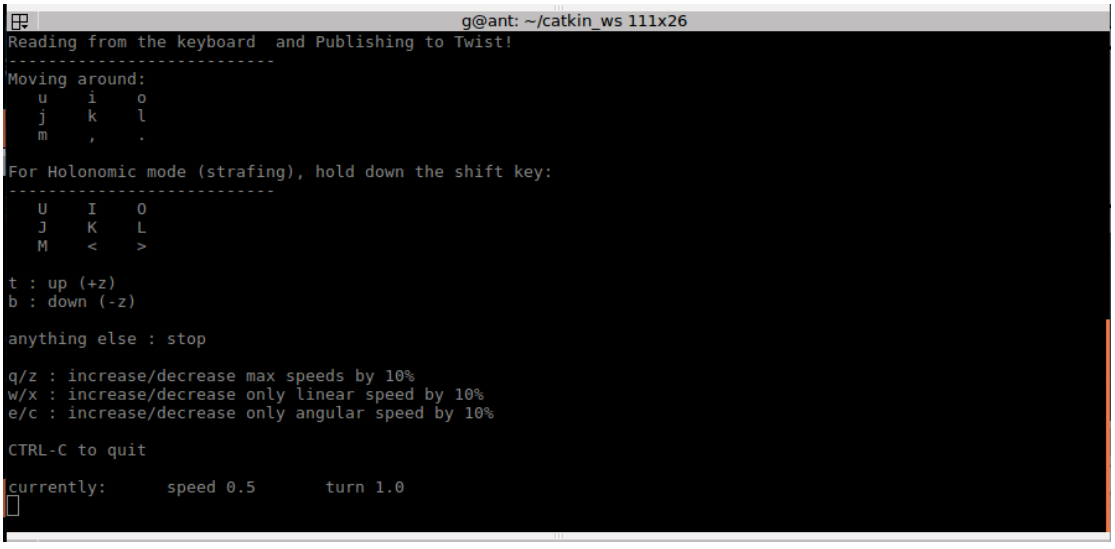
Και θα λάβουμε το μήνυμα οτι το όχημα περιμένει εντολές.

```
rospi@rospi:~/catkin_ws# ./teleop.sh
[INFO] [1549244134.285571]: Waiting for data...
```

Τώρα για να του δώσουμε εντολές, θα εκτελέσουμε στον υπολογιστή ελέγχου το πληκτρολόγιο κίνησης του ROS όπως αναλύσαμε προηγουμένως, και θα δίνουμε εντολές κίνησης απο εκεί. Ανοίγουμε ένα νέο τερματικό στον υπολογιστή και πληκτρολογούμε:

rosrun teleop_twist_keyboard teleop_twist_keyboard.py

και όταν ανοίξει θα μας δείξει την παρακάτω οθόνη βοηθώντας μας με τα πλήκτρα ελέγχου εξηγώντας πως πρέπει να κινηθούμε:



Σε αυτό το σημείο, πατώντας το πλήκτρο "i" θα πρέπει το όχημα να κινείται εμπρός, πατώντας την τελεία πίσω, πατώντας το πλήκτρο "j" αριστερά και το "L" δεξιά και με αυτό τον τρόπο μπορούμε να πλοηγηθούμε στον χώρο και να τον χαρτογραφήσουμε, με την οπτική βοήθεια της κάμερας φυσικά.

Για να δούμε τι "βλέπει" το όχημα απο την κάμερα, πρέπει πρώτα να ενεργοποιήσουμε την λειτουργία της στο όχημα. Θα ανοίξουμε ένα νέο τερματικό και θα συνδεθούμε ξανά στο Raspberry Pi μέσω SSH, και πληκτρολογούμε τις δύο παρακάτω εντολές:

rosparam set cv_camera/device_id 0
rosrun cv_camera cv_camera_node

Έτσι ανοίξαμε την κάμερα και την "δημοσιεύουμε" σε ένα node το οποίο θα βλέπουμε απο τον υπολογιστή. Για να δούμε τη ροή αυτή σε πραγματικό χρόνο στον υπολογιστή, εκτελούμε σε ένα νέο τερματικό την εντολή:

rosrun image_view image_view image:=/cv_camera/image_raw/
_image_transport:=compressed

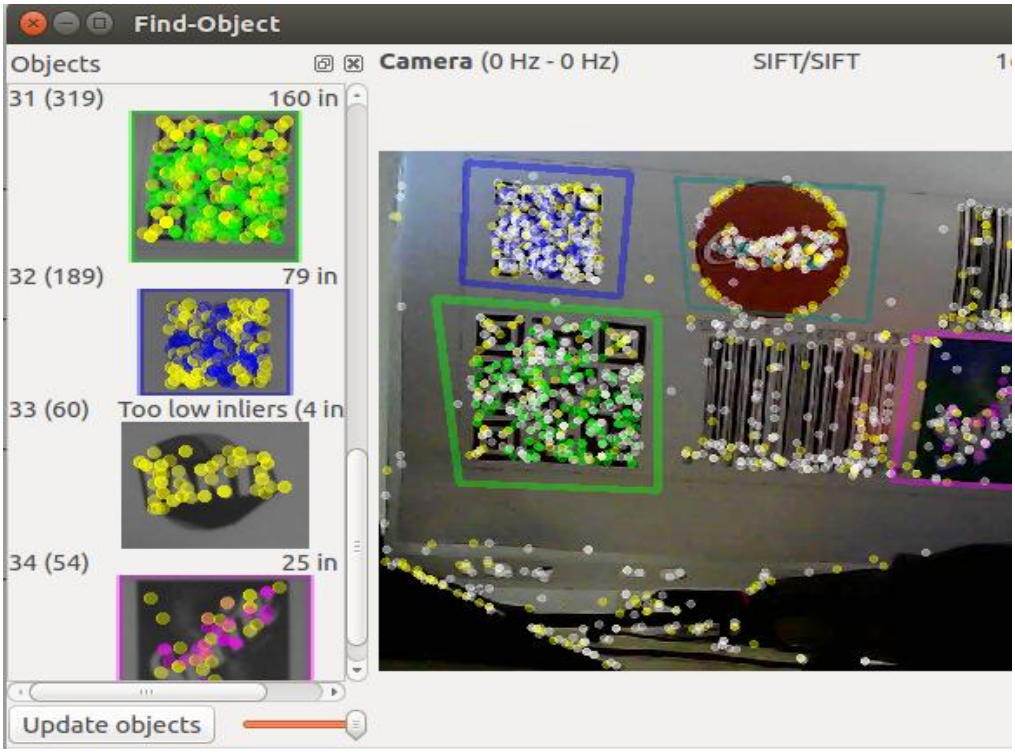
και θα ανοίξει ένα παράθυρο που θα βλέπουμε τα δεδομένα της κάμερας στον υπολογιστή μας. Αυτό μας βοηθά να πλοηγηθούμε στον χώρο καλύτερα, και μπορούμε να το αξιοποιήσουμε και σε συνδυασμό με το πρόγραμμα εύρεσης αντικειμένων **find object**. Για να ανοίξουμε το πρόγραμμα find object πληκτρολογούμε σε ένα νέο τερματικό στον υπολογιστή:

rosrun find_object_2d find_object_2d
image:=/cv_camera/image_raw

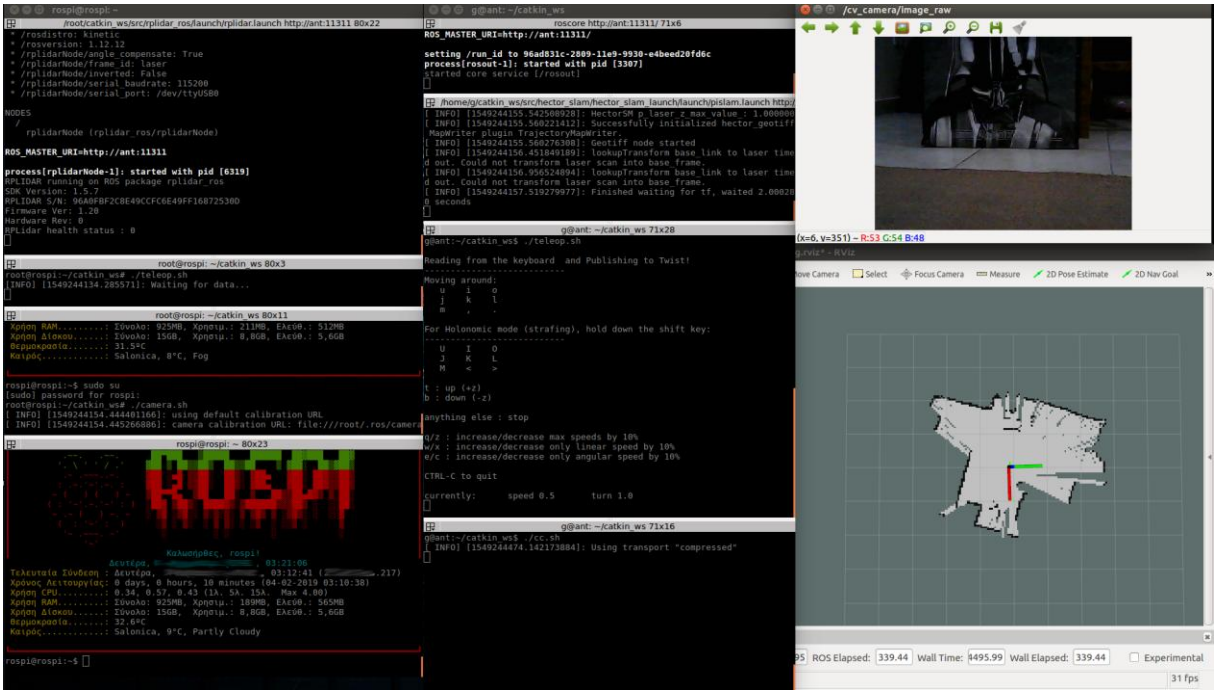
και θα ανοίξει το πρόγραμμα, το οποίο χρησιμοποιεί την κάμερά μας και μια βάση δεδομένων με εικόνες που εμείς φορτώσαμε σε αυτήν, για να ανιχνεύσει αντικείμενα στον χώρο.

Τα αντικείμενα ανιχνεύονται χρησιμοποιώντας γνωστές τεχνολογίες ανίχνευσης **SURF και SIFT** με descriptors (περιγραφείς) που δεν θα αναλύσουμε. Όταν ένα αντικείμενο στην βάση δεδομένων, έχει παρόμοιο αριθμό descriptors με το αντικείμενο απο την ροή της κάμερας, τότε αυτό

θεωρείται οτι βρέθηκε και το πρόγραμμα το χρωματίζει με περίγραμμα, όπως βλέπουμε και στην εικόνα.



Στο τέλος, ένα παράδειγμα του τι πρέπει να βλέπουμε στην οθόνη του υπολογιστή μας:

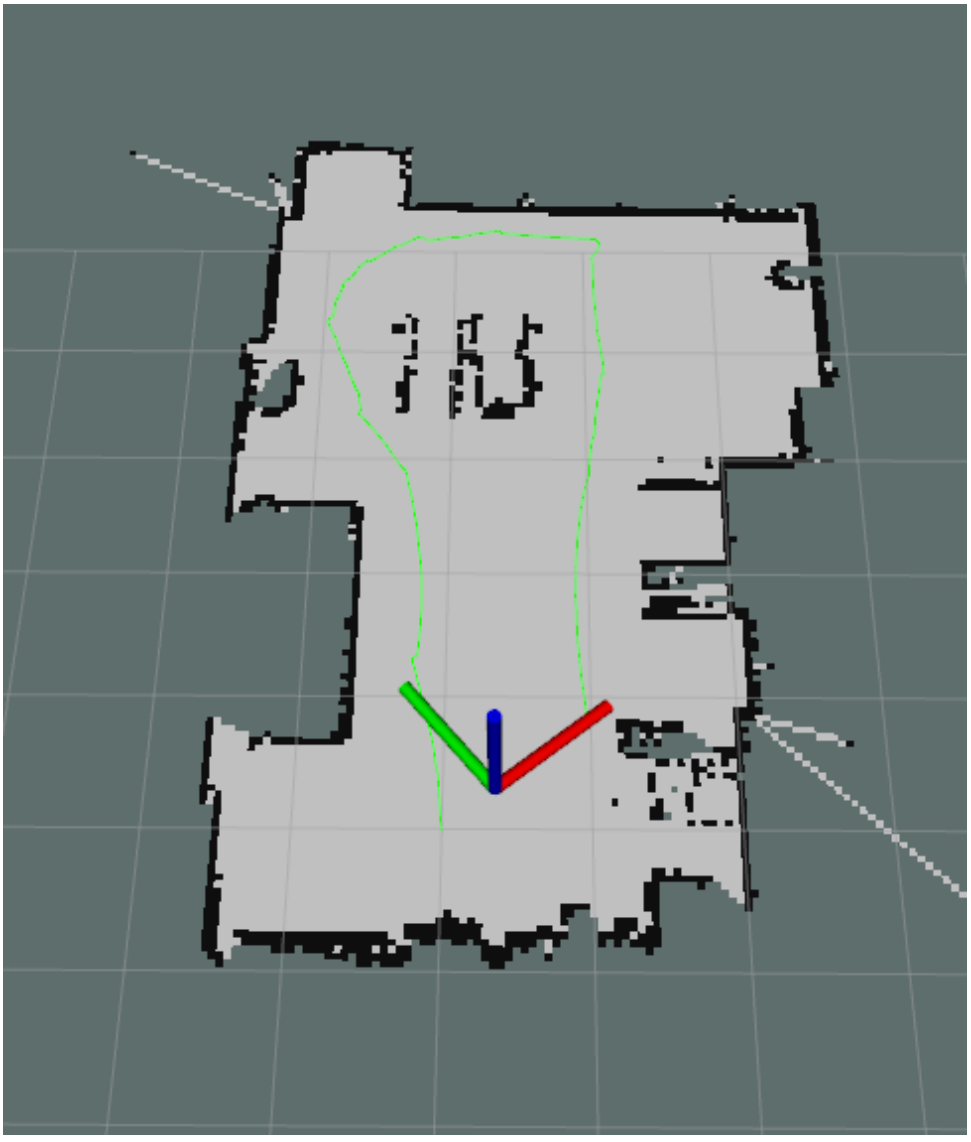


Μόλις τελειώσουμε με την χαρτογράφησή μας, θα πρέπει να εξάγουμε το αρχείο με τα δεδομένα του χάρτη που μόλις φτιάξαμε για να τα χρησιμοποιήσουμε αργότερα. Όταν τελειώσουμε την χαρτογράφηση και δίχως να κλείσουμε κάτι, ανοίγουμε ένα τερματικό και πληκτρολογούμε:

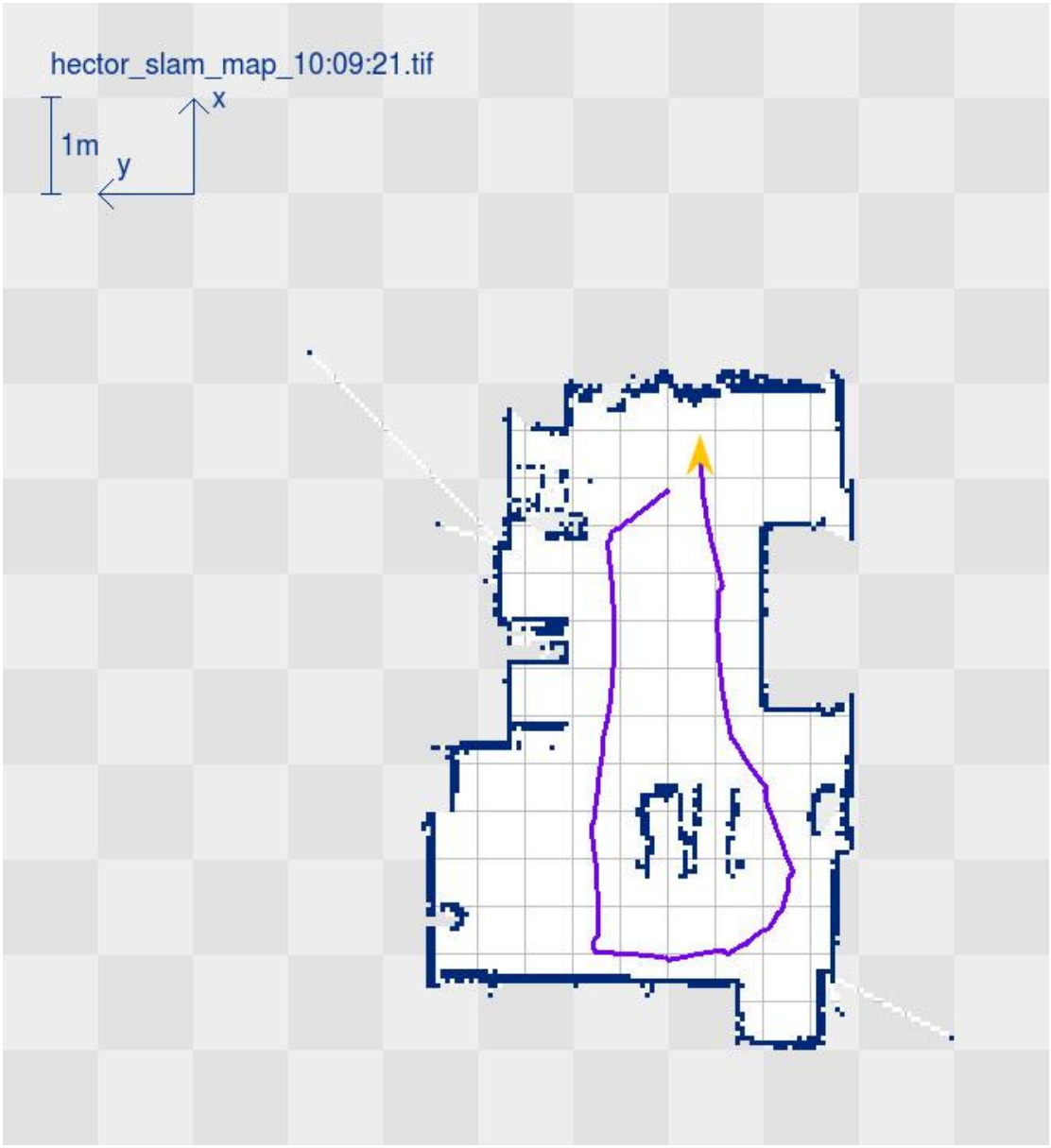
rostopic pub syscommand std_msgs/String "savegeotiff"

Από προεπιλογή, οι χάρτες geotiff αποθηκεύονται στον φάκελο 'hector_slam / hector_geotiff / maps'. Θα βρείτε ένα αρχείο .tif που περιέχει τα δεδομένα εικόνας και ένα αρχείο .tfw που περιέχει τις πληροφορίες γεωαναφοράς εκεί.

Παράδειγμα απεικόνισης χαρτογράφησης ενός χώρου απο το rviz:



Παράγωγο αποθήκευσης χαρτογράφησης του ίδιου χώρου:

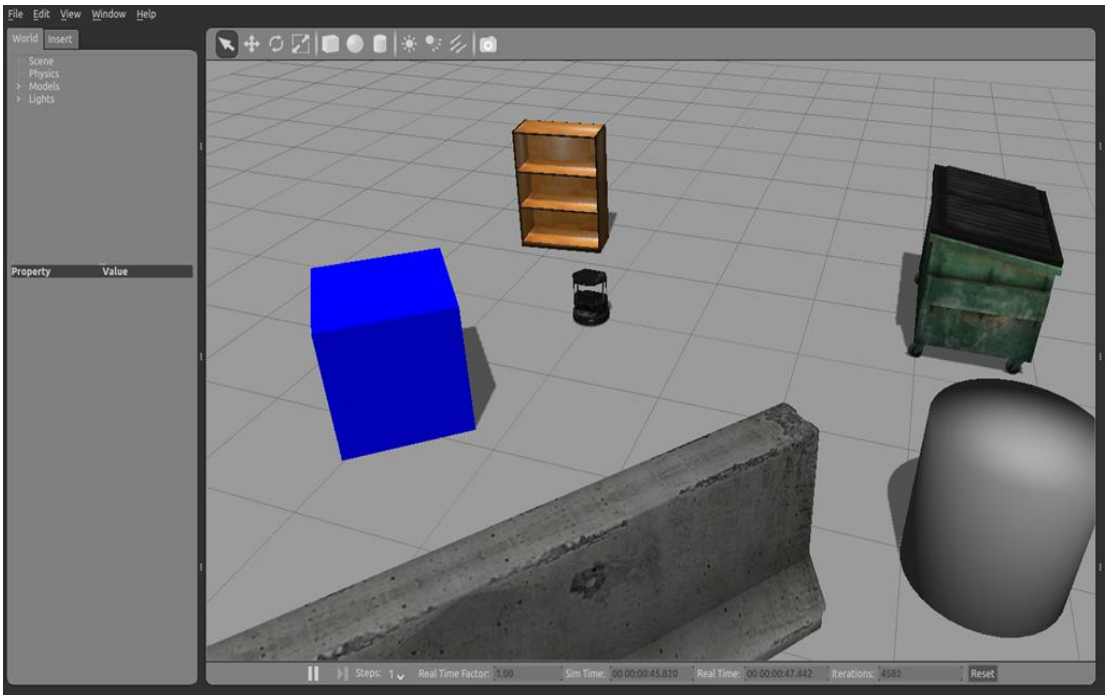


Έτσι μπορούμε να εκμεταλλευτούμε αυτόν τον χάρτη αργότερα για την αναβάθμιση της πλοήγησης ενός οχήματος για διάφορες διεργασίες και μεταφορές στο περιβάλλον αυτό, να αυξήσουμε την παραγωγικότητα της εργασίας, την απόδοση και την αποτελεσματικότητα, και να μειώσουμε τους χρόνους παράδοσης.

2.2.10 Λογισμικό Προσομοίωσης

Στον επιστημονικό χώρο ανάπτυξης ρομποτικών εφαρμογών οι δοκιμές και τα πειράματα είναι από τα πιο σημαντικά κομμάτια. Μπορεί θεωρητικά η εφαρμογή να δουλεύει χωρίς σφάλματα και στην πράξη να βγαίνουν προβλήματα που δεν υπήρχε πρόβλεψη για αυτά, καθώς στην προσομοίωση οι περισσότερες μεταβλητές είναι ιδανικές. Πλέον, οι δοκιμές σε κάθε στάδιο ανάπτυξης γίνονται μέσα από προσομοιωτές με σκοπό να να αποφευχθούν σφάλματα που μπορεί να επιφέρουν βλάβη στο μηχάνημα.

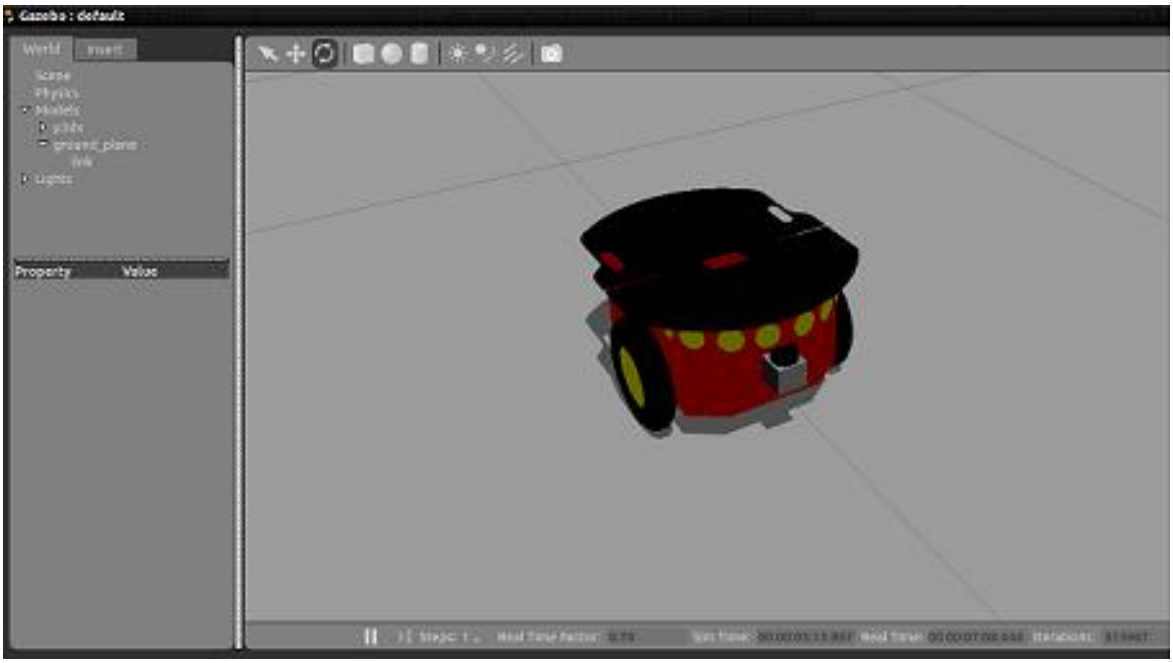
Το ROS επίσης προσφέρει κάποια εργαλεία προσομοίωσης αλλά και απεικόνισης, όπως το Gazebo και το Rviz αντίστοιχα . Η προσομοίωση ρομπότ είναι ένα βασικό εργαλείο στην εργαλειοθήκη του ROS. Ένας καλά σχεδιασμένος προσομοιωτής καθιστά δυνατή την ταχεία δοκιμή αλγορίθμων, σχεδιασμό ρομπότ, εκτέλεση δοκιμών παλινδρόμησης, και εκπαίδευση συστήματος τεχνητής νοημοσύνης χρησιμοποιώντας κοντά στην πραγματικότητα ή ρεαλιστικά σενάρια.



Το περιβάλλον προσομοίωσης Gazebo.

Το Gazebo προσφέρει τη δυνατότητα προσομοίωσης με ακρίβεια και αποτελεσματικότητα των ρομπότ σε πολύπλοκα εσωτερικά και εξωτερικά περιβάλλοντα με βολικές προγραμματιστικές και γραφικές διεπαφές.
Αποτελεί έναν τρισδιάστατο προσομοιωτή (3D Dynamic Simulator) σε λειτουργικό Linux με την υποστήριξη του ROS. Επειδή το Gazebo είναι δωρεάν λογισμικό ανοιχτού κώδικα, με σύνδεση και συνεργασία με το ROS, προσφέροντας 3D μοντέλα για ελεύθερη χρήση, έχει γίνει πλέον ένα από τα πιο διαδεδομένα και χρησιμοποιούμενα λογισμικά προσομοίωσης που υπάρχουν.

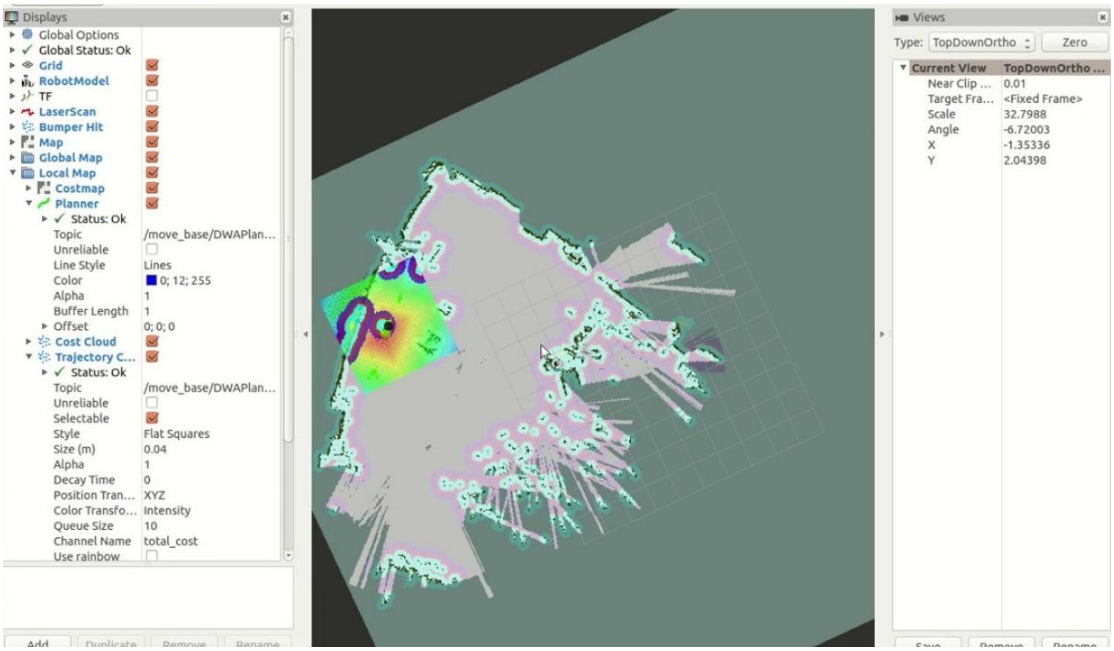
Το πλεονέκτημα του να μπορεί ένας χρήστης να δημιουργήσει έναν ολόκληρο εικονικό κόσμο, ο οποίος μπορεί και μιμείται με μεγάλη ακρίβεια και αποτελεσματικότητα τις πραγματικές συνθήκες της εκάστοτε εφαρμογής, ανάλογα και με τον προγραμματισμό του χρήστη, κι έτσι παρέχει ένα σημαντικό βοήθημα στην ανάπτυξη της εφαρμογής του.Υποστηρίζει πληθώρα διαφορετικών ιδιοτήτων φυσικής αναπαράστασης, μεγάλης ποικιλίας βιβλιοθήκών ανοιχτού κώδικα, μοντέλων ρομπότ, αισθητήρων, και παρέχει στον χρήστη ένα πάρα πολύ φιλικό περιβάλλον εργασίας (interface).



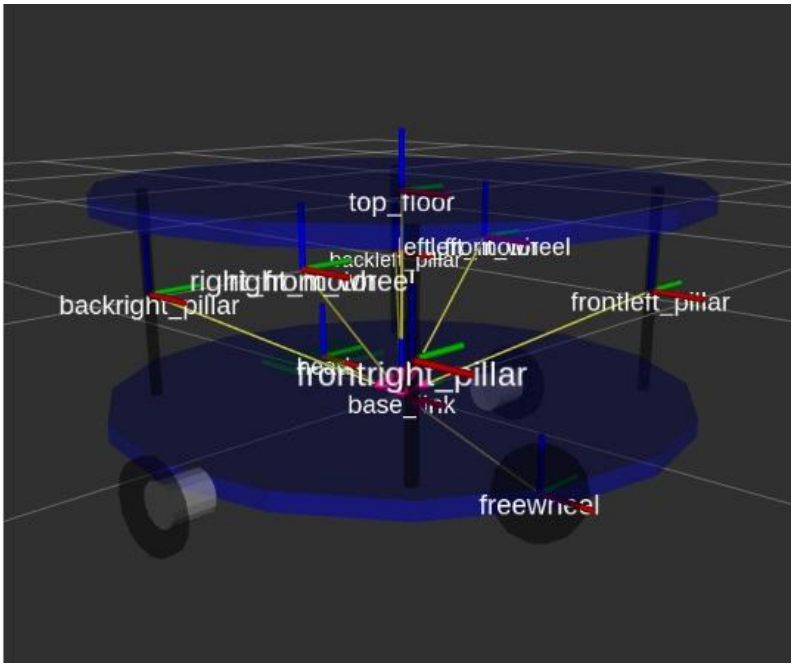
Το μοντέλο του AGV στο Gazebo

Το Rviz είναι ένα εργαλείο απεικόνισης ρομπότ. Παρέχει ένα γραφικό περιβάλλον για την απεικόνιση των δεδομένων των αισθητήρων, των

μοντέλων ρομπότ, των χαρτών περιβάλλοντος, όσο και τροχιών κίνησης του ρομπότ και μετασχηματισμών μεταξύ συστημάτων συντεταγμένων, το οποίο είναι ιδιαίτερα χρήσιμο για την ανάπτυξη και την αποσφαλμάτωση των ελεγκτών ρομπότ. Έτσι είναι δυνατή η ανά πάσα στιγμή γραφική απεικόνιση της κατάστασης του συστήματος, η παρακολούθηση της κίνησης ενός ρομπότ και των δεδομένων που παρατηρούνται από τους αισθητήρες του.



Χαρτογράφηση χώρου στο Rviz

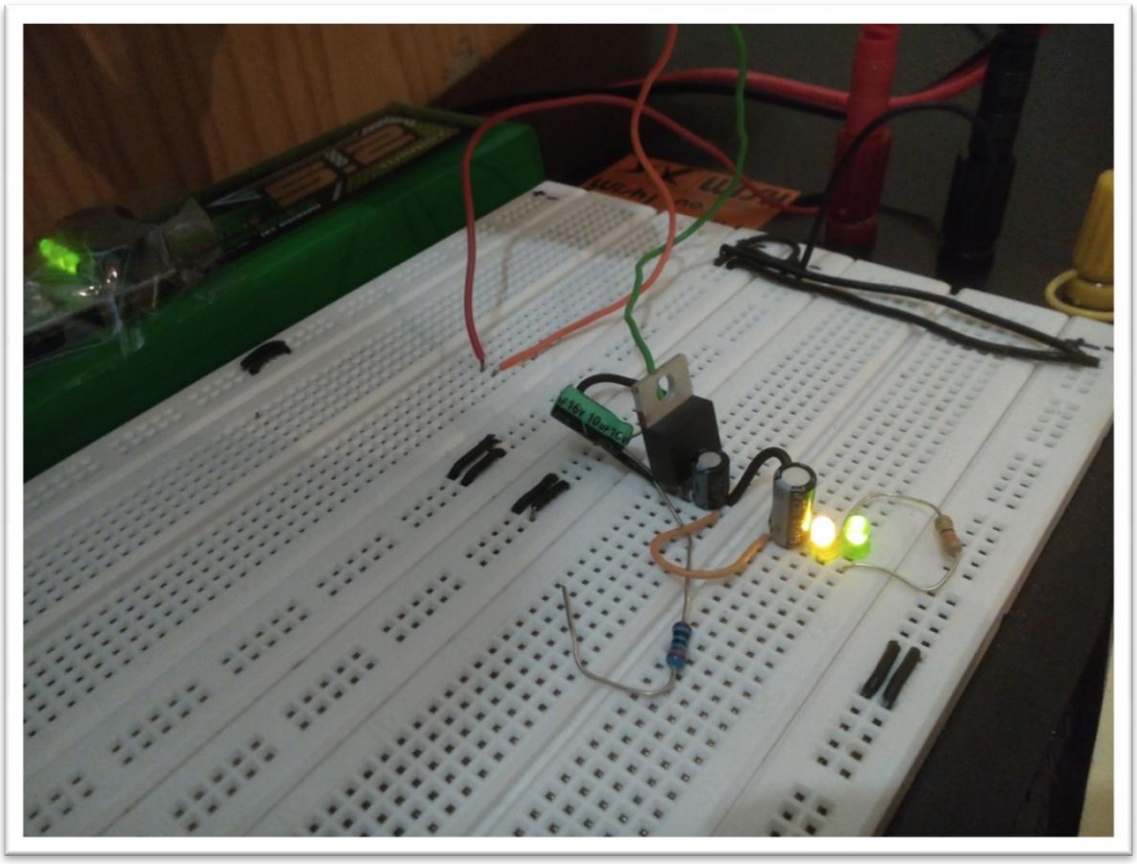


Το μοντέλο του AGV στο Rviz

2.3.Ανάλυση Δοκιμών και προσπάθειες

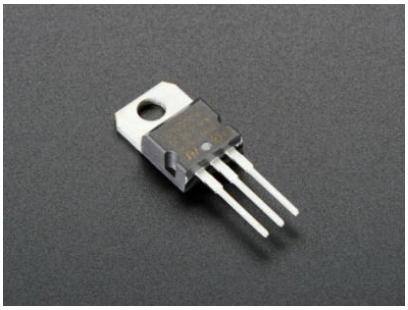
2.3.1 Ελλιπής τροφοδοσία του Raspberry Pi

Η ελλιπής τροφοδοσία του Raspberry Pi 2, ήταν ένα από τα πρώτα προβλήματα που αντιμετωπίσαμε κατά την ανάπτυξη του εγχειρήματος, στον τομέα των ηλεκτρονικών .



Εικόνα 2.3. Δοκιμή διάταξης γραμμικού ρυθμιστή τάσης

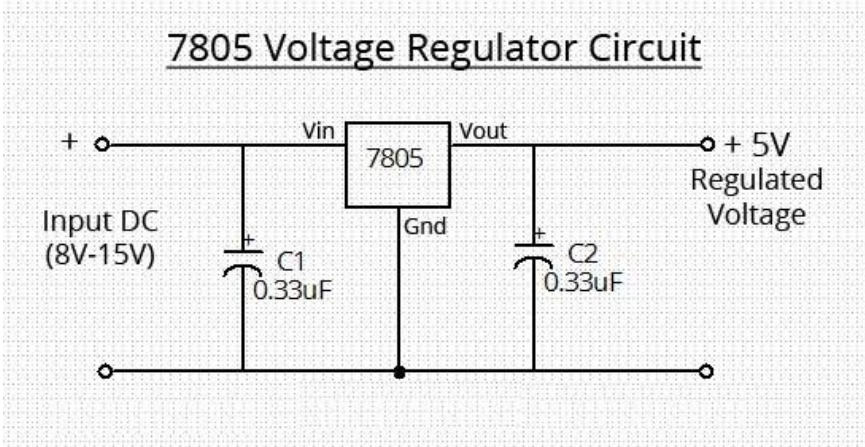
Ο επεξεργαστής έχει usb υποδοχή για την τροφοδοσία του.
Ένας κοινός φορτιστής κινητού τηλεφώνου (5Volt DC , 2 Ampere output) μπορεί να χρησιμοποιηθεί για την επίτευξη του στόχου, όμως για να καταστήσουμε τον επεξεργαστή και άρα το ρομπότ ανεξάρτητο από το δίκτυο παροχής ενέργειας, η τροφοδοσία του έπρεπε να γίνεται από μπαταρία. Συνεπώς, έπρεπε να δημιουργηθεί μία ηλεκτρονική διάταξη, ικανή να τροφοδοτεί τον επεξεργαστή με την κατάλληλη και



Linear regulator 7805

σταθερή τάση (5V) που χρειάζεται για να λειτουργήσει.
Αρχικά επιλέχθηκε να δοκιμασθεί ο Linear regulator 7805 ο οποίος απαιτεί τάση εισόδου μεταξύ 8-15 Vdc και βγάζει στην έξοδο του σταθερή τάση 5V DC-0.8 Ampere,καλύπτοντας έτσι εκ πρώτης όψεως, τις ανάγκες τροφοδοσίας που χρειαζόμασταν.

Η υπόλοιπη διάταξη που απαιτείται για να δουλέψει σωστά ο γραμμικός ρυθμιστής είναι η παρακάτω:



Διάταξη voltage regulator 7805

Μετά την ολοκλήρωση της συγκεκριμένης διάταξης όπως φαίνεται στην εικόνα 2.3. παραπάνω, καθώς και την εφαρμογή ψύκτρας, δοκιμάσαμε την λειτουργία του επεξεργαστή , όταν όμως μπήκαμε στην διαδικασία να κάνουμε δοκιμή καταπόνησης (stress test) στον επεξεργαστή ,
στην οθόνη ο Raspberry μας ενημέρωσε ότι τροφοδοτείται με λίγο ρεύμα .
Κατόπιν αρκετών δοκιμών διαπιστώθηκε ότι ο linear regulator δεν έχει την δυνατότητα να τροφοδοτήσει με αρκετό **ρεύμα** τον επεξεργαστή, και γι αυτό είχαμε το παραπάνω φαινόμενο. Μειονεκτήματα λοιπόν του 7805

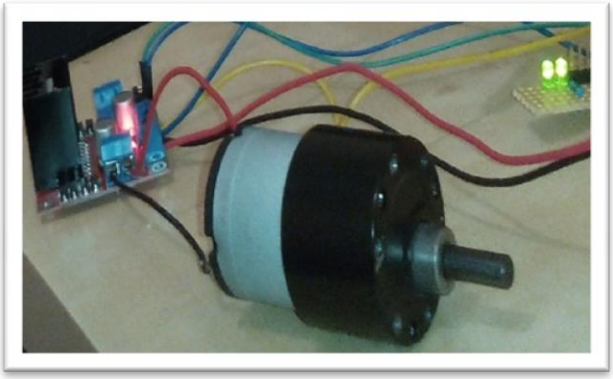
μετατροπέας dc-dc stepdown switching regulator

είναι, χαμηλή δυνατότητα παροχής ρεύματος (0.8 Amp max) ,πτώση τάσης και μετατροπή της υψηλής σε χαμηλή τάση, μέσω απαγωγής θερμότητας ,άρα και σπατάλη ενέργειας .
Η επόμενη και αποτελεσματική λύση ήταν η χρήση του μετατροπέα DC-DC (stepdown) switching regulator .

Ο συγκεκριμένος μετατροπέας παρήχε τις δυνατότητες που χρειαζόταν το σύστημα για να λειτουργήσει σωστά.

Αξιοσημείωτο είναι να αναφερθεί ότι, ακόμα και με τον μετατροπέα DC-DC, αντιμετωπίσαμε προβλήματα τροφοδοσίας ρεύματος όταν το σύστημα υπερφορτωνόταν από χρήση (χρήση του επεξεργαστή με βαριές ρουτίνες αλλά και ανάγκες ρεύματος άλλων επιμέρους μερών ή αισθητήρων) . Ο λόγος αυτήν την φορά ήταν τα "αδύναμα" καλώδια.
Μπορεί να μιλάμε για μικρά ρεύματα και τάσεις, αλλά πολλές φορές καλώδια αγνώστων προδιαγραφών και ποιότητας, μπορούν να επιφέρουν προβλήματα .

2.3.2. Απόλυτος έλεγχος κινητήρων

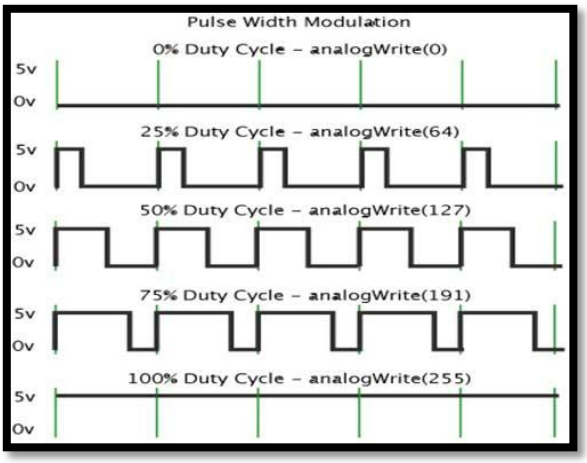


rotary encoder κινητήρας

Ένα άλλο πρόβλημα που αντιμετωπίσαμε ήταν η αδυναμία απόλυτου ελέγχου των κινητήρων. (ταχύτητα, ακρίβεια, αλλαγή στροφών)

Οι κινητήρες που επιλέχθηκαν δεν είναι ακριβείας παρόλο που έχουν εγκατεστημένους rotary encoders για την δυνατότητα μέτρησης πλήθους στροφών, λειτουργούν σαν απλοί DC κινητήρες και ως εκ τούτου χωρίς κάποιο προγραμματιστικό έλεγχο, δεν μπορεί να επιτευχθεί ακριβής έλεγχος.

Δίνοντας δηλαδή την ονομαστική τάση στους κινητήρες, αυτοί αρχίζουν να περιστρέφονται με την προκαθορισμένη τους ταχύτητα αλλά και με συγκεκριμένη δεξιόστροφη ή αριστερόστροφη περιστροφή. Για αλλαγή περιστροφής, θα πρέπει να αλλαχθεί η πολικότητα των καλωδίω.



pwm outputs

Η λύση του προβλήματος ήρθε μέσω του ελέγχου των κινητήρων από τον Raspberry Pi, με PWM (pulse width modulation), μέσω ενός Motor Driver.

Η δυνατότητα διαμόρφωσης πλάτους παλμού (pwm) μέσω προγράμματος, προσφέρει απόλυτο έλεγχο επί των κινητήρων ως προς την ακρίβεια, την περιστροφή αλλά και την ταχύτητα.

Έτσι το ρομπότ κινείται στον χώρο με την επιθυμητή ταχύτητα πλοήγησης ,(δυνατότητα αυξομείωσης ταχύτητας), μπορεί να στρίψει (δυνατότητα ελέγχου στροφών του ενός ή του άλλου κινητήρα), να περιστραφεί γύρω από τον εαυτό του, αλλά και να οπισθοχωρήσει (δυνατότητα τροφοδοσίας αρνητικού ρεύματος).

Σημειωτέο ότι, 2 κινητήρες ιδίων χαρακτηριστικών εργοστασιακά, δεν έχουν τις ίδιες στροφές ανά λεπτό (rpm) , και αυτό για ποικίλους λόγους.

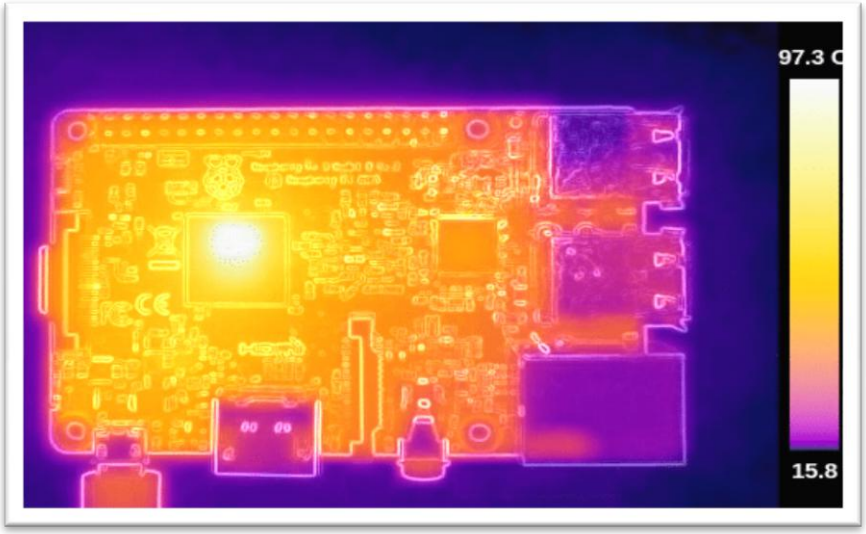
Γενικώς επειδή πρόκειται για μηχανικά μέρη, ποτέ τίποτα δεν είναι ίδιο με κάτι άλλο, άσχετα αν υπάρχει αυτή η ψευδαίσθηση.
Πρώτον, τα τυλίγματα τους , που μπορεί να φτάνουν και αρκετά μέτρα , δεν είναι πάντα ίδια σε μήκος , αλλά και τα ίδια τα υλικά, ακόμα και εργοστασιακά φτιαγμένα, δεν έχουν τις απόλυτες ίδιες προδιαγραφές. Για παράδειγμα, σφίγγοντας κάποια βίδα λίγο περισσότερο , μπορεί να αυξηθεί η τριβή των μηχανικών μερών μεταξύ των, και ως εκ τούτου η ελάττωση των στροφών. Επίσης υπάρχει πτώση των στροφών σε κάποιον κινητήρα, αν η ονομαστική τάση είναι 12V για παράδειγμα, και εμείς τον τροφοδοτήσουμε με 11.8V. κ.α. Είναι λοιπόν αναπόφευκτος ο έλεγχος των στροφών των κινητήρων για αυτήν την εφαρμογή.

Ένα άλλο συμπληρωματικό πρόβλημα που προέκυψε αυτήν την φορά στους τροχούς, είναι η ολίσθηση των τροχών επί του εδάφους . Αναλόγως σε ποια επιφάνεια, υπό ποιες συνθήκες (βρεγμένο δάπεδο - ολισθηρό) ή απότομο φρενάρισμα, προκαλείται ολίσθηση των τροχών. Έγινε αλλαγή λοιπόν, σε πιο πλατύς και με λαστιχένια επένδυση για την αποφυγή του φαινομένου.

λάστιχα με οδόντωση στους τροχούς

2.3.3. Υπερθέρμανση επεξεργαστή

Ένα επιπλέον πρόβλημα, εύκολα όμως αντιμετωπίσιμο ήταν η υπερθέρμανση του επεξεργαστή Raspberry Pi 2 υπό φορτίου.



θερμική απεικόνιση raspberry σε λειτουργία overclock

Ο Raspberry είναι ένας δυνατός επεξεργαστής με ισχυρή επεξεργαστική ισχύ. Οι προδιαγραφές του είναι: 900MHz τετραπύρηνος επεξεργαστής ARM Cortex - A7.

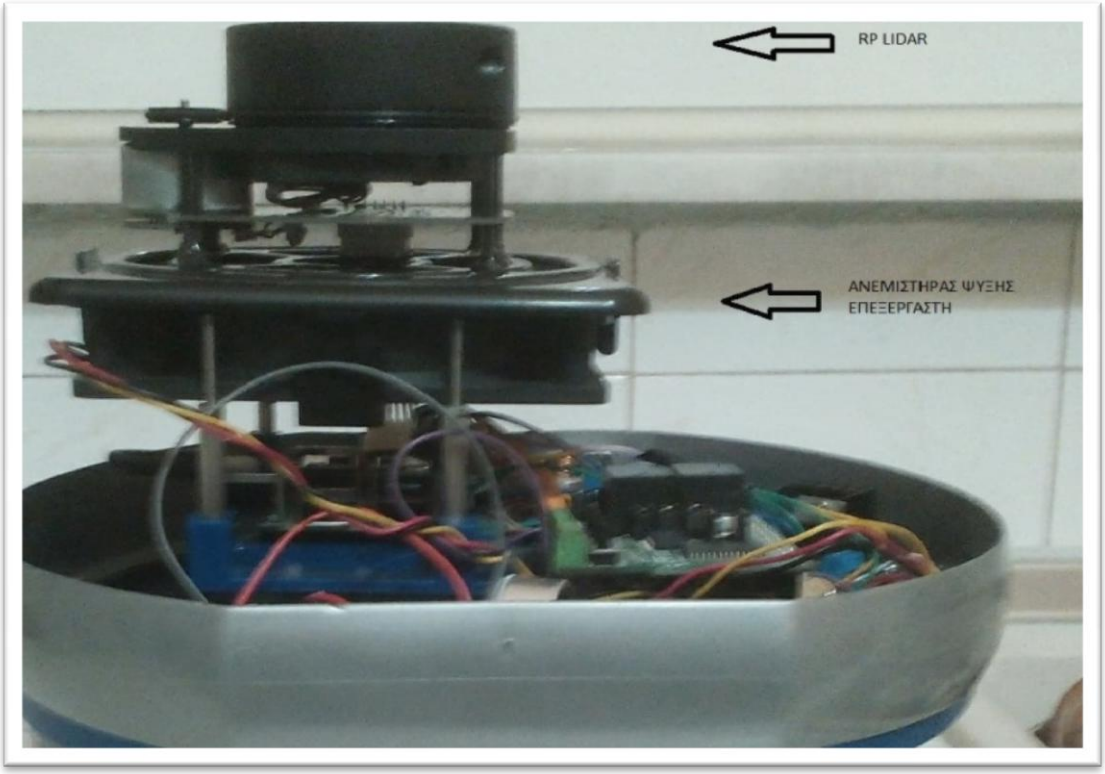
Έχει ακόμη και την δυνατότητα υπερχρονισμού σε 1Ghz και γενικώς μπορεί να ανταπεξέλθει σε υψηλές απαιτήσεις.

ψύκτρα επεξεργαστή

Όπως συμβαίνει όμως με όλους τους επεξεργαστές , οι απαιτήσεις επιφέρουν αύξηση της εσωτερικής θερμοκρασίας και άρα πρέπει να προστατευθεί από την υπερθέρμανση η οποία δεν επιφέρει μόνον άμεσα βλάβη όταν ξεπεραστεί το όριο ασφαλείας, αλλά και δυνητικά στον χρόνο , μειώνει την "ζωή" του επεξεργαστή και των κυκλωμάτων του όταν αυτός λειτουργεί συνεχώς σε υψηλή θερμοκρασία. Επίσης μετά από το κατώφλι ασφαλείας (threshold) των περίπου 80 °C, ο

επεξεργαστής λόγω της εσωτερικής ρουτίνας ασφαλείας που έχει, ρίχνει αυτόματα την επεξεργαστική του ισχύ, με αποτέλεσμα να γίνεται πιο αργός, ώστε να προστατευθεί από την υπερθέρμανση.

Για τον λόγο αυτό εγκαταστάθηκε μία ψύκτρα θερμότητας επάνω στον επεξεργαστή της πλακέτας Raspberry όπως επίσης, και ένας ανεμιστήρας ψύξης για τον ίδιο λόγο. Το αποτέλεσμα ήταν πλήρως ικανοποιητικό μιας και με αυτόν τον τρόπο , η γενική θερμοκρασία του επεξεργαστή έπεσε κατα μέσο όρο 20% .



ανεμιστήρας ψύξης

2.3.4. Το ρεύμα λειτουργίας που απαιτεί ο αισθητήρας LiDAR

Αρχικά ο αισθητήρας LiDAR επειδή τροφοδοτείται μέσω της διασύνδεσης USB δεν μπορούσε να λειτουργήσει. Μέσω έρευνας καταλήξαμε στο οτι δεν τροφοδοτείται σωστά, γιατί το συγκεκριμμένο μοντέλο Raspberry Pi 2 Model B+ έχει όριο 0.6 Ampere ρεύμα και στις 4 USB θύρες του, καθιστώντας μεγάλο πρόβλημα καθώς ο αισθητήρας χρειάζεται λίγο λιγότερο απο 1 Ampere ρεύμα για να λειτουργήσει σωστά, και επίσης η κάμερα και η ασύρματη σύνδεση WiFi τροφοδοτούνται επίσης μέσω USB.

Για να αλλάξει αυτό πρέπει να ρυθμιστεί το GPIO 38 του Raspberry Pi σε 1, το οποίο με τη σειρά του ενεργοποιεί ένα τρανζίστορ πεδίου (FET) που ελέγχει όριο ρεύματος στις θύρες USB, διπλασιάζοντας το σε 1.2 Ampere. Το τρανζίστορ πεδίου (FET), είναι μια ηλεκτρονική συσκευή που χρησιμοποιεί ένα ηλεκτρικό πεδίο για τον έλεγχο της ροής ρεύματος. Αυτό επιτυγχάνεται με την εφαρμογή τάσης στο τερματικό πύλης, η οποία με τη σειρά του μεταβάλλει την αγωγιμότητα μεταξύ των αγωγών εξόδων και των τερματικών πηγής. Η διαδικασία που πρέπει να ακολουθήσουμε είναι απλή.

Πρέπει να συνδέσουμε την κάρτα μνήμης του Raspberry Pi σε έναν υπολογιστή, και ανοίγοντας το αρχείο **/boot/config.txt** με έναν οποιοδήποτε κειμενογράφο, να προσθέσουμε την σειρά **max_usb_current=1** και να αποθηκεύσουμε.

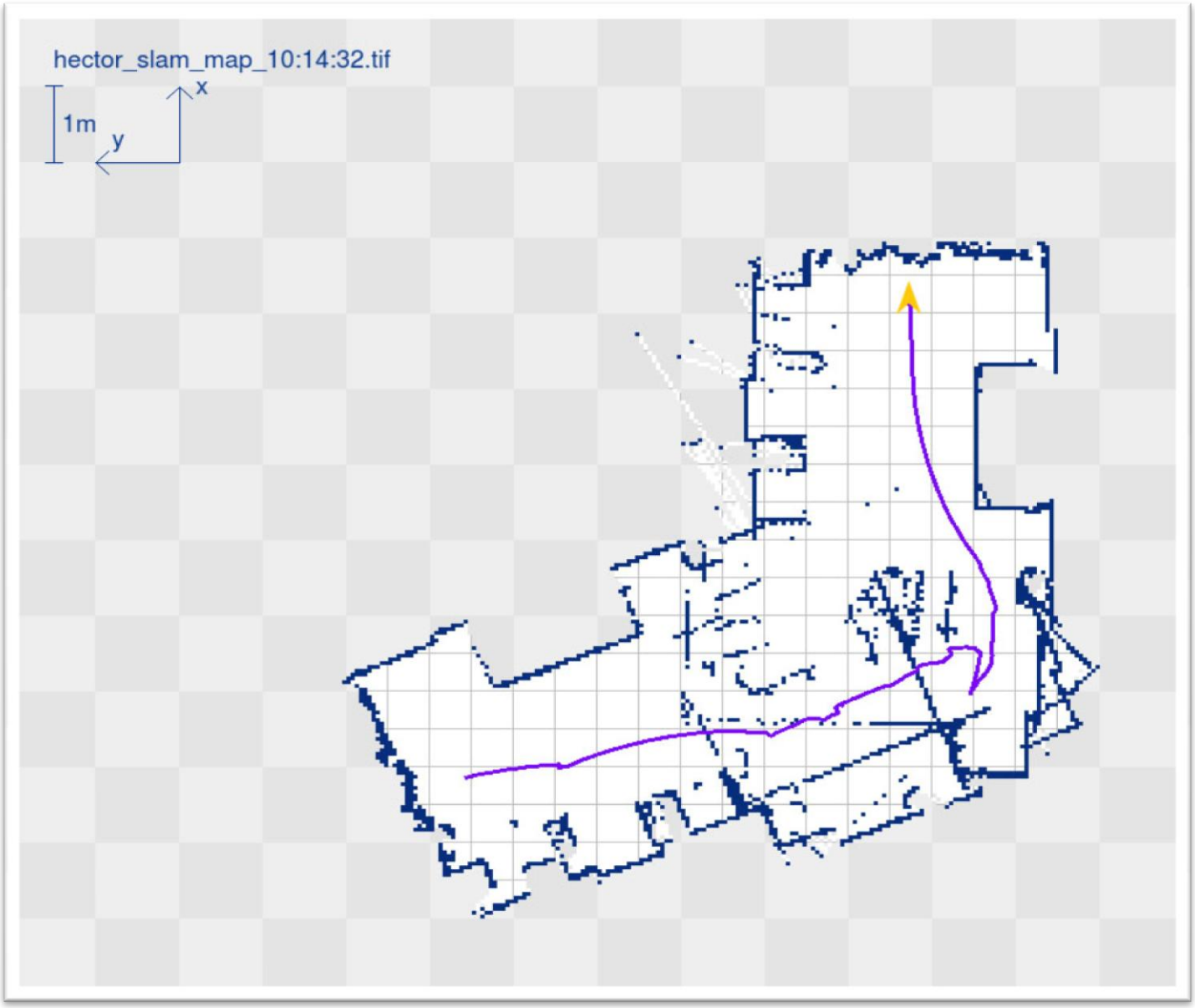
2.3.5. Πολλαπλοί σχεδιασμοί χαρτών λόγω παρεμβολών

Κατά τη διάρκεια της χαρτογράφησης, κάποιες φορές εμφανίζονταν πολλαπλοί χάρτες, κάνοντας το αποτέλεσμα να είναι λάθος. Αυτό συνέβαινε επειδή η μονάδα μέτρησης αδρανείας (IMU) που βρίσκεται ενσωματωμένη στον αισθητήρα LiDAR δεχόταν μαγνητική παρεμβολή απο το μαγνητικό πεδίο του κινητήρα του ανεμιστήρα, λόγω της τοποθέτησης του ανεμιστήρα χιλιοστά κάτω απο τον αισθητήρα, ως βάση του.Παρόλο που μπορεί να φαίνεται υπερβολικό το παράδειγμα που παρατίθεται ,τα αποτελέσματα άλλαξαν όταν ο ανεμιστήρας μετακινήθηκε , κάτι που δείχνει πως πάντοτε πρέπει να προσέχουμε τα πάντα που αφορούν ευαίσθητα όργανα και αισθητήρες , κυρίως σε θέματα που αφορούν παρεμβολές από 3 σώματα ή περιβάλλον .

Ο αισθητήρας μέτρησης αδρανείας (IMU) είναι μια ηλεκτρονική συσκευή που μετράει και αναφέρει την επιτάχυνση του σώματος, το γωνιακό ρυθμό και μερικές φορές το μαγνητικό πεδίο που περιβάλλει το σώμα, χρησιμοποιώντας ένα συνδυασμό επιταχυνσιόμετρων και γυροσκοπίων, μερικές φορές και μαγνητόμετρα.

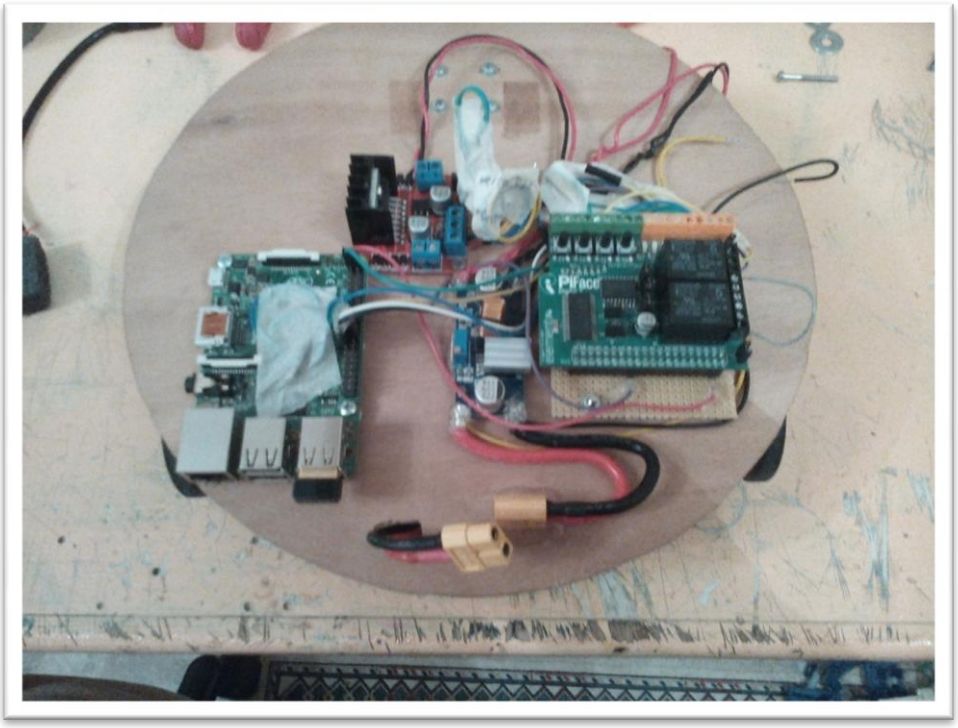
Αυτό προκαλούσε πρόβλημα γιατί ο αισθητήρας έδινε δεδομένα οτι το όχημα έχει στάση (pose) διαφορετική απο την πραγματικότητα, και έτσι ως παράγωγο υπήρχαν λάθος δεδομένα χάρτη.

Άλλο ένα παρόμοιο πρόβλημα, ήταν πως το όχημα κινείται και στρίβει πολύ γρήγορα για το ρυθμό ανανέωσης του αισθητήρα LiDAR και του αισθητήρα μέτρησης αδράνειας του, και είχε ως συνέπεια πάλι, κάποιες φορές, να δείχνει λάθος στάση στα δεδομένα. Αυτό το πρόβλημα λύθηκε κάνοντας το όχημα να κινείται πιο αργά μέσω ελέγχου στροφών όταν εκτελείται η διαδικασία χαρτογράφησης του χώρου .



πολλαπλοί σχεδιασμοί χαρτών λόγω παρεμβολών

2.3.6.Δημιουργία πρωτότυπου και αρχικά στάδια υλοποίησης



Το πρωτότυπο σώμα με τα βασικά μέρη

Το παρόν τελικό αποτέλεσμα προέκυψε μετά από την αρχική δημιουργία ενός πρωτότυπου ξύλινου στο σώμα ρομπότ .

Έχοντας στην κατοχή μας τα βασικά "κομμάτια" που θα αποτελούσαν το ρομπότ, έπρεπε να σχεδιασθεί και υλοποιηθεί ένα σώμα κατάλληλο για την φιλοξενία των μερών του και φυσικά να γίνουν δοκιμές στην χωροταξία των αντικειμένων .Επιλέχθηκε να δημιουργηθεί ένα κυκλικό ρομπότ για αρκετούς λόγους .

Πρωτίστως, το περιβάλλον εργασίας του ρομπότ θα είναι ένας χώρος μη ενιαίος, με διάφορα αντικείμενα εντός του, ξεχωριστούς χώρους και άλλα. Η κυκλική μορφή, διευκολύνει την ευκολότερη κίνηση μέσα στον χώρο, με δυνατότητα αποφυγής σύγκρουσης με άλλα αντικείμενα και τοίχους.

Δευτερευόντως, η δυνατότητα χαρτογράφησης από το ρομπότ, δίνει ένα "στίγμα" του ίδιου του ρομπότ στον χώρο και με την κυκλική του μορφή, εξαλείφεται το προφίλ του από τον χώρο.

Τέλος, η ικανότητα να περιστρέφεται γύρω από τον εαυτό του, αλλά και η δυνατότητα οπισθοπορείας, δίνει πάντοτε το ίδιο ακριβώς προφίλ του οχήματος στον χώρο .

Σχετικά με την χωροταξία, οι πλακέτες του επεξεργαστή, του Pi Face και άλλων μερών, έχουν αρκετές υποδοχές εισόδων και εξόδων, και για την βέλτιστη χρήση όπως και διευκόλυνση του χρήστη, αλλά και για την μέγιστη απόδοση παραδείγματος χάρην της ψύκτρας, έπρεπε να γίνει προσεκτική μελέτη και εγκατάσταση.

Επιπλέον, θα έπρεπε να προβλεφθεί και κενός χώρος προς χρήση για μελλοντικές επεκτάσεις, προσθήκες αισθητήρων και μηχανικών μερών.

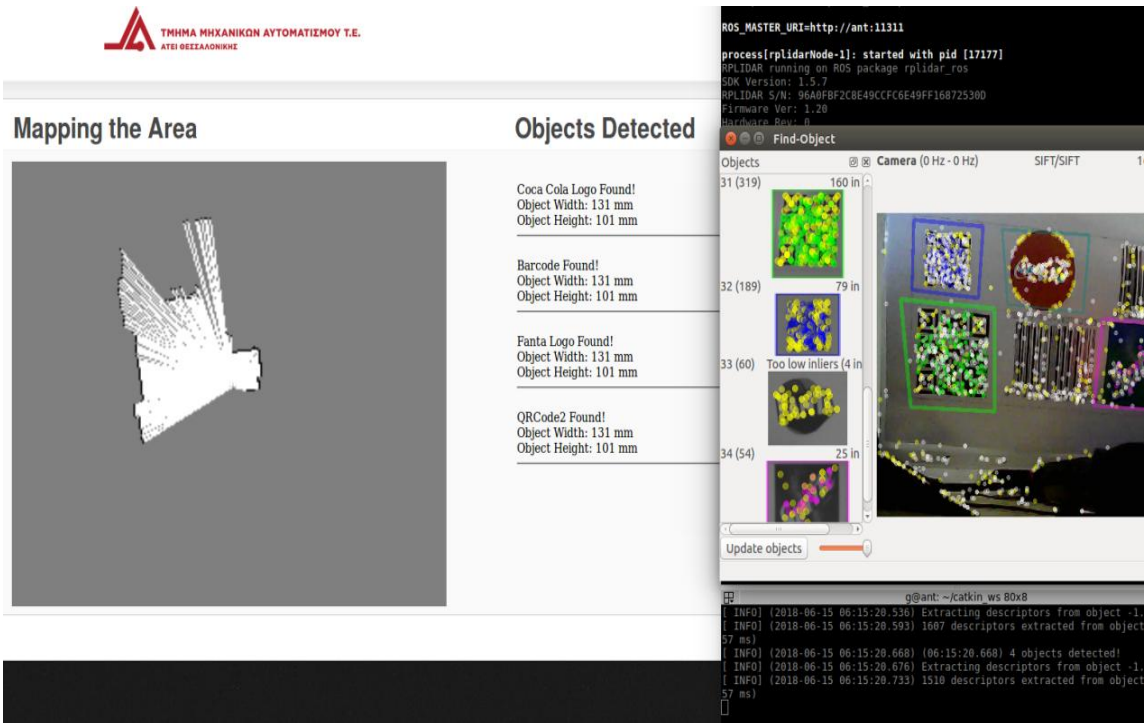
Το τελικό αποτέλεσμα που προέκυψε είναι ένα μετρίου σε μέγεθος ρομπότ, χωρίς να έχει παραφορτωθεί από μηχανικά μέρη, λειτουργικό και αξιόπιστο.

Το ρομπότ στα τελικά του στάδια

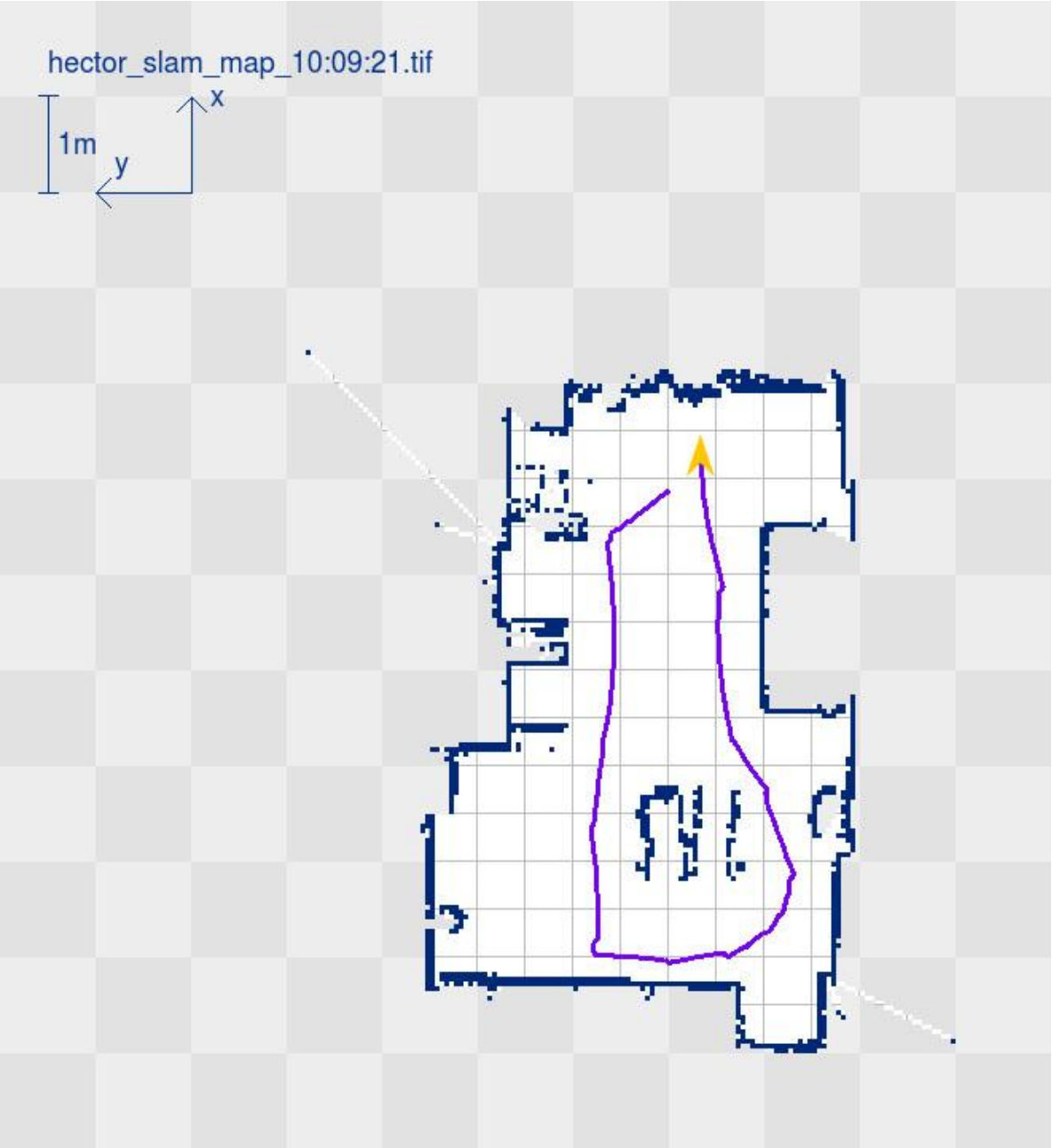
2.4. Λειτουργίες της παρούσας υλοποίησης

Η παρούσα υλοποίηση μπορεί να πλοηγείται στον χώρο, με σκοπό την χαρτογράφησή του, μέσω τηλεχειρισμού, απο τον υπολογιστή ελέγχου. Ο τηλεχειρισμός είναι εφικτός είτε με τα πλήκτρα WASD μέσω διασύνδεσης Secure Shell είτε μέσω του πληκτρολογίου Twist Teleop Keyboard του ROS, χρησιμοποιώντας την βιβλιοθήκη getch που λαμβάνει χαρακτήρες , καθώς και τις αντίστοιχες βιβλιοθήκες geometry_msgs και std_msgs του ROS.

Το πληκτρολόγιο αυτό στέλνει μηνύματα τύπου Twist του ROS τα οποία περιέχουν εντολές ταχύτητας, γωνιακής ταχύτητας και κατεύθυνσης, και το πρόγραμμά μας στο όχημα τα μεταφράζει στην αντίστοιχη κίνηση των κινητήρων. Για να είναι εφικτός και εύκολος ο τηλεχειρισμός χρησιμοποιούνται δυο εργαλεία, η ζωντανή εικόνα απο την κάμερα σε πραγματικό χρόνο, αλλά και η εικόνα του χώρου σε πραγματικό χρόνο απο τον αισθητήρα LiDAR. Με την χρήση της κάμερας επίσης, μπορούμε να ανιχνεύσουμε αντικείμενα τα οποία έχουμε αποθηκεύσει στην βάση μας προς αναγνώριση μέσω του πακέτου **find_object**.



Τέλος μπορεί να γίνει εξαγωγή αυτής της χαρτογράφησης σε αρχείο με σκοπό να εκμεταλλευτούμε αυτόν τον χάρτη αργότερα για την αναβάθμιση της πλοήγησης ενός οχήματος για διάφορες διεργασίες και μεταφορές στο περιβάλλον αυτό, να αυξήσουμε την παραγωγικότητα της εργασίας, την απόδοση και την αποτελεσματικότητα, και να μειώσουμε τους χρόνους παράδοσης.

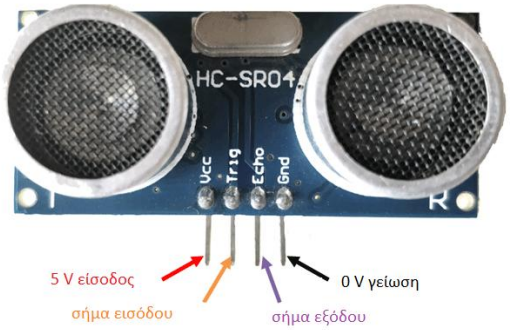


παράδειγμα εξαγωγής της χαρτογράφησης σε εικόνα

2.5. Μελλοντική επέκταση και βελτιστοποίηση.

Το ρομπότ που δημιουργήσαμε έχει πολλές δυνατότητες επέκτασης, βελτιστοποίησης και αναβάθμισης.
Το λειτουργικό του σύστημα, όπως και τα μηχανολογικά του μέρη, είναι σε θέση να υποστηρίξουν μία μεγάλη σειρά από αισθητήρες και εφαρμογές.
Συγκεκριμένα χρησιμοποιώντας τις πολλαπλές εξόδους τάσης της πλακέτας τροφοδοσίας, τις διαθέσιμες θύρες εξόδου και εισόδου του Pi Face Digital 2 όπως και τα διαθέσιμα GPIO του Raspberry Pi 2, μπορούν να συνδεθούν διαφόρων ειδών αισθητήρια όργανα μερικά από τα οποία είναι τα παρακάτω:

- Ultrasonic sensors.



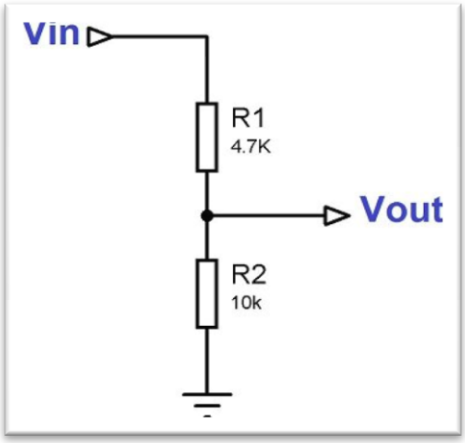
Οι αισθητήρες υπερήχων είναι σε θέση αφού τους δοθεί τάση 5V και συνδεθούν κατάλληλα με τον επεξεργαστή ώστε να τρέξει μία ρουτίνα αποτελεσματικά, να πληροφορούν το ρομπότ για την εγγύτητα σε κάποιο εμπόδιο - επιφάνεια ή αντικείμενο .

Στο συγκεκριμένο ρομπότ , όπου τον ρόλο της απεικόνισης και άρα της αντίληψης τον έχει ο RP LiDAR, μιας και είναι υπεύθυνος για την χαρτογράφηση του περιβάλλοντος χώρου, ο συγκεκριμένος αισθητήρας μπορεί να λειτουργήσει υποστηρικτικά, για τυχόν εμπόδιο - αντικείμενο που παρουσιάστηκε κατόπιν της χαρτογράφησης που έκανε το ρομπότ.
Επίσης προσαρμοσμένος με κατεύθυνση προς το έδαφος στο εμπρόσθιο ή άλλο - άλλα μέρη του ρομπότ, για να μπορεί έγκαιρα να ειδοποιήσει για τυχόν τρύπες στο έδαφος ή εμπόδια χαμηλού προφίλ .

Σε άλλα ρομποτικά οχήματα , όπου ο LiDAR απουσιάζει ως αισθητήριο όργανο, οι ultrasonic sensors είναι απαραίτητοι ίσως να υπάρχουν, μιας και είναι εύκολης χρήσης αισθητήρες, χαμηλού κόστους, και έχουν

δυνατότητα αρκετά μεγάλης απόστασης εντοπισμού και γωνίας. (2-40cm εντοπισμού, με κατά προσέγγιση σφάλμα του 1%)

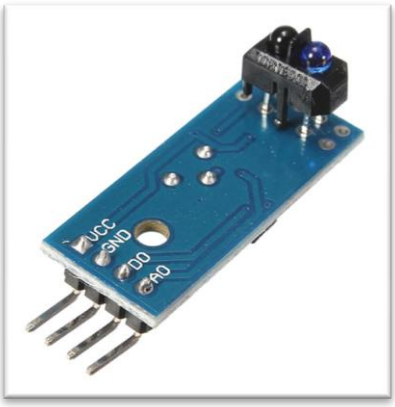
Καλό είναι να δοθεί προσοχή όμως στο γεγονός ότι τα ανακλώμενα ηχητικά κύματα, είναι "απορροφήσιμα" και ευκόλως διασκορπιζόμενα από κάποιες επιφάνειες, που έχει ως αποτέλεσμα την χαμηλή ή και λανθάνουσα αντιληπτική ικανότητα του αισθητήρα.
Επίσης οι είσοδοι του Pi λειτουργούν με 3.3V. Ως εκ τούτου, θα πρέπει να δημιουργηθεί μία διάταξη (voltage divider) ώστε να μην βλάψουμε με υπέρταση τον Pi, μιας και ο αισθητήρας λειτουργεί με 5V τάση, και περίπου 15mA ένταση.



Διάταξη voltage divider

- IR sensors .

Οι αισθητήρες εγγύτητας υπέρυθρης ακτινοβολίας, είναι αισθητήρες οι οποίοι μπορούν να διαδραματίσουν διπλό ρόλο σε χρήση .
Πρωτίστως μπορούν να χρησιμοποιηθούν με αντίστοιχο τρόπο όπως και οι ultrasonic, δηλαδή ως αισθητήρες εγγύτητας.



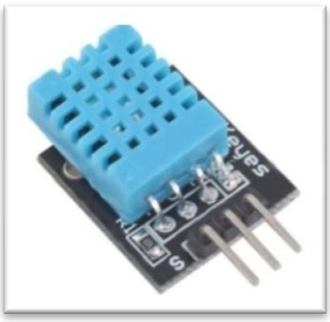
αισθητήρας IR sensor

Η περιορισμένη ακτίνα δράσης τους όμως σε μόλις μερικά εκατοστά (2-3cm είναι ένα μειονέκτημα, ωστόσο είναι πολύ πιο ακριβείς ως αισθητήρες, μιας και δεν επηρεάζονται τόσο εύκολα από το υλικό της επιφάνειας που προσπίπτουν .
Δευτερευόντως είναι κατάλληλοι για χρήση στο να αντιλαμβάνονται χρώματα (αντίληψη συγκεκριμένου μήκους κύματος χρώματος) .
Αυτή η ικανότητα του αισθητήρα, μπορεί να κάνει το ρομπότ ικανό, να ακολουθεί μία "γραμμή" συγκεκριμένου χρώματος στο έδαφος ή να συλλέγει κάποια πληροφορία όταν συναντά ο αισθητήρας το συγκεκριμένο χρώμα ως στίγμα στο έδαφος, μιας και είναι αυτό που τον διεγείρει.

Άλλοι αισθητήρες που θα μπορούσαν να χρησιμοποιηθούν για διάφορες εφαρμογές με πιο εξιδεικευμένο και ειδικό ρόλο είναι: Αισθητήρες καπνού, υγρασίας, θερμοκρασίας, κίνησης κ.α.

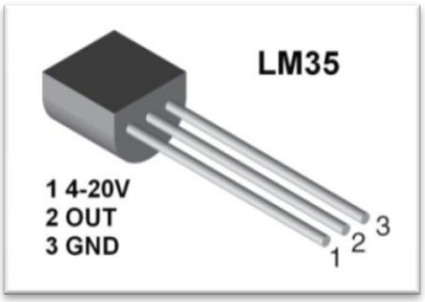
αισθητήρας αερίων

αισθητήρας κίνησης

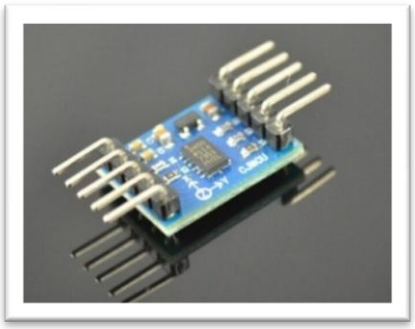


αισθητήρας υγρασίας

Επίσης αισθητήρες προσανατολισμού όπως γυροσκόπιο ή αισθητήρας επιτάχυνσης βαρύτητας.

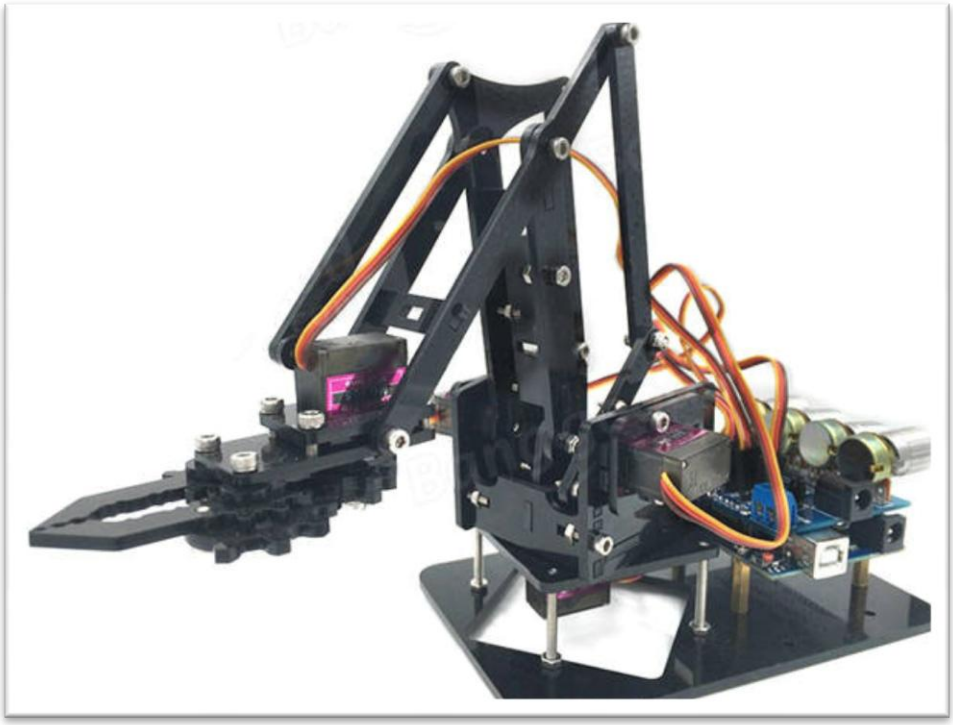


αισθητήρας θερμοκρασίας



αισθητήρας επιτάχυνσης βαρύτητας

Σημαντική επέκταση θα μπορούσε να γίνει και όσο αναφορά την άμεση αλληλεπίδραση του ρομπότ με το περιβάλλον , όπως για παράδειγμα η εισαγωγή ενός ρομποτικού servo βραχίονα ή μία κινούμενη αρπάγη, ώστε να υπάρχει η δυνατότητα να πιάνει, και να μετακινεί αντικείμενα.



ρομποτικός σερβοβραχίονας με αρπάγη

Πρόκειται σαφώς για μία σύνθετη και δύσκολη ενέργεια, για πολλούς λόγους.

Η συγκεκριμένη επέκταση, στην παρούσα διαμόρφωση του ρομπότ , μπορεί να φέρει στα όρια την επεξεργαστική ισχύ του Raspberry Pi 2 Model B+, αφού ήδη ο επεξεργαστής είναι φορτωμένος με αρκετές λειτουργίες.

Μπορεί λοιπόν, να χρειαστεί αντικατάσταση με έναν πιο δυνατό επεξεργαστή, που θα μπορεί ταυτόχρονα να διαχειρίζεται παραπάνω λειτουργίες και πιο αποτελεσματικά ή η εγκατάσταση ενός δεύτερου επεξεργαστή ως συνεργού (slave) του Pi αποκλειστικά για την συγκεκριμένη εφαρμογή.

Για να επιτευχθεί αυτό, απαιτούνται καλές γνώσεις προγραμματισμού, και πολλές προσπάθειες μέχρι ένας ρομποτικός βραχίονας να μπορεί αυτόνομα να λειτουργεί αποτελεσματικά.

Επίσης η συγκεκριμένη επέκταση, προϋποθέτει να υπάρχει κάμερα βάθους (depth camera) εγκατεστημένη επάνω στο ρομπότ , και όχι μία απλή κάμερα . Αυτό συμβαίνει , διότι για να μπορέσει να πιάσει κάτι αποτελεσματικά ένα ρομπότ , θα πρέπει να έχει την δυνατότητα αντίληψης όχι μόνο του χώρου και των αντικειμένων του ,αλλά και του βάθους αυτού που βλέπει .

παράδειγμα depth κάμερας

Τέλος απαιτούνται επιπλέον αισθητήρες που θα μπορούν να υποστηρίξουν τον βραχίονα ώστε να μην υπάρχει περιθώριο λάθους στις κινήσεις του , αφού για να καταφέρει να πιάσει κάτι , απαιτείται απόλυτη ακρίβεια και μηδενικές αποκλίσεις .

3. ΕΠΙΛΟΓΟΣ

3.1. Πληροφορίες για αυριανούς κατασκευαστές.

Οι αυριανοί ενδιαφερόμενοι κατασκευαστές και προγραμματιστές , πριν ξεκινήσουν την υλοποίηση ενός τέτοιου εγχειρήματος θα πρέπει να γνωρίζουν ότι δεν πρόκειται για μία εύκολη και απλή υπόθεση , αλλά ούτε ακατόρθωτη .
Αναλόγως των γνώσεων που κατέχουν γύρω από τον προγραμματισμό, υπολογιστές, μηχανολογικά, ηλεκτρονικά κ.α. ,πρέπει αυτοί να επιλέξουν και τον βαθμό δυσκολίας που επιθυμούν .
Σίγουρα ο τομέας έχει πάρα πολλές διαβαθμίσεις δυσκολίας, αλλά και κόστους. Πρωτίστως λοιπόν ,θα πρέπει να γίνει σοφή επιλογή αυτών που θέλουν να αναπτύξουν ένα τέτοιο ρομπότ , στο τι δυνατότητες θα θέλουν να έχει . Αν είναι η πρώτη φορά που κάποιος ασχολείται με κάτι παρόμοιο , συμβουλεύουμε ρητά να μην βάλει ψηλά τον πήχη , και να αρχίσει από απλές εφαρμογές. Παραδείγματος χάρη , να βάζει μικρούς στόχους αρχικά , και στην συνέχεια , με την επίτευξη αυτών , να εμπλουτίζει το εγχείρημα του. Αφού επιλεχθεί ο στόχος των δυνατοτήτων του ρομπότ, θα πρέπει να επιλεχθεί πρώτα ο επεξεργαστής που θα χρησιμοποιηθεί και μετά όλα τα άλλα.

Ο Επεξεργαστής αποτελεί το βασικό κομμάτι του εγχειρήματος , όχι μόνον ως κριτήριο επιλογής την υπολογιστική του ισχύ αλλά και των γενικότερων δυνατοτήτων του , όπως:
Δυνατότητες συνδέσεων περιφερειακών (mounted πλακέτες , modules , θύρες, αισθητήρων κ.α.) , Γλώσσα προγραμματισμού, κόστος , δυνατότητα ελέγχου κινητήρων , γραφικό περιβάλλον .

Δευτερεόντως, ο τρόπος κίνησης του ρομπότ στο περιβάλλον .
Οι τρόποι κίνησης είναι πολλοί και ο κάθε ένας έχει την δική του δυσκολία αλλά και χρησιμότητα .
Όσο αναφορά ρομποτικό όχημα που θα κινείται στην ξηρά , θα πρέπει να επιλεχθεί το πλήθος των κινητήρων , αν θα έχει τιμόνι ή αν θα στρίβει με διαφορετικό τρόπο , (ερπυστριοφόρο) , πολυκατευθυντικές ρόδες κ.α.

Σημειωτέο ότι προκύπτουν διαφορετικές ανάγκες προγραμματισμού, αναλόγως την επιλογή του τρόπου κίνησης .

Στην συνέχεια θα προκύψει το ζήτημα της αντίληψης του ρομπότ όταν αποκτήσει την δυνατότητα της κίνησης στον χώρο .Με ποιον τρόπο θα αντιλαμβάνεται που βρίσκεται στον χώρο , τι "αισθάνεται" , τι βλέπει και πως αντιδρά. Για το συγκεκριμένο ζήτημα , η ανάλυση στην παρούσα πτυχιακή είναι αδύνατη, μιας και πρόκειται για ένα από τα πιο σύνθετα προβλήματα σε παρόμοιες εφαρμογές, και οι τρόποι ποικίλουν .

4. Βιβλιογραφία

URL : https://el.wikipedia.org/wiki/Ρομπότ/

URL : https://en.wikipedia.org/wiki/William_Grey_Walter

Čapek, K. (1920). *Rossum's Universal Robot (R.U.R).* Translated by Paul Selver and Nigel Playfair Mineola, New York: Dover Publications

Asimov, I. (1942). *Runaround. Robot series*, Astounding Science Fiction, Street & Smith.

Asimov I. (1950). *I, Robot.* Gnome Press

Ηροδότου Ιστορίες .Υπουργείο Εθνικής Παιδείας και Θρησκευμάτων. Παιδαγωγικό Ινστιτούτο . Ζωή Σπανάκου. Μετάφραση: Ηλίας Σ. Σπυρόπουλος

URL : http://athinodromio.gr/η-ιστορία-των-ρομπότ-η-εξέλιξή-τους

http://nefeli.lib.teicrete.gr/browse/stef/epp/2015/AthanasakiDespoina/attached-document-1446739716-939836-18786/AthanasakiDespoina2015.pdf

URL: http://www.ros.org/about-ros/

Lentin J. (2015). Mastering ROS for Robotics Programming. Packt Publishing Ltd. p.3.

URL: http://wiki.ros.org/ROS/Tutorials/rosdep

Fundamentals of Robotics: Analysis and Control Robert J. Schilling 1990

ΕΙΣΑΓΩΓΗ ΣΤΗ ΡΟΜΠΟΤΙΚΗ, Μηχανική και Αυτόματος Έλεγχος John J.Craig 2009