ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ

ΤΜΗΜΑ ΨΗΦΙΑΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

Σχεδίαση και Υλοποίηση Συστήματος Αναγνώρισης Θέσης και Καθορισμού Διαδρομής σε Εσωτερικό Χώρο με Ρομποτικό Αμαξίδιο Μικροϋπολογιστή Raspberry Pi 3

του Κουτρουμάνου Ιάσονος, φοιτητή Ψηφιακών συστημάτων του Πανεπιστημίου Πειραιώς.

Κουτρουμάνος Ιάσων (Ε12081)

Επιβλέπων Καθηγητής: Μηλιώνης Απόστολος

Πτυχιακή Εργασία που υποβλήθηκε στο Τμήμα Ψηφιακών Συστημάτων του Πανεπιστημίου Πειραιώς ως μέρος των απαιτήσεων για την απόκτηση του προπτυχιακού τίτλου σπουδών.

Πειραιάς

2018-2019

Περιεχόμενα:

Αντικείμενο πτυχιακής εργασίας:

Αντικείμενο της παρούσας πτυχιακής εργασίας είναι ο σχεδιασμός, η ανάλυση και η υλοποίηση ενός *συστήματος αναγνώρισης θέσης και καθορισμού διαδρομής σε εσωτερικό χώρο με τη τεχνολογία IOT*.
Το αποτέλεσμα της εργασίας αυτής έχει δύο κατηγορίες χρηστών:

- τους χρήστες οι οποίοι ελέγχουν το σύστημα μέσω input devices, όπως είναι ένα πληκτρολόγιο,
- το ίδιο το σύστημα, το οποίο με κατάλληλο σχεδιασμό, βρίσκεται σε θέση να ελέγχει πλήρως τον εαυτό του και αφετέρου να παράγει δεδομένα τα οποία επιστρέφονται πίσω στον χρήστη ο οποίος ελέγχει τη λειτουργία του.

Οι χρήστες μπορούν να διεκπεραιώνουν τις εξής λειτουργίες:

- Έλεγχο με πληκτρολόγιο του συστήματος για καθορισμό των κινήσεων του.
- Καθορισμός διαδρομής του συστήματος.
- Χειροκίνητη «χαρτογράφηση» εσωτερικού χώρου.
- Έλεγχος απόστασης σε αντικείμενα, τα οποία βρίσκονται στην πορεία του, μετά από εντολή του χρήστη.
- Έλεγχος δεδομένων που καταγράφονται από το σύστημα.
- Αποθήκευση δεδομένων που καταγράφονται από το σύστημα.

Το σύστημα μπορεί να διεκπεραιώνει τις εξής λειτουργίες:

- Ανατροφοδότηση στο χρήστη δεδομένων σχετικά με την θέση στην οποία βρίσκεται το σύστημα σε ένα χώρο.
- Λήψη απόφασης για κατεύθυνση κίνησης.
- Εντοπισμός εμποδίων τα οποία ίσως μπορεί να εμποδίσουν την απρόσκοπτη ολοκλήρωση της πορείας του.
- Εξερεύνηση άγνωστου χώρου και χαρτογράφηση αυτού καθώς και εμποδίων που τυχόν βρίσκονται μέσα σε αυτόν τον χώρο.(SLAM Algorithm)
- Αποθήκευση των δεδομένων αυτών.

Σκοπός της εργασίας:

Σκοπός της εργασίας και της υλοποίησης του συστήματος αυτού είναι η εύρεση λύσεων για την ολοένα και μεγαλύτερη ανάγκη που παρουσιάζεται στις μέρες μας για αυτοματοποιημένα συστήματα υποβοήθησης. Οι εφαρμογές που έχει το σύστημα αυτό είναι ανάλογες των αναγκών των επιμέρους χρηστών. Επίσης με τις κατάλληλες τροποποιήσεις μπορεί να αποκτήσει ό,τι δυνατότητες απαιτεί ο καταναλωτής στον οποίο απευθύνεται. Μερικές από αυτές πχ μπορεί να είναι:

- Χρήση του συστήματος για την εκμάθηση προγραμματισμού και βασικών αρχών αυτοματισμού σε μαθητές διαφόρων ηλικιών.
- Χρήση από συνεργεία διάσωσης για εντοπισμό ατόμων σε κτήρια που έχουν καταρρεύσει από σεισμούς, εκρήξεις ή άλλους λόγους.
- Χρήση του από τις αρχές με σκοπό την αντικατάσταση ανθρώπινου δυναμικού σε περιπτώσεις που απειλείται η υπόσταση του, όπως ο εντοπισμός βομβών.
- Χρήση ως συμπλήρωμα συστήματος συναγερμού μέσα σε σπίτια, δηλαδή να είναι συνδεδεμένο με το σύστημα συναγερμού και να βιντεοσκοπεί ή να φωτογραφίζει το σπίτι σε περίπτωση που χτυπήσει ο συναγερμός ή ανοίξει κάποια πόρτα και να αποστέλλει τα δεδομένα στον ιδιοκτήτη.
- Χρήση μέσα σε εσωτερικούς χώρους για την υποβοήθηση ατόμων με διάφορες μορφές αναπηρίας προσφέροντας βοήθεια όπως:
    - να τους δείχνει ακολουθώντας το τον δρόμο για συγκεκριμένα δωμάτια του σπιτιού,
    - να τους πηγαίνει αντικείμενα που χρειάζονται σε κάποια δεδομένη στιγμή σε οποιοδήποτε χώρο βρίσκονται μέσα στο σπίτι,
    - να παρακολουθεί αδύναμα άτομα και να ενημερώνει τους οικείους σε περίπτωση ανάγκης.
    - να βοηθά στη καθαριότητα του σπιτιού.

Ορισμός συστήματος:

Ως σύστημα έχουμε ορίσει τα εξής:

- Ένα τετράτροχο μικρό αμαξίδιο το οποίο θα περιέχει ένα μοτέρ για κάθε ρόδα, 4 στο σύνολο.
- Μια ηλεκτρονική πλακέτα Raspberry PI 3 model B +
- Ένα H-Bridge το οποίο θα είναι υπεύθυνο για την καλύτερη λειτουργία και το συντονισμό των κινήσεων
- Ένα breadboard στο οποίο θα τοποθετηθεί ένας αισθητήρας απόστασης(μοντέλο HC-SR04)
- Ένα σετ μπαταριών από 5AA ή 2 μπαταρίες τύπου 18650 που θα είναι υπεύθυνες για την ηλεκτροδότηση των τροχών και ένα power-bank το οποίο θα τροφοδοτεί με ενέργεια το Raspberry PI.

Λογισμικό-Γλώσσες υλοποίησης:

Στο Raspberry χρησιμοποιείται ως λογισμικό το Rasbian το οποίο και συνίσταται για τη συγκεκριμένη πλακέτα και αποτελεί μια διανομή Linux.

Η γλώσσα που χρησιμοποιείται για την υλοποίηση των script είναι η python μέσα από τον nano editor ο οποίος προσφέρει δημιουργία script μέσω terminal και είναι εύχρηστος και απλός.

Στάδια υλοποίησης:

Παρακάτω παρουσιάζονται επιγραμματικά τα στάδια υλοποίησης:

1. *Συναρμολόγηση του αμαξιδίου.*
2. *Εγκατάσταση λογισμικού στο Raspberry*
3. *Σύνδεση H-bridge με το Raspberry και το αμαξίδιο*
4. *Σύνδεση Raspberry σε υπολογιστή για ασύρματο έλεγχο του και ευκολότερη εφαρμογή των scripts*
5. *Σύνταξη script που ελέγχει τις κινήσεις και με την εισαγωγή σε αυτό χρόνου σε δευτερόλεπτα να κινεί το σύστημα.*
6. *Σύνταξη script που ελέγχει τις κινήσεις και μέσω πληκτρολογίου ο χρήστης καθορίζει την πορεία του συστήματος.*
7. *Σύνταξη script που αξιοποιεί τα δεδομένα του αισθητήρα HC-SR04.*
8. *Σύνδεση δεδομένων αισθητήρα με το script των κινήσεων με στόχο να ειδοποιείται ο χρήστης για υπάρχοντα εμπόδια στην πορεία του.*
9. *Αυτοματοποίηση κινήσεων συστήματος και συνδυασμός δεδομένων με αυτά του αισθητήρα για αποφυγή εμποδίων από το ίδιο το σύστημα.*
10. *Υλοποίηση και σύνδεση SLAM αλγορίθμου με κινήσεις που καθορίζονται από το χρήστη (8.) και μέσω της αυτοματοποιημένης κίνησης (9.) με στόχο την αναγνώριση περιοχών και αποθήκευση αυτών.*
11. *Αποθήκευση δεδομένων τοπικά ή σε υπολογιστή Windows.*

Υλοποίηση και ανάλυση hardware

Προκειμένου το σύστημα να μπορεί να καλύψει τις ανάγκες μας και να λειτουργήσει ομαλά και αποτελεσματικά έπρεπε να γίνει μια λίστα με τα εξαρτήματα που θα χρησιμοποιηθούν. Μετά την αγορά των εξαρτημάτων επόμενο στάδιο είναι η σύνδεση τους και ο προγραμματισμός τους.

Α) Κόστος και Ανάλυση Hardware

Το πρώτο εξάρτημα που χρειάστηκε να αγοραστεί ήταν το ίδιο το αμαξίδιο. Αυτό περιείχε μέσα τα μοτέρ καθώς και τις 4 ρόδες του, όπως επίσης τις απαραίτητες βίδες, κατσαβίδι και οδηγίες για την συναρμολόγηση τους.

Δεύτερο βασικό εξάρτημα ήταν το Raspberry PI. Το μοντέλο που επιλέχθηκε ήταν το Raspberry PI 3 model B+ όντας το πιο καινούργιο αλλά και καλό σε προδιαγραφές μοντέλο.

Προκειμένου να λειτουργήσει το Raspberry χρειαζόταν εξωτερικά ένας φορτιστής για σύνδεση σε πρίζα τοίχου καθώς το USB του υπολογιστή δεν παρείχε σταθερή και απαραίτητη τάση με αποτέλεσμα να κινδυνεύουν τα αρχεία να γίνουν corrupted αλλά και η ασφάλεια της ίδιας της πλακέτας.

Καθώς το Raspberry δεν έχει εσωτερικό αποθηκευτικό χώρο μια κάρτα μνήμης micro SD ήταν απαραίτητη για την εγκατάσταση του Rasbian.

Ένα WIFI dongle αγοράστηκε το οποίο όταν συνδέεται σε θύρα USB του Raspberry προσφέρει καλύτερο σήμα WIFI.

Στη συνέχεια προκειμένου να ελέγχουμε σωστά τα μοτέρ και να μπορεί το όχημα μας να κινείται, όπως εμείς επιθυμούμε, ένα H-Bridge ήταν απαραίτητο. Με τη χρήση αυτού ελέγχουμε τη ροή του ρεύματος και άρα επιτρέπουμε σε συγκεκριμένες ρόδες να παίρνουν ή όχι ρεύμα και με ποιό τρόπο, επιτυγχάνοντας έτσι τις κινήσεις.

Για την σύνδεση όλων αυτών μεταξύ τους χρειάστηκαν jumper wires όλων των ειδών και σε διαφορετικά μήκη.

Επίσης, απαραίτητο εξάρτημα για την μέτρηση της απόστασης που αποτελεί και σημείο κλειδί του συστήματος ήταν η αγορά ενός αισθητήρα απόστασης καθώς και ένα breadboard για την σύνδεση του με το υπόλοιπο σύστημα.

Στη πορεία διαπιστώθηκε ότι ήταν απαραίτητοι και κάποιοι αντιστάτες οι οποίοι θα ελέγχουν το ρεύμα, ώστε να μην δημιουργηθεί βραχυκύκλωμα.

Το αρχικό breadboard όντας μικρό αντικαταστάθηκε από ένα μεγαλύτερο και καλύτερης ποιότητας.

Ανεξάρτητα αγοράστηκε και μια προστατευτική θήκη για το Raspberry καθώς και ένα κουτί που είχε σαν στόχο να μπουν όλα τα εξαρτήματα μέσα για προστασία. Το κουτί τελικά, σαν λύση απορρίφθηκε.

Για να λυθεί το πρόβλημα της εξάρτησης παροχής ρεύματος από πρίζα απαραίτητη ήταν και η αγορά ενός power bank το οποίο θα προσέδιδε αυτονομία πολλών ωρών στο σύστημα μας, για απρόσκοπτη λειτουργία.

Για το σκοπό αυτό, βοηθητικά, έγινε αντικατάσταση και των αρχικών μπαταριών τύπου AA με μπαταρίες τύπου 18650.

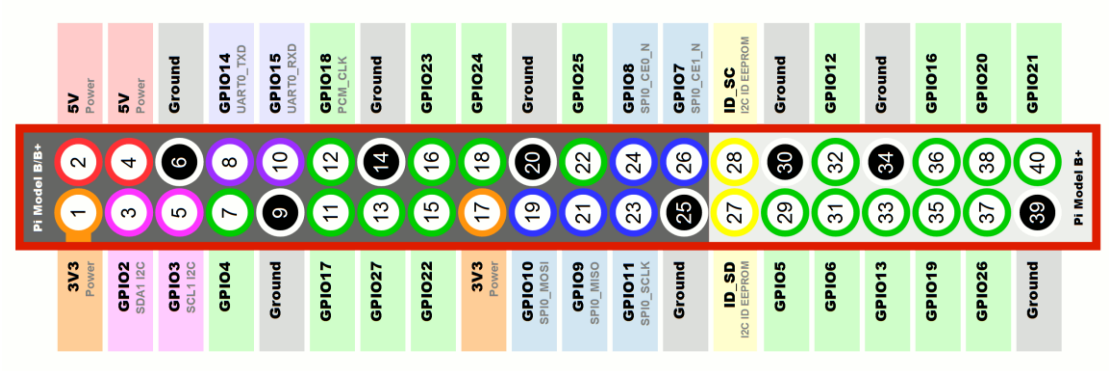
Β) Πίνακας Εξόδων:

Εξάρτημα(με σειρά αναφοράς)	Κόστος σε €
Multi-chassis 4wd kit(basic)	40,00
Raspberry PI 3 model B+	41,90
Power supply for Raspberry PI original	9,90
Micro SD card	9,50
WIFI Dongle	7,90
DC motor driver (L298N)	4,20
Jumper Wires x8	15,18
Ultrasonic sensor HC-SR04	2,50
Adafruit quarter sized breadboard	3,60
Resistor metal	0,10
Breadboard big	3,20
Raspberry PI 3 case	3,60
Project box	3,00
Power Bank	20,00
2 12650 batteries	11,28
2 spot Cell for 18650 battery	1,00
Charger for 18650 batteries	6,45
Άλλα έξοδα(μπαταριές AA, κουμπιά κλπ)	11,00
ΣΥΝΟΛΙΚΟ ΚΟΣΤΟΣ:	194,31

Γ) Υλοποίηση

Πρώτο βήμα για την υλοποίηση και συναρμολόγηση του συστήματος ήταν η σύνδεση των μοτέρ με τις ρόδες και η προσαρμογή τους επάνω στο μεταλλικό σκελετό του αμαξιδίου. Στη συνέχεια από την επίσημη ιστοσελίδα του Raspberry επιλέχθηκε η διανομή Rasbian και εγκαταστάθηκε στην SD κάρτα. Στη συνέχεια, μετά από αναζήτηση στο internet επιλέχθηκε για δημιουργία του κώδικα ο nano editor και το VNC viewer για περιβάλλον απομακρυσμένης σύνδεσης με το Laptop ή το Desktop που φέρνει λογισμικό Windows.

Επόμενο βήμα ήταν η μελέτη λειτουργίας των GPIO pins που έχει το Raspberry καθώς και ο τρόπος με τον οποίο θα συνδεθούν σε αυτά τα εξαρτήματά μας. Το μοντέλο που επιλέχθηκε διαθέτει 40 pins, όπως φαίνεται και στη φωτογραφία παρακάτω.



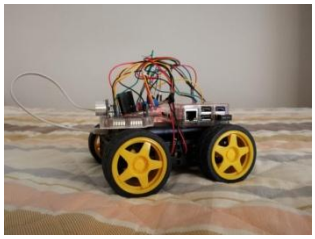
Έπειτα από κατανόηση των λειτουργιών τους ξεκίνησε η σύνδεση των εξαρτημάτων με το ίδιο το raspberry αλλά και μεταξύ του raspberry και του H-Bridge, με χρήση jumper wires. Συγκεκριμένα, στη θύρα 2 που παρέχει τάση 5V πραγματοποιήθηκε σύνδεση με την αντίστοιχη υποδοχή στο H-Bridge. Με το τρόπο αυτό το H-bridge ελέγχει το ρεύμα προς τις ρόδες χωρίς να υπάρχει κίνδυνος υπερφόρτωσης και άρα βραχυκυκλώματος του Raspberry. Στη θύρα 9 συνδέθηκε το ground μεταξύ Raspberry και H-bridge.

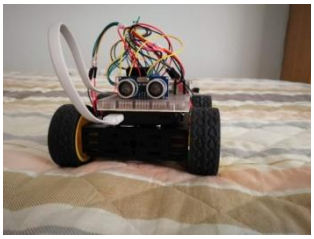
Τα pins 7,11,13,15 επιλεχθήκαν για τις 4 ρόδες του αμαξιδίου και συνδεθήκαν με τα αντίστοιχα input 1-4 στο H-Bridge. Η καθεαυτό σύνδεση των ροδών με το Raspberry δεν γίνεται να πραγματοποιηθεί, επειδή χρειάζεται το H-Bridge ως ελεγκτής. Κάθε ρόδα έχει δύο καλώδια τα οποία χωρίζονται ανάλογα με το ρεύμα. Το H-Bridge έχει 4 ειδικούς υποδοχείς για τις ρόδες, οι οποίοι βρίσκονται ανά 2 απέναντι ο ένας από τον άλλο. Για το λόγο αυτό, στη μια πλευρά συνδεθήκαν οι ρόδες της δεξιάς πλευράς και στην άλλη της αριστερής. Στη κάθε πλευρά επιλεχθήκαν τα καλώδια αναλόγως, ώστε να γίνει ο διαχωρισμός του ρεύματος και άρα της λειτουργίας των ροδών. Με το τρόπο, αυτό το H-Bridge μπορεί να ελέγξει που και πως θα στείλει ρεύμα και άρα πως θα στρίψει το αμαξίδιο σε συνδυασμό με το κώδικα που έχουμε ορίσει, ο οποίος «έρχεται» μέσα από τα input 1-4 του H-Bridge που είναι συνδεδεμένα με το Raspberry στα pins 7,11,13,15.

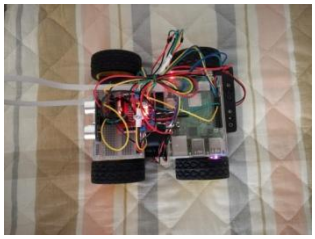
Επόμενο βήμα ήταν η σύνδεση του αισθητήρα και η αξιοποίηση των δεδομένων που αυτός παράγει. Επειδή και σε αυτή την περίπτωση, όπως και με τις ρόδες, ο αισθητήρας δεν ήταν δυνατό να συνδεθεί άμεσα με το Raspberry, το διαμεσολαβητή έκανε το breadboard. Το Breadboard είναι ένα εξάρτημα απαραίτητα για την συνδεσμολογία των εξαρτημάτων μας και την δημιουργία κυκλωμάτων. Το breadboard παρέχει την δυνατότητα σύνδεσης εξαρτημάτων χωρίς να χρειάζεται soldering, καθώς φέρει μικρά clips τα οποία κρατάνε σταθερά τα εξαρτήματα.

Ο αισθητήρας έχει τέσσερα pins τα οποία συνδέονται στο breadboard. Ένα το οποίο παρέχει γείωση, ένα το οποίο στέλνει υπέρηχο και άρα ξεκινά την λειτουργία του αισθητήρα, ένα για να δέχεται πίσω τα δεδομένα που αποστέλλει το προηγούμενο και ένα pin το οποίο παρέχει την απαραίτητη τάση. Ο αισθητήρας εκμεταλλεύεται στο έπακρο το φαινόμενο Doppler, το οποίο σε ορισμένες περιπτώσεις, όπως θα αναλυθεί και παρακάτω, έχει περιορισμούς. Η γείωση του αισθητήρα συνδέεται στην ίδια υποδοχή που είναι και αυτή για τις ρόδες επάνω στο H-Bridge. Στην αντίστοιχη θέση στο H-Bridge συνδέεται και η παροχή ρεύματος. Στο GPIO 24 του Raspberry συνδέεται το trigger του αισθητήρα και στο GPIO 25 το echo. Για αποφυγή βραχυκυκλώματος, επειδή γίνεται απευθείας σύνδεση με το Raspberry έγινε χρήση ενός resistor.

Στο επόμενος κεφάλαιο θα αναλυθεί ο αλγόριθμος που υλοποιήθηκε για τον έλεγχο της συσκευής και πως αυτός διαβάζει και αξιοποιεί τα δεδομένα από τα διάφορα εξαρτήματα του συστήματος.







Λειτουργική παρουσίαση συστήματος

Στο κεφάλαιο αυτό θα γίνει μια λειτουργική παρουσίαση του συστήματος και θα μελετηθεί η λειτουργία του. Ο τρόπος σύνδεσης του συστήματος καθώς και ο έλεγχος του απομακρυσμένα από υπολογιστή θα παρουσιαστεί το κεφάλαιο **User Manual** αναλυτικότερα. Επίσης, θα γίνει παρουσίαση του αλγορίθμου καθώς και τα βήματα που ακολουθήθηκαν για την διεκπεραίωση του και ο τρόπος σκέψης πίσω από την υλοποίηση. Ακόμη θα παρουσιαστούν στιγμιότυπα χρήσης του και όπως αυτό φαίνεται σε υπολογιστή και όπως αυτό είναι σε πραγματικό πείραμα.

Α) Περιγραφή Βημάτων υλοποίησης Αλγορίθμου

Προκειμένου να ελεγχθεί αποτελεσματικά το σύστημα και να συνταχθεί σωστά ο αλγόριθμος έπρεπε να γίνουν κάποια βήματα με συγκεκριμένη σειρά.

i)Κώδικας

Πρώτα έπρεπε να φτιαχτεί μια συνάρτηση η οποία θα αρχικοποιεί τις ρόδες.

```
1   import RPi.GPIO as gpio
2   import time
3
4   def init():
5       gpio.setmode(gpio.BOARD)
6       gpio.setup(7, gpio.OUT)
7       gpio.setup(11, gpio.OUT)
8       gpio.setup(13, gpio.OUT)
9       gpio.setup(15, gpio.OUT)
```

Αυτό επιτυγχάνεται μέσω του παραπάνω κώδικα εισάγοντας την βιβλιοθήκη που επιτρέπει τον έλεγχο των GPIO pins.

Στη συνέχεια, επόμενο βήμα ήταν η δημιουργία των συναρτήσεων που επιτρέπουν την κίνηση στις διάφορες κατευθύνσεις του οχήματος.

```
10
11  def forward(tf):
12      init()
13      gpio.output(7, False)
14      gpio.output(11, True)
15      gpio.output(13, True)
16      gpio.output(15, False)
17      time.sleep(tf)
18      gpio.cleanup()
19
20  def reverse(tf):
21      init()
22      gpio.output(7, True)
23      gpio.output(11, False)
24      gpio.output(13, False)
25      gpio.output(15, True)
26      time.sleep(tf)
27      gpio.cleanup()
28
29  def turn_right(tf):
30      init()
31      gpio.output(7, True)
32      gpio.output(11, True)
33      gpio.output(13, True)
34      gpio.output(15, False)
35      time.sleep(tf)
36      gpio.cleanup()
37
```

```
38  def turn_left(tf):
39      init()
40      gpio.output(7, False)
41      gpio.output(11, True)
42      gpio.output(13, False)
43      gpio.output(15, False)
44      time.sleep(tf)
45      gpio.cleanup()
46
47  def pivot_right(tf):
48      init()
49      gpio.output(7, True)
50      gpio.output(11, False)
51      gpio.output(13, True)
52      gpio.output(15, False)
53      time.sleep(tf)
54      gpio.cleanup()
55
56  def pivot_left(tf):
57      init()
58      gpio.output(7, False)
59      gpio.output(11, True)
60      gpio.output(13, False)
61      gpio.output(15, True)
62      time.sleep(tf)
63      gpio.cleanup()
64
65  forward(1)
```

Στο τέλος, καλώντας την συνάρτηση που θέλουμε και επιλέγοντας χρόνο κινείται το ρομπότ, για όσο χρόνο εμείς ορίσαμε.

Επόμενο βήμα ήταν η τροποποίηση του κώδικα, ώστε να μπορεί ο χρήστης να ελέγχει το σύστημα μέσω πληκτρολογίου. Προκειμένου να ολοκληρωθεί το βήμα αυτό έγινε χρήση του Tkinter της python.

```
73  def key_input(event):
74      init()
75      print ('Key:', event.char)
76      key_press = event.char
77      sleep_time = 0.10
78      if key_press.lower() == 'w':
79          forward(sleep_time)
80
81      elif key_press.lower() == 's':
82          reverse(sleep_time)
83
84      elif key_press.lower() == 'a':
85          turn_left(sleep_time)
86
87      elif key_press.lower() == 'd':
88          turn_right(sleep_time)
89
90      elif key_press.lower() == 'q':
91          pivot_left(sleep_time)
92
93      elif key_press.lower() == 'e':
94          pivot_right(sleep_time)
95
96      elif key_press.lower() == 'z':
97      print('OK I exit. Goodbye!')
98      gpio.cleanup()
99      sys.exit()
100
101     else:
102         gpio.cleanup()
```

Μετά από τον έλεγχο για ομαλή λειτουργία όλων των παραπάνω είχε φτάσει το σημείο που θα εισάγονταν τα δεδομένα του αισθητήρα στον κώδικα. Για το σκοπό αυτό ένα καινούργιο αρχείο κώδικα συντάχθηκε, το οποίο αξιοποιεί τα δεδομένα του κώδικα και μας επιτρέπει να εισάγουμε το script, όπου μας κρίνεται απαραίτητη η χρήση του.

```
import RPi.GPIO as GPIO
import time

def distance():
    GPIO.setmode(GPIO.BOARD)

    TRIG = 24
    ECHO = 23

    GPIO.setup(TRIG,GPIO.OUT)
    GPIO.output(TRIG,0)

    GPIO.setup(ECHO,GPIO.IN)

    time.sleep(0.1)

#    print "Starting"

    GPIO.output(TRIG,1)
    time.sleep(0.00001)
    GPIO.output(TRIG,0)

    while GPIO.input(ECHO) == 0:
        pass
    start = time.time()

    while GPIO.input(ECHO) == 1:
        pass
    stop = time.time()

    distance = (stop - start)*17000
    return distance
    GPIO.cleanup()
#print distance()


#speed=340m/s
#emas to shma paei kai erxetai ara 2fores h apostash
#speed=distance/time
#340=2d/time
#d=170*time
#170*100 gia CM
```

Συνεχίζοντας, έπρεπε να προταθεί μια λύση για το τρόπο με τον οποίο το σύστημα θα αντιλαμβανόταν τον κόσμο και θα αποθήκευε τα δεδομένα που θα διάβαζε. Η πιο βιώσιμη λύση, δεδομένου των περιορισμών του αισθητήρα, ήταν η μετατροπή της απόστασης σε ένα τετράγωνο πίνακα ή η δυνατότητα να επιλέγει ο χρήστης το μέγεθος ενός πίνακα. Ο πίνακας αρχικά θα ήταν γεμάτος 0 τα οποία θα αναπαριστώνται με μαύρο χρώμα. Όσο το σύστημα χαρτογραφεί θα αλλάζει το χρώμα των κελιών που επισκέπτεται ανάλογα με το πόση απόσταση έχει μπροστά του - σε κόκκινο αν πλησιάζει σε εμπόδιο ή γαλάζιο αν έχει αρκετό χώρο ακόμα. Ο κώδικας για την δημιουργία του πίνακα παρουσιάζεται παρακάτω.

```python
import RPi.GPIO as gpio
from measure_distance import distance
import math
from move_with_time_given import *
import numpy as np




def do(): #metra apostash kai vriskei kelia gia pinaka
    dis = int(round(distance()))
    cels = dis/10+1
    return cels
    gpio.cleanup()


def cols():
    #time.sleep(1)
    c1 = do()
    #print("front will have", c1,"cells") #mprosta
    pivot_left(0.5) #strivei 180moires
    time.sleep(2.0)
    c2 = do() #metra ta pisw
    #print("back will have", c2, "cells")
    c=(c1+c2)*1 #prosthetei
    #print("A total of", c,"cols") #synolo sthlwn

    return c
    gpio.cleanup()

def rows():
    pivot_left(0.5) #opws einai kai koita pisw stivei kai koita dexia
    r1 = do() #metra
    #print("right will have", r1,"cells")
    time.sleep(3.0)
    pivot_left(0.5) #strivei apo dexia 180 moires kai koita aristera
    r2 = do()
    #print("left will have", r2, "cells")
    r=(r1+r2)*1 #kanei add
    #print("A total of", r, "rows")

    return r
    gpio.cleanup()

def mat():
    #ftiaxnei pinaka me tis diastaseis aytes
    x=np.zeros((rows(),cols()))
    print(x)
    return x
    gpio.cleanup()
```

Η δυνατότητα επιλογής από το χρήστη του μεγέθους του πίνακα παρουσιάζεται με comments στην επόμενη φωτογραφία. Επίσης, παρουσιάζεται και ο τρόπος με τον οποίο το κυρίως script δέχεται, αποθηκεύει και αξιοποιεί τον δημιουργημένο πίνακα.

```
69  #r = int(input("Please enter the number of rows: "))
70  #c = int(input("Please enter ther number of colums: "))
71  #print("You entered: ", r , "rows and ", c , "colums.")
72  #x=np.zeros((r,c))
73  #gpio.cleanup()
74  rows=rows()
75  cols=cols()
76  #x=mat()
77  x=np.zeros([rows,cols,3], dtype=np.uint8)
78  i=rows/2
79  y=cols/2
80
81  #h Skepsh edw einai h prvth metrhsh na ginetai kai na apo8hkeyetai sto
82  #kentro tou pinaka apo thn arxh
83  init()
84  curDis = float(round(distance(),2))
85  x[rows/2,cols/2] = curDis
86  #print x
87  gpio.cleanup()
88
89
90  x[0,:]=[255,0,0]
91  x[:,0]=[255,0,0]
92  x[rows-1,:]=[255,0,0]
93  x[:,cols-1]=[255,0,0]
94
```

Όπως τονίζεται και στα σχόλια της φωτογραφίας κάνουμε την παραδοχή ότι το ρομποτάκι είναι τοποθετημένο στο κέντρο του δημιουργημένου πίνακα. Οι γραμμές 90 μέχρι 93 του κώδικα κοκκινίζουν τα άκρα του πίνακα, καθώς αποτελούν τα όρια μας σύμφωνα με το πρότυπο RGB.

Επόμενο βήμα ήταν το πέρασμα όλων αυτών των προσθηκών στον αλγόριθμο που ελέγχει ο χρήστης. Ο τρόπος που έγινε αυτό παρουσιάζεται παρακάτω.

```
103     if key_press.lower() == 'w' :
104         init()
105         curDis = float(round(distance(),2))
106         forward(sleep_time)
107         i=i-1
108         if curDis < 20:
109             x[i,y] = [255,0,0]
110             print('Current Distance is: ', curDis)
111             return x
112             gpio.cleanup()
113         else:
114             x[i,y] = [0,255,255]
115             print('Current Distance is: ',curDis)
116             return x
117             gpio.cleanup()
118         gpio.cleanup()
119
120     elif key_press.lower() == 's' :
121         init()
122         curDis = float(round(distance(),2))
123         reverse(sleep_time)
124         i=i+1
125         if curDis < 20:
126             x[i,y] = [255,0,0]
127             print('Current Distance is: ', curDis)
128             return x
129             gpio.cleanup()
130         else :
131             x[i,y] = [0,255,255]
132             print('Current Distance is: ',curDis)
133             return x
134             gpio.cleanup()
135         gpio.cleanup()
```

Με τον ίδιο τρόπο συνεχίζει και για όλες τις άλλες κινήσεις. Ουσιαστικά αυτό που κάνει είναι να μετακινείται με κάθε πάτημα πλήκτρου και ένα κελί προς την αντίστοιχη κατεύθυνση, όπως φαίνεται στις γραμμές 107 και 124. Έπειτα, ελέγχει την απόσταση και ανάλογα με τη μέτρηση χρωματίζει το κελί γαλάζιο ή κόκκινο.

Το σημείο που τερματίζεται ο αλγόριθμος καθώς και η δυνατότητα αποτύπωσης στιγμιότυπων και μέτρησης της απόστασης από το χρήστη παρουσιάζεται παρακάτω. Αξίζει να τονισθεί ότι επειδή η αρχική φωτογραφία είναι μικρού μεγέθους, για ευκολία στην διακριτικότητα του μονοπατιού καλείται και μια συνάρτηση από την έτοιμη βιβλιοθήκη numpy της Python, η οποία παίρνει την αρχική εικόνα και την μεγενθύνει σε συγκεκριμένο μέγεθος ορισμένο μέσα στο κώδικα(500x500 pixel).

```
210     elif key_press.lower() == 'z' :
211     print('Let me save what I have done so far')
212     print('OK, now I exit. Goodbye!')
213     img = Image.fromarray(x, 'RGB')
214     img.save('z.png')
215     img.show()
216
217     im2= Image.open("./z.png")
218     newsize =(500,500)
219     im2 = im2.resize(newsize)
220     im2.save('done.png')
221
222     sys.exit()
223
224     elif key_press.lower() == 'p' :
225     print('I just printed a copy of my table')
226     img = Image.fromarray(x, 'RGB')
227     img.save('y.png')
228     img.show()
229
230     im2= Image.open("./y.png")
231     newsize=(500,500)
232     im2 = im2.resize(newsize)
233     im2.save('Copy.png')
234
235
236     elif key_press.lower() == 'm' :
237     curDis = float(round(distance(),2))
238         print('Current Distance is: ', curDis)
239     gpio.cleanup()
240     else:
241         print('Wrong Button Pressed')
242     gpio.cleanup()
243
```

Αφού ολοκληρώθηκε και δοκιμάστηκε επιτυχώς ο αλγόριθμος χρήσης του συστήματος από το χρήστη, επόμενο βήμα ήταν η σύνταξη της αυτοματοποιημένης λύσης. Οι διαφορές, αν και καθοριστικές είναι μικρές σε σχέση με την προηγούμενη λύση και παρουσιάζονται στις παρακάτω φωτογραφίες.

```
74  def check_front():
75      init()
76      dist = distance()
77
78      if dist < 15:
79          print('Too close: ', dist)
80      init()
81      reverse(0.9)
82      dist = distance()
83      if dist < 15:
84          print('Still too close: ',dist)
85          init()
86          pivot_left(0.9)
87          dist = distance()
88          if dist < 15:
89              print('I give up', dist)
90          img = Image.fromarray(x, 'RGB')
91              img.save('auto.png')
92              img.show()
93
94              im2= Image.open("./auto.png")
95              newsize =(500,500)
96              im2 = im2.resize(newsize)
97              im2.save('Auto_Done.png')
98          sys.exit()
99
```

Στη φωτογραφία αυτή το ρομποτάκι αξιοποιώντας τα δεδομένα του αισθητήρα ελέγχει το χώρο μπροστά του και ανάλογα με την απόσταση από υπάρχον εμπόδιο κάνει όπισθεν ή περιστροφή προς τα αριστερά.

Στη περίπτωση που η απόσταση παραμένει μικρή από το εμπόδιο και άρα έχει εγκλωβιστεί, γίνεται έξοδος από το πρόγραμμα και αποθήκευση της προόδου.

Στις φωτογραφίες που ακολουθούν παρουσιάζεται η συνάρτηση αυτονομίας και πώς αυτή λειτουργεί συνδυάζοντας όλα τα δεδομένα που υπάρχουν διαθέσιμα.

```
123 def autonomy():
124     tf = 0.20
125     z = random.randrange(0,5)
126     global i
127     global y
128 
129 #to h ka8orizei ton ari8mo twn forwn poy 8a trexei ka8e fora edw einai 5
130 #mporoume na valoume oti 8eloume
131     if z == 0:
132         for h in range(5):
133         check_front()
134             init()
135         forward(tf)
136         i=i-1
137         if curDis < 16:
138             #x[i,y] = curDis
139             x[i,y] = [255,0,0]
140             #print(x)
141             print('Current Distance is: ', curDis)
142             return x
143             gpio.cleanup()
144         else:
145             x[i,y] = [0,255,255]
146             #print(x)
147             print('Current Distance is: ',curDis)
148             return x
149             gpio.cleanup()
150 
151     elif z == 1:
152     for h in range(5):
153         check_front()
154         init()
155             pivot_left(tf)
156             y=y-1
157         if curDis < 16:
158             #x[i,y] = curDis
159                 x[i,y] = [255,0,0]
160                 #print(x)
161                 print('Current Distance is: ', curDis)
162                 return x
163             gpio.cleanup()
164 
165             else :
166                 x[i,y] = [0,255,255]
167                 #print(x)
168                 print('Current Distance is: ',curDis)
169                 return x
170             gpio.cleanup()
```

```
172     elif z == 2:
173     for h in range(5):
174         check_front()
175         init()
176         pivot_right(tf)
177             y=y+1
178         if curDis < 20:
179             #x[i,y] = curDis
180                 x[i,y] = [255,0,0]
181                 #print(x)
182                 print('Current Distance is: ', curDis)
183                 return x
184             gpio.cleanup()
185 
186             else :
187                 x[i,y] = [0,255,255]
188                 #print(x)
189                 print('Current Distance is: ',curDis)
190                 return x
191             gpio.cleanup()
192 
193     elif z == 3:
194     for h in range(5):
195         check_front()
196         init()
197         turn_left(tf)
198             i=i-1
199         y=y-1
200         if curDis < 16:
201             #x[i,y] = curDis
202                 x[i,y] = [255,0,0]
203                 #print(x)
204                 print('Current Distance is: ', curDis)
205                 return x
206             gpio.cleanup()
207 
208             else :
209                 x[i,y] = [0,255,255]
210                 #print(x)
211                 print('Current Distance is: ',curDis)
212                 return x
213             gpio.cleanup()
214
```

Στη φωτογραφία που ακολουθεί απεικονίζεται το τέλος της συνάρτησης αυτονομίας καθώς και ο τρόπος που αυτή τρέχει καθώς και τι δεδομένα θα δείξει στο χρήστη.

```
    elif z == 4:
        for h in range(5):
        check_front()
        init()
            turn_right(tf)
            i=i-1
            y=y+1
        if curDis < 16:
            #x[i,y] = curDis
                x[i,y] = [255,0,0]
                #print(x)
                print('Current Distance is: ', curDis)
                return x
            gpio.cleanup()

            else :
                x[i,y] = [0,255,255]
                #print(x)
                print('Current Distance is: ',curDis)
                return x
            gpio.cleanup()



for u in range(30):
    autonomy()
    print('i did it', u+1,'times')
    img = Image.fromarray(x, 'RGB')
    img.save('auto.png')
    img.show()

    im2= Image.open("./auto.png")
    newsize =(500,500)
    im2 = im2.resize(newsize)
    im2.save('Auto_Done.png')

```

Συνεχίζοντας στην επόμενη υπό-ενότητα, θα παρουσιαστούν κάποια βασικά στιγμιότυπα χρήσης του συστήματος. Επίσης πως τα δεδομένα απεικονίζονται στην κονσόλα ώστε να τα διαβάζει ο χρήστης. Ακόμη κάποια τυχαία αποτελέσματα που προκύπτουν από τις ενέργειες αυτές θα παρουσιαστούν. Εμπεριστατωμένο πείραμα όμως θα πραγματοποιηθεί στην επόμενη ενότητα. Εκεί θα γίνει περαιτέρω ανάλυση των αποτελεσμάτων και παρουσίαση αυτών.

ii) Στιγμιότυπα Αλγορίθμου

Δειγματοληπτικά παρουσιάζονται μερικά στιγμιότυπα του αλγορίθμου.

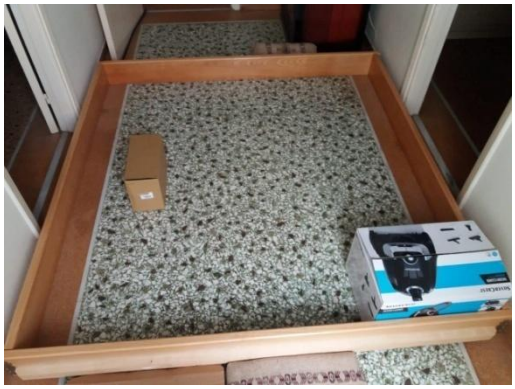
```
pi@raspberrypi:~/Desktop/Robot $ sudo python keyboard_full.py
Key: w
('Current Distance is: ', 18.65)
Key: w
('Current Distance is: ', 13.77)
Key: w
('Current Distance is: ', 8.62)
Key: w
('Current Distance is: ', 20.37)
Key: d
('Current Distance is: ', 18.58)
Key: a
('Current Distance is: ', 18.55)
Key: s
('Current Distance is: ', 9.49)
Key: w
('Current Distance is: ', 13.51)
Key: p
I just printed a copy of my table
```

```
Key: e
('Current Distance is: ', 18.22)
Key: m
('Current Distance is: ', 18.62)
Key: h
Wrong Button Pressed
Key: z
Let me save what I have done so far
OK, now I exit. Goodbye!
pi@raspberrypi:~/Desktop/Robot $
```

Β) Πείραμα σε οριοθετημένη περιοχή

Πρώτη όψη οριοθετημένης περιοχής

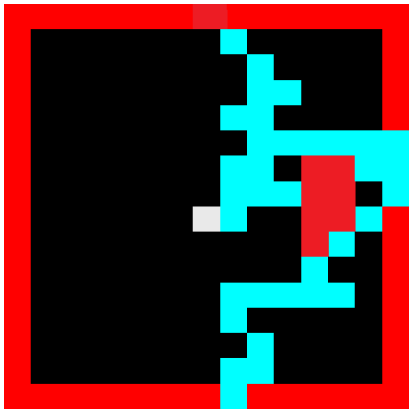
Δεύτερη όψη οριοθετημένης περιοχής



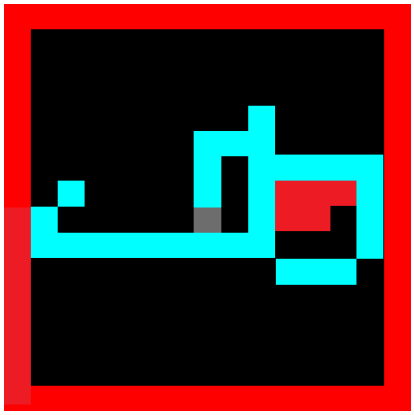
Επόμενο στάδιο ήταν η δοκιμή του συστήματος σε μια περιοχή οριοθετημένη, με στόχο την λεπτομερή αξιολόγηση του και καταγραφή της απόδοσης του αλγορίθμου που έχει υλοποιηθεί. Με την δοκιμή αυτή φανήκαν και όποια προβλήματα υπήρχαν στην λειτουργία ή την απόδοση του συστήματος. Η περιοχή που δοκιμάστηκε το σύστημα είναι ένας τετράγωνος χώρος 1,80τμ. Τυχαία, επιλέχθηκαν και τοποθετήθηκαν δύο εμπόδια, όπως αυτά φαίνονται στις φωτογραφίες, με στόχο τον έλεγχο αποφυγής εμποδίων του αλγορίθμου.

Τα προβλήματα που προέκυψαν αφορούν την φύση του αισθητήρα καθώς και την απουσία βαθμονόμησης των μοτέρ για καλύτερη μέτρηση διανυόμενης απόστασης. Περαιτέρω ανάλυση και προτεινόμενες λύσεις για τα προβλήματα αυτά θα δούμε στο επόμενο κεφάλαιο.

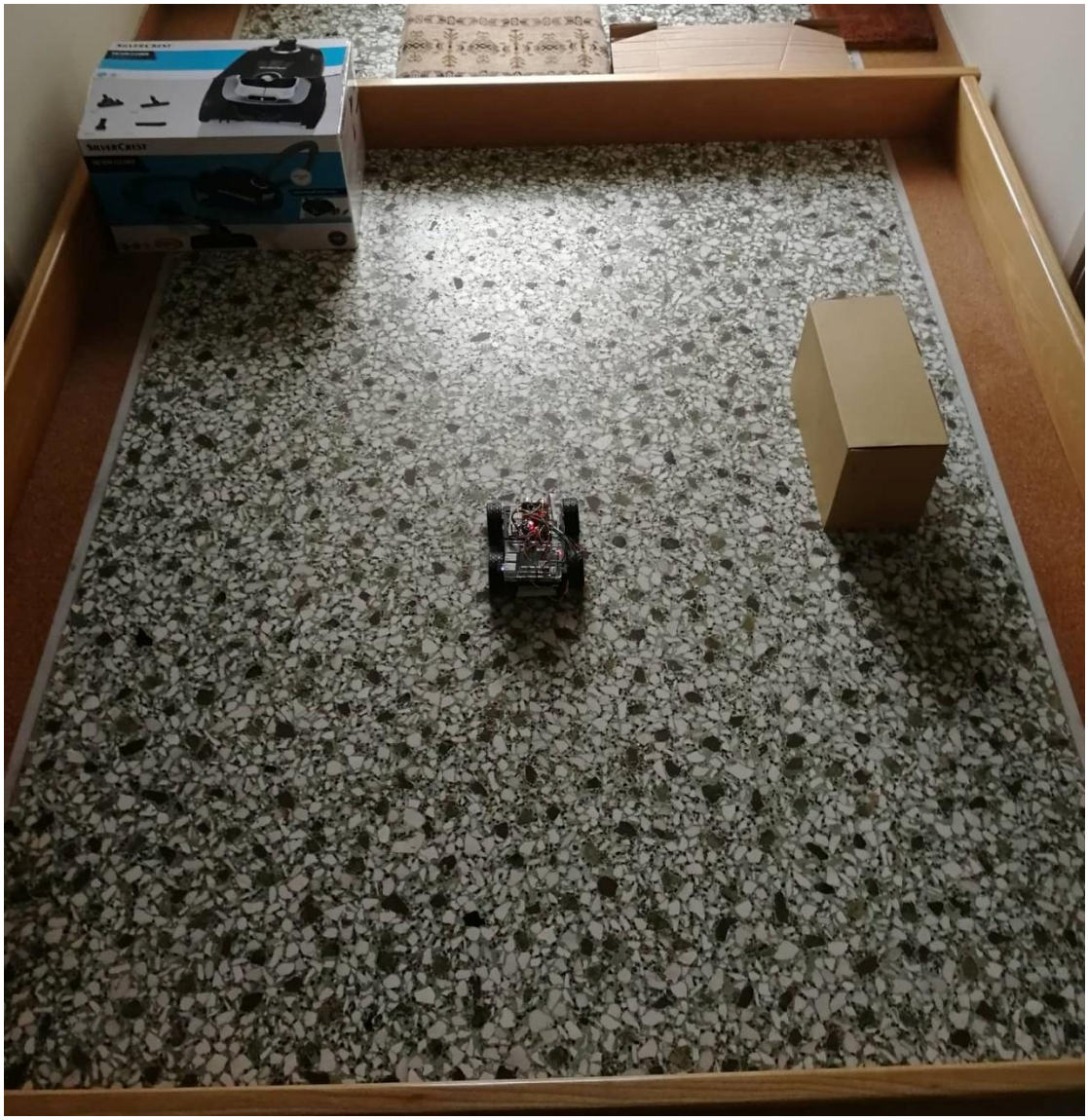
Στις επόμενες φωτογραφίες φαίνεται το αποτέλεσμα του πειράματος και για τις δύο περιπτώσεις χρήσης, είτε με χειρισμό από το χρήστη είτε αυτόνομα. Παρατηρούμε ότι το αποτέλεσμα του αυτόνομου πειράματος είναι σαφώς ελλιπές και κατώτερης απόδοσης από αυτό στο οποίο το σύστημα χειριζόταν ο χρήστης. Αυτό κατά κύριο λόγο, συμβαίνει γιατί το σύστημα δεν έχει στη διάθεση του τα κατάλληλα δεδομένα για την αποτελεσματική λειτουργία του. Επακολούθως και ο κώδικας που το ελέγχει είναι πιο απλός και άρα λιγότερο αποδοτικός, αφού λείπουν τα δεδομένα των αισθητήρων που θα τον εμπλούτιζαν.



Αυτοματοποιημένο Πείραμα



Πείραμα χειρισμού από χρήστη


SilverCrest

Προβλήματα, προτάσεις βελτίωσης, αξιολόγηση συστήματος και συμπεράσματα.

Στο κεφάλαιο αυτό θα γίνει μια αξιολόγηση του συστήματος, καθώς και παρουσίαση κάποιων βασικών περιορισμών και προβλημάτων του. Επίσης, θα παρουσιαστούν προτάσεις βελτίωσης και πως αυτές θα μπορούσαν να βοηθήσουν είτε στην ομαλότερη και αποδοτικότερη λειτουργία του, είτε στην κλιμάκωση του ώστε να καλύψει μεγαλύτερο εύρος εφαρμογής.

Α)Προβλήματα Hardware και τρόπος επίλυσης τους

Το πρώτο πρόβλημα το οποίο παρουσιάστηκε από την αρχή της υλοποίησης, αν και μικρό, ήταν το πρόβλημα του χώρου. Η μεταλλική κατασκευή παρείχε χώρο μόνο για τα μοτέρ στο εσωτερικό της, σε αντίθεση με άλλες μεγαλύτερες, οι οποίες παρέχουν χώρο και για τα εξαρτήματα, προστατεύοντάς τα με αυτό το τρόπο.

Το επόμενο πρόβλημα που προέκυψε και το οποίο είναι παράγωγο πρόβλημα της έλλειψης χώρου, αφορά τα εξαρτήματα. Μη έχοντας κάποια κάλυψη είναι εύκολο να κουνηθούν στις απότομες αλλαγές κατεύθυνσης του οχήματος. Αυτό έχει σαν αποτέλεσμα να χάνεται η επαφή στα καλώδια και άρα να δυσλειτουργεί το σύστημα.

Επόμενο πρόβλημα ήταν το cable management. Έπρεπε να βρεθεί τρόπος να λειτουργεί το σύστημα χωρίς να εμποδίζεται από τα καλώδια τα οποία μπλέκονταν στις ρόδες του αμαξιδίου, όπως αυτό κινούταν ή ακόμα και στα εμπόδια, που αυτό συναντούσε καθώς κινείται.

Τα τρία αυτά προβλήματα λύθηκαν με ταινία διπλής όψης. Κολλώντας τα εξαρτήματα μεταξύ τους και επάνω στην μεταλλική κατασκευή και με χρήση απλής ταινίας για την ομαδοποίηση των καλωδίων, το σύστημα πλέον είναι πιο σταθερό και λειτουργεί ομαλότερα.

Από τις πρώτες στιγμές λειτουργίας του συστήματος, έγινε φανερό ότι πρόβλημα αποτελούσε και η πηγή ενέργειας για τις ρόδες. Οι μπαταρίες ΑΑ που έρχονταν στη συσκευασία δεν παρείχαν την απαραίτητη τάση και ούτε είχαν κατάλληλη διάρκεια ζωής, για να λειτουργεί σωστά το ρομποτάκι. Εξαιτίας της φύσης των μπαταριών το σύστημα δυσκολευόταν να κινηθεί επάνω σε τραχείες επιφάνειες και η κίνηση των τροχών καθώς και η ταχύτητα τους εξαρτιόνταν από την τάση των μπαταριών. Προκειμένου να δοθεί λύση στο πρόβλημα αυτό υπήρξε αντικατάσταση των μπαταριών ΑΑ με μπαταρίες τύπου 18650. Οι μπαταρίες αυτές είναι σχεδιασμένες να παρέχουν σταθερή τάση μέχρι το τέλος της ενέργειας τους και η διάρκεια κάθε κύκλου φόρτισης είναι πολλαπλάσια της διάρκειας του προηγούμενου τύπου μπαταριών. Έτσι το αμαξίδιο κινείται σταθερά και ομαλά επάνω σε κάθε είδους επιφάνεια με διακυμάνσεις ανάλογες του υλικού στο οποίο κινείται.

Κατά την διάρκεια υλοποίησης και δοκιμής του αλγορίθμου στα διάφορα στάδια του, προέκυψε και το εξής πρόβλημα. Οι ρόδες δεν παρείχαν δυνατότητα μέτρησης απόστασης με αποτέλεσμα τη δημιουργία διαφόρων προβλημάτων. Όπως είδαμε, στην υλοποίηση του αλγορίθμου, γίνεται η χρήση χρόνου αντί για την απόσταση στο τρόπο που κινούνται οι ρόδες. Αυτό εγείρει διάφορα προβλήματα, με βασικότερο ότι σε διαφορετικές επιφάνειες χρειάζεται και διαφορετικός χρόνος, για να καλυφθεί η ίδια απόσταση. Επίσης,

όσο μεγαλώνει η διάρκεια της κίνησης, το αμαξίδιο επιταχύνει άρα είναι δύσκολο να υπολογιστεί σωστά ο χρόνος, όταν υπάρχουν τόσοι παράγοντες που το επηρεάζουν, όπως είναι οι εναλλαγές μεταξύ επιφανειών, το είδος της μπαταρίας, το επίπεδο φόρτισης της μπαταρίας, το βάρος που κουβαλά το αμαξίδιο κ.α. Αυτό έχει άμεσο αντίκτυπο στην απόδοση και αξιοπιστία του αλγορίθμου. Η σωστή μέτρηση απόστασης και η καταγραφή της είναι ο στόχος του πειράματος. Η απουσία σωστών μοτέρ ή κάποιου τρόπου καταγραφής απόστασης από κάποιο αισθητήρα κάνει το σύστημα αναξιόπιστο. Το πρόβλημα αυτό λύνεται με την αντικατάσταση των μοτέρ με άλλα βαθμονομημένα, Stepper Motor. Εναλλακτικά, προτείνεται η αγορά Wheel Encoder Kit. Το εξάρτημα αυτό μετρά τις περιστροφές των ροδών, μετά με απλούς υπολογισμούς γίνεται η μέτρηση της απόστασης. Φυσικά, υπάρχουν και άλλοι τρόποι, όπως η χρήση rotary encoder που μετατρέπει την γωνιακή ταχύτητα σε αναλογικό ή ψηφιακό σήμα αλλά είναι πιο περίπλοκη η λειτουργία τους ή έχουν μεγαλύτερο κόστος για να ληφθεί το ίδιο αποτέλεσμα με τις λύσεις που προτάθηκαν.

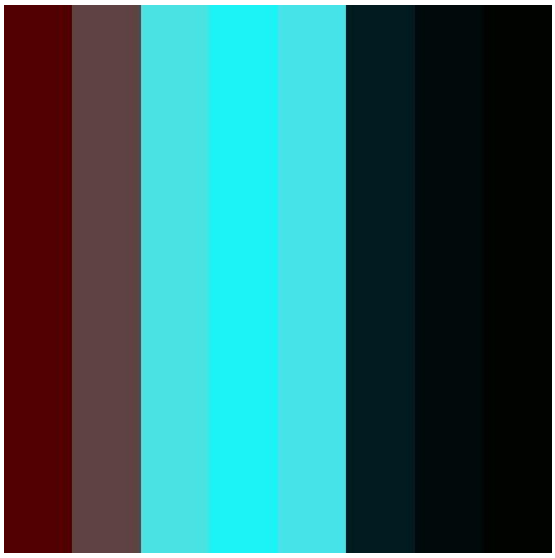
Σε επόμενο στάδιο ήταν απαραίτητη η εκμετάλλευση των δεδομένων του αισθητήρα. Αφού ξεπεράστηκε το πρόβλημα με το χαμηλής ποιότητας breadboard μέσω αναβάθμισης σε καλύτερης ποιότητας, τα δεδομένα του αισθητήρα μπορούσαν πλέον να χρησιμοποιηθούν. Κατά την επεξεργασία και χρήση τους όμως προέκυψε ένα πρόβλημα το οποίο αφορά τη φύση του αισθητήρα. Όπως αναφέρθηκε και σε προηγούμενο κεφάλαιο ο αισθητήρας λειτουργεί αξιοποιώντας το φαινόμενο Doppler. Αυτό έχει κάποιους φυσικούς περιορισμούς. Όταν η επιφάνεια που γίνεται η μέτρηση δεν είναι κάθετη με τον αισθητήρα, επειδή το σήμα κάνει πολλές αντανακλάσεις, αργεί να επιστρέψει στον αισθητήρα. Το αποτέλεσμα είναι να λαμβάνονται λάθος δεδομένα. Αυτό είναι ένα πρόβλημα, το οποίο με τον αισθητήρα αυτό δεν λύνεται είτε αλλάζοντας τον αλγόριθμο είτε προσθέτοντας επιπλέον αισθητήρες ίδιου είδους. Η λύση στο πρόβλημα αυτό δίνεται με την εισαγωγή άλλου είδους αισθητήρα. Συγκεκριμένα ένα LIDAR(Light Detection and Ranging) είναι το καταλληλότερο για την περίπτωση χρήσης που εξετάζεται. Ειδικά, αν επιλεχθεί λύση που σαρώνει περιστροφικά (360^{o}) η αξιοπιστία του αλγορίθμου ανεβαίνει πολύ. Το LIDAR χρησιμοποιώντας το φως μπορεί να κάνει χιλιάδες μετρήσεις το δευτερόλεπτο, με μεγάλη ακρίβεια και σε συνδυασμό με κάποια βιβλιοθήκη όπως αυτή του TensorFlow από την Google μπορεί και εύκολα να παρέχει ζωντανά αποτελέσματα μέσω GUI. Στα αρνητικά του LIDAR είναι το μέγεθος του αισθητήρα και το βάρος του συγκριτικά με τον ήδη υπάρχοντα HC-SR04 αλλά κυρίως το κόστος αγοράς του. Τέτοιου είδους συστήματα LIDAR σε συνδυασμό με άλλους αισθητήρες χρησιμοποιούνται και στα αυτόνομα οχήματα μεγάλων εταιρειών για εντοπισμό και αποφυγή εμποδίων, για ενημέρωση του οδηγού αλλά ακόμα και αυτόνομη οδήγηση.

Β)Προβλήματα Software και τρόπος επίλυσης τους

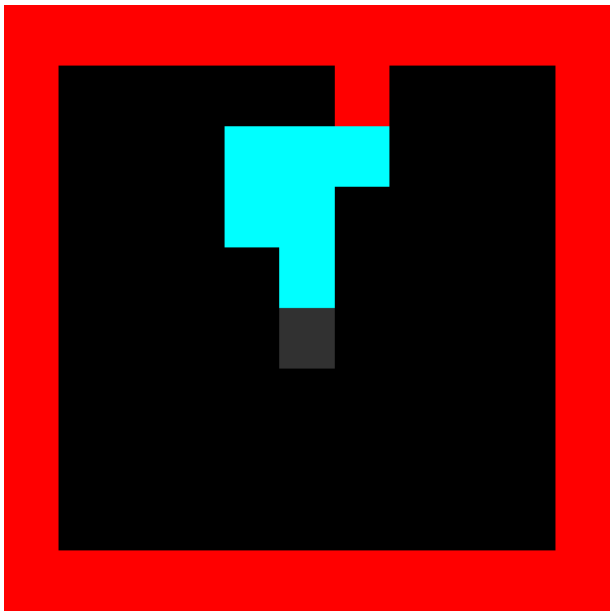
Σε επίπεδο λογισμικού δεν αντιμετωπίστηκαν σοβαρά προβλήματα. Ένα πρόβλημα το οποίο όμως είχε σαν αποτέλεσμα την καθυστέρηση της ολοκλήρωσης του αλγορίθμου, αφορούσε την έκδοση μιας βιβλιοθήκης. Βασική βιβλιοθήκη, όπως αναφέρθηκε και σε προηγούμενο κεφάλαιο ήταν η matplotlib. Δυστυχώς όμως, για ένα διάστημα δεν ήταν συμβατή με την έκδοση Rasbian, με αποτέλεσμα να καθυστερεί η ολοκλήρωση του αλγορίθμου. Παρότι υπήρχαν εναλλακτικές, καμία δεν προσέδιδε την απόδοση και ευχρηστία της προαναφερθείσας βιβλιοθήκης. Όλες υστερούσαν σε λειτουργικότητα και

απόδοση. Το πρόβλημα λύθηκε με δημοσιοποίηση έκδοσης συμβατής με την έκδοση Rasbian μετά από κάποιο διάστημα.

Ένα ακόμη βασικό πρόβλημα που αντιμετωπίστηκε προς το τέλος της υλοποίησης του προγράμματος ήταν η σωστή απόδοση των δεδομένων σε μορφή εικόνας κατανοητής για το χρήστη. Όταν η μετατροπή και εξαγωγή του πίνακα γινόταν σε jpeg αρχείο εικόνας, τότε λόγω του αλγορίθμου πίσω από το jpeg, ο οποίος επιλέγει τα δεδομένα βγάζοντας μέσους όρους για κάθε pixel, το τελικό αποτέλεσμα δεν είχε καμία σχέση με το μονοπάτι που ακολουθούσε το ρομποτάκι. Έτσι ήταν αδύνατο να το καταλάβει ο χρήστης.



Η εικόνα αυτή είναι ένα χαρακτηριστικό παράδειγμα του προβλήματος καθώς δεν θα έπρεπε να υπάρχει μετάβαση από το κόκκινο στο μπλε χρώμα και από το μπλε στο μαύρο αλλά θα έπρεπε να γίνεται με απόλυτο τρόπο όπως στο παρακάτω παράδειγμα.

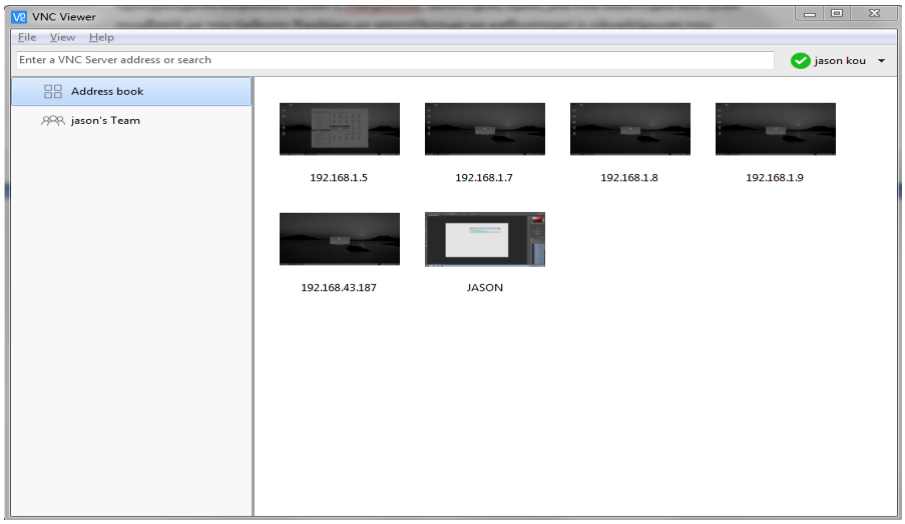


Ο τρόπος που αντιμετωπίστηκε το πρόβλημα αυτό ήταν με εξαγωγή των εικόνων σε μορφή png και όχι jpeg. Έτσι, απεικονίζεται ξεκάθαρα το αποτέλεσμα χρήσης και ο χρήστης μπορεί να καταλάβει το μονοπάτι που ακολουθήθηκε.

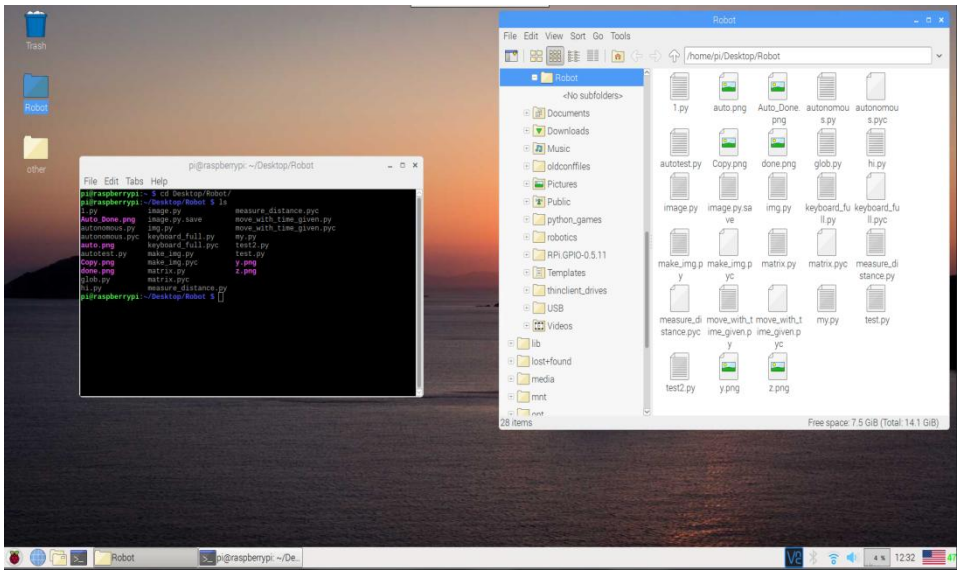
User Manual

Παρακάτω παρουσιάζεται ο οδηγός βασικών λειτουργιών και χρήσης του συστήματος.

1. Ξεκινάμε τη λειτουργία στο ρομποτάκι.
2. Ανοίγουμε το VNC viewer ή όποιο άλλο πρόγραμμα έχουμε επιλέξει και συνδεόμαστε στο ρομποτάκι απομακρυσμένα επιλέγοντας την IP.



3. Αφού πληκτρολογήσουμε το όνομα χρήστη και κωδικό για το ρομποτάκι μεταφερόμαστε στο περιβάλλον χρήσης του όπου και ανοίγουμε ένα terminal.
4. Γράφοντας την εντολή: cd Desktop/Robot μεταφερόμαστε στο φάκελο που βρίσκεται ο κώδικας. Τρέχοντας την εντολή: ls βλέπουμε το περιεχόμενο του φακέλου.



5. Τα αρχεία που θέλουμε είναι τα keyborad_full.py με το οποίο τον έλεγχο έχει ο χρήστης και το αρχείο autotest.py με το οποίο το όχημα λειτουργεί μόνο του.

6. Προκειμένου να τρέξουμε τον αλγόριθμο στο terminal πληκτρολογούμε την εντολή sudo python ____*όνομα αρχείου*___.py.
7. Για επεξεργασία του αρχείου γράφουμε sudo nano ____*όνομα αρχείου*____.py
8. Στη περίπτωση επιλογής του keyboard_full.py ο χρήστης έχει τις εξής επιλογές στη διάθεσή του.
    I. Πατώντας το πλήκτρο W πηγαίνει μπροστά.
    II. Πατώντας το πλήκτρο S πηγαίνει πίσω.
    III. Πατώντας το πλήκτρο A στρίβει αριστερά.
    IV. Πατώντας το πλήκτρο D στρίβει δεξιά.
    V. Πατώντας το πλήκτρο Q κάνει αριστερόστροφη περιστροφή.
    VI. Πατώντας το πλήκτρο E κάνει δεξιόστροφη περιστροφή.
    VII. Πατώντας το πλήκτρο P αποθηκεύει στιγμιότυπο του αλγόριθμου σε μορφή εικόνας Copy.png .
    VIII. Πατώντας το πλήκτρο M μετρά την απόσταση και την δείχνει στο χρήστη χωρίς να καταχωρείται αυτή στα δεδομένα.
    IX. Πατώντας το πλήκτρο Z τερματίζει τον αλγόριθμο και αποθηκεύει την όποια πρόοδο σε μορφή εικόνας done.png.
    X. Πατώντας κάποιο άλλο πλήκτρο βγαίνει προειδοποιητικό μήνυμα στο χρήστη ότι πάτησε λάθος πλήκτρο.
9. Στη περίπτωση επιλογής του autotest.py ο χρήστης τρέχει τον αυτοματοποιημένο αλγόριθμο και στο τέλος αποθηκεύονται τα δεδομένα σε αρχείο της μορφής Auto_Done.png.
10. Για να κλείσει το σύστημα ο χρήστης, κλείνει το terminal, επιλέγει τερματισμό από το μενού έναρξης και κλείνει το παράθυρο διαλόγου του VNC viewer καθώς και το VNC viewer.

Βιβλιογραφία

[1] Implementing Odometry and SLAM Algorithm by Patrick Butterly, Joshua Daly, Leigh Morrish

[2] Robot Localization and Map Construction Using Sonar Data by Vassilis Varveropoulos

[3] Mobile Robot Self-Localization without Explicit Landmarks by R. G. Brown and B. R. Donald

[4] BreezySLAM: A Simple, efficient, cross--platform Python package for Simultaneous Localization and Mapping by Suraj Baracharya

[5] Simultaneous localization and mapping (SLAM) with an autonomous robot

[6] Semi Autonomous Vehicle To Prevent Accident

[7] Robotics Enabling Autonomy in Challenging Environments by Ioannis Rekleitis

[8] Vehicle Control Using Raspberrypi and Image Processing

[9] Sonar and Pi Based Aid for Blind

[10] Way-finding Electronic Bracelet for Visually Impaired People

[11] Implementation of Line Tracking Algorithm using Raspberry Pi in Marine Environment

[12] Advanced Vehicle Monitoring and Tracking System based on Raspberry Pi

[13] Blind Navigation Proposal using SONAR

[14] Διάφορα threads από τα επίσημα forum του Raspberry PI https://www.raspberrypi.org/forums/

[15] Διάφορα threads και προτάσεις χρηστών από https://stackoverflow.com/

[16] Python Documentation https://docs.python.org/3/

[17] Numpy Documentation https://www.numpy.org/

[18] Matplotlib Documentation https://matplotlib.org/

[19] Instructions on using GPIO pins https://raspi.tv/2013/rpi-gpio-basics-5-setting-up-and-using-outputs-with-rpi-gpio