ΠΑΝΕΠΙΣΤΗΜΙΟ ΔΥΤΙΚΗΣ ΑΤΤΙΚΗΣ

ΣΧΟΛΗ ΜΗΧΑΝΙΚΩΝ

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

ΑΥΤΟΝΟΜΟ ΟΧΗΜΑ - ΑΥΤΟΜΑΤΟ PARKING ΜΕ ΤΗΝ ΧΡΗΣΗ RASPBERRY PI

Αναγνωστάκης Αντώνιος
Βασιλόπουλος Χρήστος

Εισηγητής: ΠΑΡΗΣ ΜΑΣΤΟΡΟΚΩΣΤΑΣ, Καθηγητής

Αθήνα, Ιούλιος 2020

ΠΡΟΛΟΓΟΣ

Η εργασία αυτή αποτελεί την πτυχιακή μας εργασία και πραγματοποιήθηκε στα πλαίσια του προγράμματος σπουδών του Πανεπιστημίου Δυτικής Αττικής στο τμήμα των Μηχανικών Ηλεκτρονικών Υπολογιστικών Συστημάτων Τ.Ε με επιβλέπων καθηγητή κ. Πάρη Μαστοροκώστα.

Η εργασία αυτή έχει ως σκοπό την ανάπτυξη συστημάτων για αυτοματοποιημένο έλεγχο της κίνησης ενός αυτοκινήτου, κάτι το οποίο θα συμβάλει στη διευκόλυνση της χρήσης του καθώς και σε τομείς που θα απέτρεπαν τροχαία ατυχήματα με την κατάλληλη εξέλιξη και υποδομή.

Σε αυτό το σημείο θα θέλαμε να ευχαριστήσουμε θερμά τον καθηγητή κ. Πάρη Μαστοροκώστα τόσο για την ανάθεση του θέματος όσο και για την βοήθειά του.

Επίσης θα θέλαμε να ευχαριστήσουμε τους συμφοιτητές μας Γιώργο και Πέτρο για την πρόθυμη και άνευ όρων βοήθειά τους σε δυσκολίες που προέκυψαν.

Τέλος θα θέλαμε να ευχαριστήσουμε τους γονείς μας για την στήριξη και συμπαράσταση σε όλα τα χρόνια των σπουδών μας και τους φίλους και συμφοιτητές μας για την στήριξη που μας παρείχαν.

ΠΕΡΙΕΧΟΜΕΝΑ

ΠΕΡΙΛΗΨΗ

Ένα σύστημα αυτοματοποιημένης οδήγησης είναι ένα σύστημα εντός τους οχήματος που αναλαμβάνει όλες τις λειτουργίες που είναι απαραίτητες για την οδήγηση του οχήματος σε πραγματικό χρόνο χωρίς την ανάγκη παρέμβασης ανθρώπινου χειριστή.

Οι πόλεις αυτοματοποιούνται με σταθερό ρυθμό και περιθώρια αμφιβολίας δεν υπάρχουν. Αυτόνομα αυτοκίνητα και αυτοκινούμενες δημόσιες συγκοινωνίες λειτουργούν ήδη σε διάφορες γωνίες του κόσμου. Σεούλ, Τόκυο, Σιγκαπούρη, Ντουμπάι, Λονδίνο και Σαν Φρανσίσκο και πολλές ακόμα φιλοξενούν αυτοματοποιημένες δομές σε επίπεδο αυτοκινήτων, υπηρεσιών ταξί και μέσων μαζικής μεταφοράς.

Αντικείμενο της παρούσας πτυχιακής εργασίας είναι να σας παρουσιάσει τον τρόπο ελέγχου και τον χειρισμό ενός ρομποτικού αμαξιδίου από τον ηλεκτρονικό υπολογιστή(windows). Επίσης θα γίνει αναφορά στα επιμέρους εξαρτήματα καθώς και στο λογισμικό που χρησιμοποιήθηκε για την επίτευξη των αυτοματοποιημένων ενεργειών του ρομποτικού αμαξιδίου όπως το αυτόματο παρκάρισμα και ο ασύρματος έλεγχος και η παρακολούθηση του μέσω του διαδικτύου.

Τα υλικά που χρησιμοποιήθηκαν είναι ένα raspberry pi, κινητήρες μετάδοσης κίνησης, ένας web server για την μεταφορά της εικόνας και των εντολών, ένας μικροελεγκτής για τον έλεγχο των κινητήρων καθώς και ultrasonic αισθητήρες.

Η γλώσσα προγραμματισμού που χρησιμοποιήθηκε για τον έλεγχο και τον προγραμματισμό του ρομποτικού αμαξιού είναι η python καθώς και η php για την σύνδεση μέσω διαδικτύου.

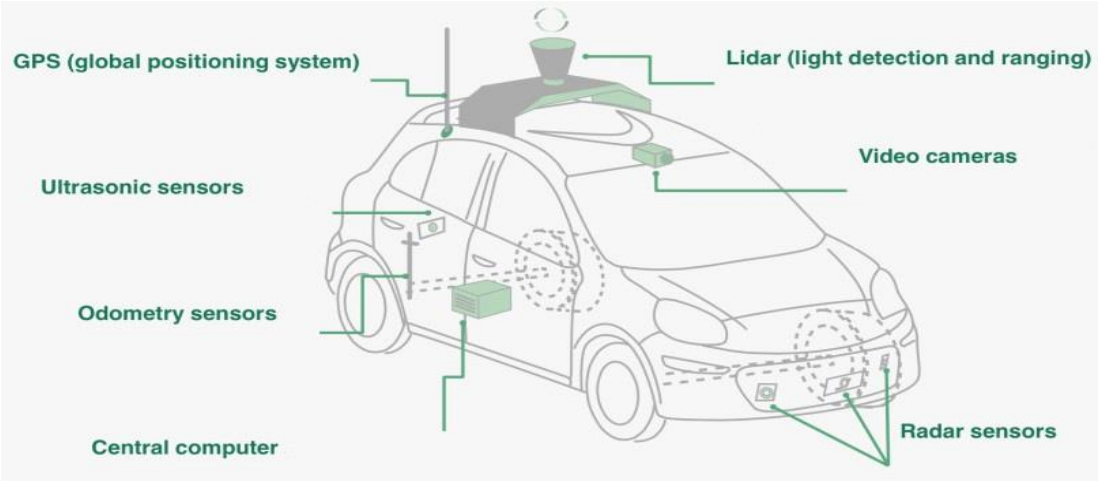
ΚΕΦΑΛΑΙΟ 1^{Ο} : ΓΕΝΙΚΕΣ ΠΛΗΡΟΦΟΡΙΕΣ ΑΥΤΟΝΟΜΟΥ ΟΧΗΜΑΤΟΣ

1.1 Τι είναι ένα αυτόνομο όχημα

Αυτόνομο ονομάζουμε, ένα όχημα που είναι ικανό να ανιχνεύει το περιβάλλον και να περιφέρεται σε αυτό χωρίς την ανθρώπινη παρέμβαση.

Τα αυτόνομα οχήματα μπορούν, χρησιμοποιώντας ραντάρ, Lidar, GPS, και κάμερες με την βοήθεια του υπολογιστή, να ανιχνεύσουν το περιβάλλον. Προηγμένα συστήματα ελέγχου ερμηνεύουν αισθητηριακές πληροφορίες για τον εντοπισμό κατάλληλων μονοπατιών πλοήγησης, καθώς και τα εμπόδια και τις σχετικές πινακίδες. Τα αυτόνομα αυτοκίνητα διαθέτουν πλέον συστήματα ελέγχου που αναλύουν τα δεδομένα που λαμβάνουν από τους αισθητήρες και διακρίνουν τις διαφοροποιήσεις μεταξύ των υπόλοιπων αυτοκινήτων στο δρόμο, με σκοπό το σχεδιασμό μιας διαδρομής στον επιθυμητό προορισμό.

Μερικά συστήματα, πρόωρα της αυτόνομης κίνησης, χρονολογούνται από τη δεκαετία του 1920 και του '30. Τα πρώτα αυτάρκης (και, ως εκ τούτου, πραγματικά αυτόνομα) αυτοκίνητα έκαναν την εμφάνισή τους στη δεκαετία του 1980, με τα προγράμματα ‘Navlab’ και ‘ALV’ στο Πανεπιστήμιο Carnegie Mellon το 1984 και η Mercedes-Benz και το Πανεπιστήμιο Bundeswehr στο Μόναχο με τα προγράμματα ‘Eureka’ και ‘Prometheus’ το 1987. Από τότε, πολλές μεγάλες εταιρείες και ερευνητικοί οργανισμοί έχουν αναπτύξει και εργάζονται σε πρωτότυπα αυτόνομα οχήματα.



1.2 Επίπεδα αυτοματισμού

Επίπεδο 0 – Κανένας αυτοματισμός

Ο οδηγός έχει τον απόλυτο έλεγχο του οχήματος ακόμα και όταν δέχεται προειδοποιητικά μηνύματα από συστήματα ασφαλείας του οχήματος.

Επίπεδο 1 – Συγκεκριμένες λειτουργίες αυτοματισμού

Το όχημα έχει την δυνατότητα αυτόματα να εναλλάσσει την ταχύτητά του επιταχύνοντας/επιβραδύνοντας ανάλογα με τις συνθήκες που εκλαμβάνει από το οδικό περιβάλλον. Ο οδηγός έχει τον γενικό έλεγχο για όλες τις άλλες λειτουργίες της οδήγησης.

Επίπεδο 2 – Συνδυασμός λειτουργιών αυτοματισμού

Το όχημα έχει την δυνατότητα με περισσότερα από ένα συστήματα υποστήριξης της οδήγησης να επεμβαίνει αυτόματα επιταχύνοντας/επιβραδύνοντας και αλλάζοντας την ταχύτητα ανάλογα με τις συνθήκες που εκλαμβάνει από το οδικό περιβάλλον. Ο οδηγός έχει ακόμα τον γενικό έλεγχο για όλες τις άλλες λειτουργίες της οδήγησης.

Επίπεδο 3 – Περιορισμένη αυτόνομη οδήγηση

Το όχημα λειτουργεί πλήρως με ένα αυτόματο σύστημα οδήγησης με την προσδοκία ότι ο οδηγός θα επέμβει σε περίπτωση που χρειαστεί. Ο οδηγός δεν έχει συνεχώς τον έλεγχο του αυτοκινήτου αλλά μπορεί να επέμβει.

Επίπεδο 4 – Πλήρως αυτόνομη οδήγηση εκτός ειδικών συνθηκών

Το όχημα είναι σχεδιασμένο να λειτουργεί πλήρως αυτόνομα χωρίς τη βοήθεια του οδηγού. Σε ειδικές καιρικές συνθήκες και ειδικές συνθήκες κυκλοφορίας ο οδηγός πρέπει να ανακτήσει τον έλεγχο.

Επίπεδο 5 – Πλήρως αυτόνομη οδήγηση

Το όχημα είναι σχεδιασμένο να λειτουργεί πλήρως αυτόνομα χωρίς τη βοήθεια του οδηγού σε όλες τις περιπτώσεις. Ο οδηγός απλά προγραμματίζει τον προορισμό της διαδρομής.

1.3 Πλεονεκτήματα αυτόνομου οχήματος

- Βελτίωση οδικής ασφάλειας
- Πλήρης τήρηση του κώδικα οδικής κυκλοφορίας
- Βέλτιστος χρόνος αντίδρασης σε μη αναμενόμενο συμβάν
- Δεν επηρεάζει η συμπεριφορά του οδηγού την οδήγηση
- Χαμηλή κατανάλωση καυσίμου
- Χαμηλές εκπομπές ρύπων στο περιβάλλον
- Δυνατότητα μετακίνησης ανθρώπων και μεταφοράς εμπορευμάτων χωρίς συνοδό.

ΚΕΦΑΛΑΙΟ 2^{Ο} : RASPBERRY Pi

1.1 Τι είναι το Raspberry pi

Το Raspberry pi είναι ένας υπολογιστής με μέγεθος πιστωτικής κάρτας.

Παρά τον μικρό όγκο του στη μεγαλύτερη έκδοσή του διαθέτει τετραπύρηνο επεξεργαστή στα 1200MHz, διπύρηνη κάρτα γραφικών, 1GB Ram, τέσσερις θύρες USB, έξοδο HDMI, έχει τη δυνατότητα τροφοδοσίας μέσω micro USB και διαθέτει 40 pins γενικής χρήσης για σύνδεση με άλλα ηλεκτρονικά και περιφερειακά.

Συνδέοντας το Raspberry pi με μία οθόνη και παράλληλα προσθέσουμε ένα πληκτρολόγιο και ένα ποντίκι μπορούμε να έχουμε έναν ολοκληρωμένο υπολογιστή, ο οποίος όμως θα υποστηρίζει συγκεκριμένες διανομές Linux όπως το Raspbian, Ubuntu και RISC OS.

Το Raspberry pi μπορούμε να το χρησιμοποιήσουμε με διάφορους τρόπους. Ένας από αυτούς είναι να παίζουμε ταινίες στην τηλεόραση μας χάρης του OSMC, ενός λογισμικού ανοιχτού κώδικα(open source) το οποίο έχει λειτουργίες ενός Media Center. Μπορούμε επίσης να το χρησιμοποιήσουμε και για παιχνίδια φορτώνοντας του emulators δίνοντας μας την δυνατότητα να απολαύσουμε παλιού τύπου gaming όπως τα Arcade.

Τέλος μία άλλη εφαρμογή του είναι ότι άμα συνδέσουμε έναν εξωτερικό δίσκο πάνω του μπορούμε να το μετατρέψουμε σε ένα σύστημα αποκλειστικά για να κατεβάζουμε torrents χωρίς χρονικό περιορισμό καθώς έχει λιγότερη κατανάλωση(4W) από έναν κοινό υπολογιστή.

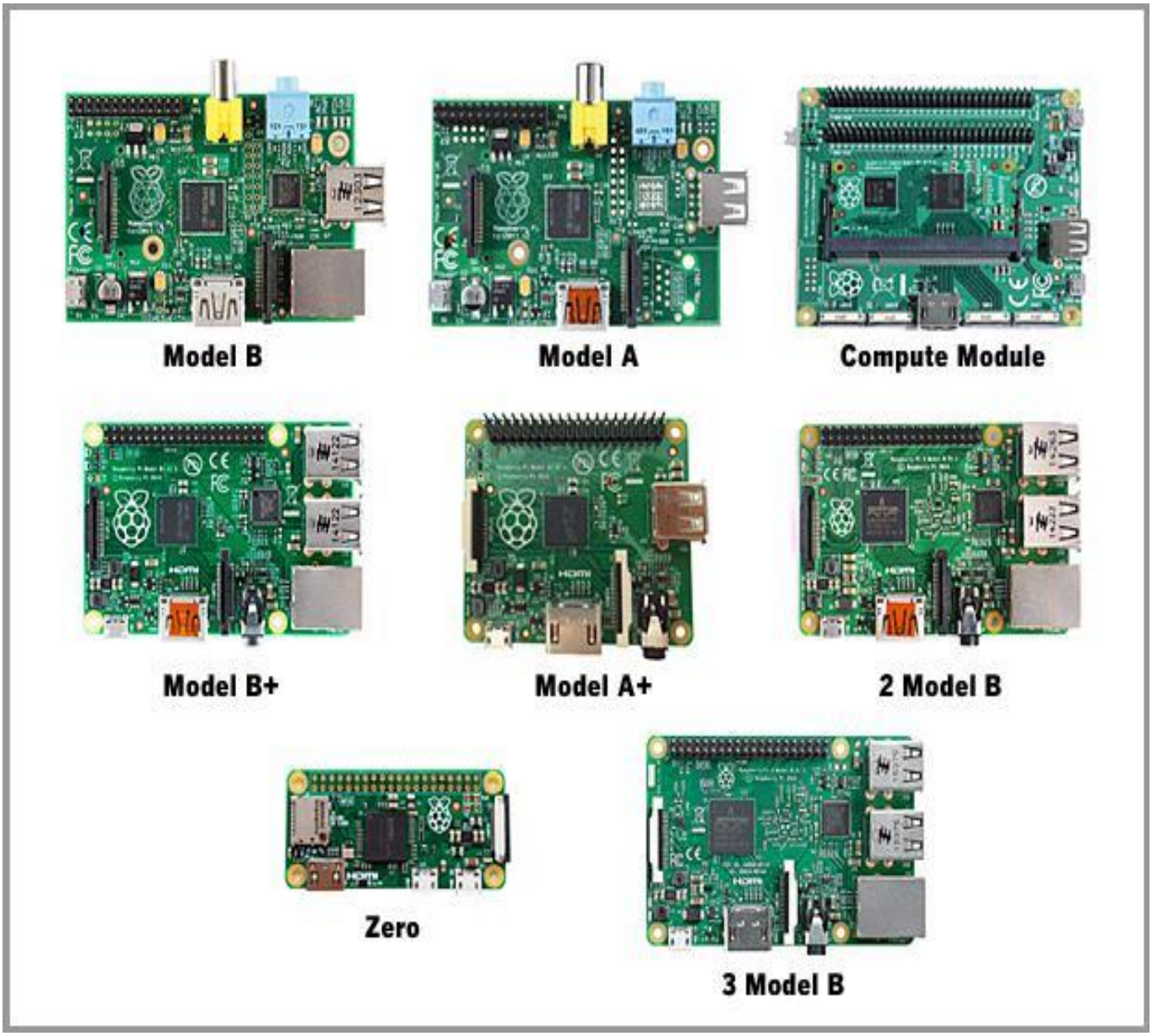
1.2 Ιστορική αναδρομή του Raspberry Pi

Στα μέσα της δεκαετίας του ’00 στο πανεπιστήμιο του Cambridge μια ομάδα εργαζομένων στο τμήμα της Πληροφορικής ανησυχούσε για το γεγονός ότι οι φοιτητές δεν έδιναν ιδιαίτερη σημασία στο συγκεκριμένο τμήμα καθώς και ότι οι γνώσεις τους γύρω από αυτό ήταν ελλιπής.

Έτσι η συγκεκριμένη ομάδα του τμήματος σκέφτηκε ότι ένας πολύ μικρός και φθηνός υπολογιστής θα τους κινούσε το ενδιαφέρον. Το πρώτο Raspberry pi κυκλοφόρησε το 2011 με τα Model A, A+ ΚΑΙ B. Διέθεταν επεξεργαστή ARMv6k στα 700MHz με 256MB RAM και κάρτα γραφικών Broadcom VideoCore IV. Το Raspberry Pi Model B μέχρι το 2013 πούλησε πάνω από 2 εκατομμύρια κομμάτια.

Σειρά πήραν τα Model B rev 2 και Model B+ με 512MB RAM. Στα τέλη του 2014 οι πωλήσεις έφτασαν τα 4 εκατομμύρια κομμάτια. Στις αρχές του 2015 βγήκε στη κυκλοφορία το Generation 2 Model B με 1GB RAM, με τον τετραπύρινο επεξεργαστή Cortex-A7 και μια διπύρηνη κάρτα γραφικών Broadcom VideoCore IV.

Στα τέλη του 2015 κυκλοφόρησε το Raspberry Pi Zero το οποίο ήταν ακόμα πιο μικρό από το Raspberry Pi με 512MB RAM και τον επεξεργαστή ARM1176JZF-S στα 1000MHZ. Τον Φεβρουάριο του 2016 βγαίνει στη κυκλοφορία το Raspberry Pi Generation 3 Model B με καλύτερο και ταχύτερο επεξεργαστή, τον ARM Cortex-A53 στα 1200MHz, 1GB RAM και κάρτα γραφικών Broadcom VideoCore IV στα 250MHz.


Model B
Model A
Compute Module
Model B+
Model A+
2 Model B
Zero
3 Model B

1.3 Περιφερειακά του Raspberry Pi

i. **Motor Controller L298n**

Κάθε ηλεκτροκινητήρας πρέπει να έχει κάποιο είδος του ελεγκτή. Ο ελεγκτής του κινητήρα έχει διαφορετικά χαρακτηριστικά και πολυπλοκότητα, ανάλογα με την εργασία που ο κινητήρας θα εκτελεί. Η απλούστερη περίπτωση είναι ένας διακόπτης για τη σύνδεση ενός κινητήρα σε μια πηγή ρεύματος, όπως σε μικρές συσκευές ή ηλεκτρικά εργαλεία. Ο διακόπτης μπορεί να λειτουργεί χειροκίνητα ή μπορεί να είναι ένα ρελέ στον οποίο συνδέεται με κάποια μορφή αισθητήρα για να ξεκινήσει αυτόματα και να σταματήσει το μοτέρ.

Ο διακόπτης μπορεί να έχει πολλές θέσεις για να επιλέξουμε διαφορετικές συνδέσεις του κινητήρα. Αυτό μπορεί να επιτρέψει την εκκίνηση μειωμένης τάσης του κινητήρα, αντιστρέφοντας τον έλεγχο ή την επιλογή πολλαπλών ταχυτήτων . Μικροί ηλεκτροκινητήρες μπορεί να έχουν ενσωματωμένες συσκευές υπερφόρτωσης για να ανοίξει αυτόματα το κύκλωμα στην υπερφόρτωση. Μεγαλύτεροι κινητήρες έχουν ένα προστατευτικό ρελέ υπερφόρτωσης ή ρελέ ανίχνευσης θερμοκρασίας στον ελεγκτή και ασφάλειες ή διακόπτες κυκλώματος για πάνω από την τρέχουσα τιμή προστασία.

Πιο πολύπλοκοι ελεγκτές κινητήρα μπορεί να χρησιμοποιηθούν για να ελέγχουν με ακρίβεια την ταχύτητα και ροπή του συνδεδεμένου κινητήρα (ή κινητήρων) και μπορεί να είναι μέρος των συστημάτων ελέγχου κλειστού βρόχου για την ακριβή τοποθέτηση ενός κινούμενου μηχανήματος. Όταν συνδέσουμε έναν κινητήρα dc σε { + } πόλο από την μια και σε { - } πόλο από την άλλη , ο κινητήρας γυρίζει μια προς μια συγκεκριμένη κατεύθυνση . Όταν αλλάξουμε τα καλώδια με τους ακροδέκτες , ο κινητήρας γυρίζει στην άλλη κατεύθυνση . Ηλεκτρονικά μπορούμε να κάνουμε αλλαγή πολικότητας χρησιμοποιώντας κυκλώματα H - Bridge .

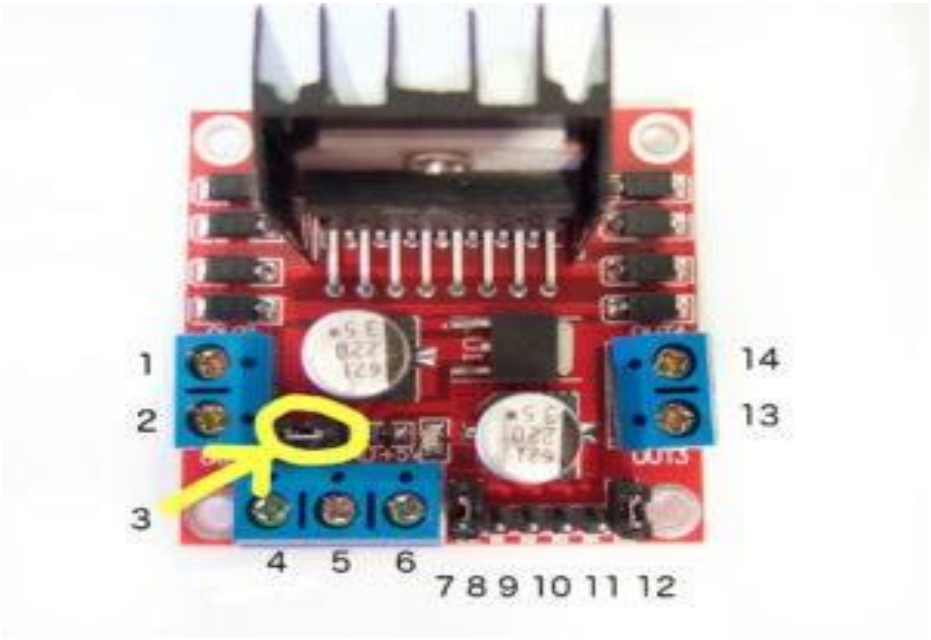
Ένα τέτοιου είδους κύκλωμα είναι ο ελεγκτής κινητήρα L298N.

Ο ελεγκτής L298N είναι μια απλή μονάδα με διπλό κύκλωμα H-Bridge. Αυτό σημαίνει ότι έχει τη δυνατότητα να ελέγχει δύο κινητήρες σε κάθε κατεύθυνση.

Τα χαρακτηριστικά του L298n είναι:

- Driver: L298N Dual H Bridge DC Motor Driver IC
- Motor Power Supply Vs: +5 V to +35 V
- Max average current: 2A
- Peak current Io: 3A
- Logic Level Power Vss: +5 V ~ +7 V (Onboard 5V Regulator can be used if Motor Power is > 7.0V)
- Logic level power: 0 - 36mA
- Logic signal input range: [Low:-0.3V ≤ Vin ≤ 1.5V] [High: 2.3V ≤ Vin ≤ Vss]
- Maximum power dissipation: 20W (when the temperature T = 75 ℃)

<u>Συνδεσμολογία L298n:</u>



1. DC motor 1 “+” or stepper motor A+
2. DC motor 1 “-” or stepper motor A-
3. 12V jumper – **remove this if using a supply voltage greater than 12V DC**. This enables power to the onboard 5V regulator
4. Connect your motor supply voltage here, maximum of 35V DC. Remove 12V jumper if >12V DC
5. GND
6. 5V output if 12V jumper in place
7. DC motor 1 enable jumper. Leave this in place when using a stepper motor. Connect to PWM output for DC motor speed control.
8. IN1
9. IN2
10. IN3
11. IN4
12. DC motor 2 enable jumper
13. DC motor 2 “+” or stepper motor B+
14. DC motor 2 “-” or stepper motor B-

ii. **<u>DC Κινητήρας</u>**

Ένας DC κινητήρας είναι μία τάξη των ηλεκτρικών μηχανών που μετατρέπει την συνεχής ηλεκτρική ενέργεια σε μηχανική ενέργεια. Οι πιο κοινοί τύποι επικαλούνται τις δυνάμεις που παράγονται από τα μαγνητικά πεδία. Σχεδόν όλοι οι τυπικοί κινητήρες συνεχούς ρεύματος έχουν κάποιο εσωτερικό μηχανισμό, είτε ηλεκτρομηχανικό ή ηλεκτρονικό, που μπορεί να αλλάξει περιοδικά την κατεύθυνση της ροής του ρεύματος σε ένα μέρος του κινητήρα. Οι περισσότεροι τύποι παράγουν περιστροφική κίνηση. Ένας γραμμικός κινητήρας παράγει απευθείας δύναμη και κίνηση σε μια ευθεία γραμμή.

Ο DC κινητήρας ήταν ο πρώτος τύπος χρησιμοποιήθηκε ευρέως, δεδομένου ότι θα μπορούσε να τροφοδοτείται από τα υπάρχοντα συστήματα διανομής ισχύος ρεύματος. Η ταχύτητα ενός κινητήρα συνεχούς ρεύματος μπορεί να ελεγχθεί σε μεγάλο εύρος, χρησιμοποιώντας είτε μια μεταβλητή τάση τροφοδοσίας ή με την αλλαγή της δύναμης του ρεύματος σε περιελίξεις τομέα του. Μικροί κινητήρες συνεχούς ρεύματος χρησιμοποιούνται σε εργαλεία, παιχνίδια και συσκευές. Οι μεγαλύτεροι κινητήρες συνεχούς

ρεύματος χρησιμοποιούνται στην κίνηση ηλεκτρικών οχημάτων, ασανσέρ και ανελκυστήρων.

Ο κινητήρας DC brushless χρησιμοποιεί εναν ή περισσότερες μόνιμους μαγνήτες στο ρότορα και ηλεκτρομαγνήτες στο περίβλημα του κινητήρα για τον στάτορα. Ένας ελεγκτής κινητήρα μετατρέπει DC σε AC. Αυτό το σχέδιο είναι μηχανικά απλούστερο από εκείνο του brushed κινητήρα διότι εξαλείφει την επιπλοκή της μεταφοράς ισχύος από το εξωτερικό του κινητήρα για την περιστροφή του ρότορα.

Τα χαρακτηριστικά του DC κινητήρα είναι:

- Motor rotating speed(3V):125
- Wheel rotating speed(3V):48
- Voltage Range:3- 6v
- No-load Current(6V): 70mA
- Torque(6V): 5.5Kg*cm
- Size:70.5mm×27mm×23mm
- Weigh:about 40g

iii. **Σερβοκινητήρας**

Ο σερβοκινητήρας είναι ένα συγκεκριμένο είδος κινητήρα που συνδυάζεται με ένα περιστροφικό κωδικοποιητή ή ένα ποτενσιόμετρο για να σχηματίσουν ένα σερβομηχανισμό. Αυτός η διάταξη μπορεί με τη σειρά της να αποτελεί τμήμα ενός άλλου σερβομηχανισμού. Ένα ποτενσιόμετρο παρέχει ένα απλό αναλογικό σήμα για να δείξει τη θέση, ενώ ένας κωδικοποιητής παρέχει θέση και συνήθως επιταχύνει την ανάδραση, η οποία με τη χρήση ενός ελεγκτή PID επιτρέπει τον ακριβέστερο έλεγχο της θέσης και επομένως ταχύτερη επίτευξη μιας σταθερής θέσης (για μια δεδομένη ισχύ κινητήρα) .

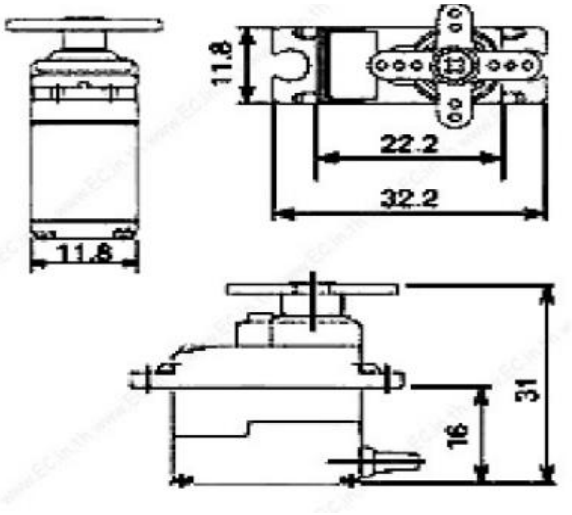
Οι σερβοκινητήρες χρησιμοποιούνται τόσο για υψηλού επιπέδου όσο και για χαμηλού επιπέδου εφαρμογές. Ο όρος σερβοκινητήρας αναφέρεται γενικά σε ένα high-end βιομηχανικό στοιχείο, ενώ ο όρος σέρβο πιο συχνά χρησιμοποιείται για να περιγράψει τις φθηνές συσκευές που χρησιμοποιούν ένα ποτενσιόμετρο. Βηματικά μοτέρ δεν θεωρούνται σερβοκινητήρες, αν και αυτοί χρησιμοποιούνται για την κατασκευή μεγαλύτερων σερβομηχανισμών.

Τα RC servos χρησιμοποιούνται για να παρέχουν ενεργοποίηση για τα διάφορα μηχανικά συστήματα, όπως το σύστημα διεύθυνσης του αυτοκινήτου, τις επιφάνειες ελέγχου σε ένα αεροπλάνο, ή το πηδάλιο του σκάφους. Λόγω των προσιτών τιμών τους, την αξιοπιστία και την απλότητα του ελέγχου από μικροεπεξεργαστές, χρησιμοποιούνται συχνά σε μικρής κλίμακας εφαρμογές ρομποτικής. Ένα συνηθισμένος δέκτης RC (ή ένας μικροελεγκτής) στέλνει διαμόρφωση εύρους παλμού (PWM) σήμα στον σέρβο. Τα ηλεκτρονικά μέσα στο σέρβο μεταφράζουν το πλάτος του παλμού σε μία θέση. Όταν στον σέρβο έχει δοθεί εντολή να περιστρέφεται, ο κινητήρας τροφοδοτείται μέχρι το ποτενσιόμετρο φτάσει την τιμή που αντιστοιχεί στην εντολή θέση.

Στο ρομποτικό αμαξίδιό μας χρησιμοποιήσαμε τον σερβοκινητηρα S99 ο οποίος είναι μικρός και ελαφρύς με υψηλή ισχύ εξόδου . Ο σέρβο μπορεί να περιστραφεί περίπου 180 μοίρες (90 σε κάθε κατεύθυνση) .

Τα χαρακτηριστικά του σερβοκινητήρα είναι:

- Weight: 9 g
- Dimension: 22.2 x 11.8 x 31 mm approx.
- Stall torque: 1.8 kgf cm
- Operating speed: 0.1 s/60 degree
- Operating voltage: 4.8 V (~5V)
- Dead band width: 10 μs
- Temperature range: 0 ºC – 55 ºC



iv. **Αισθητήρας Υπερύθρων**

Τα περισσότερα ρομπότ του σήμερα είναι σχεδόν κουφά και τυφλά. Οι αισθητήρες μπορούν να παρέχουν κάποια περιορισμένη ανάδραση για το ρομπότ έτσι ώστε να μπορεί να εκτελέσει την εργασία του. Συγκρίνοντας τις αισθήσεις και τις ικανότητες του με τα πιο απλά πράγματα διαβίωσης, τα ρομπότ έχουν ένα πολύ μακρύ δρόμο να διανύσουν.

Ο αισθητήρας για να αναγνωρίσει την κατάσταση του κόσμου γύρω του στέλνει πληροφορίες, με τη μορφή ηλεκτρονικών σημάτων πίσω σε έναν ελεγκτή.

Ο αισθητήρας υπερήχων βασίζεται στην αρχή του χρόνου-πτήσης. Ο πομπός παράγει μια δέσμη ήχου, που ταξιδεύει μακριά από την πηγή, και, αν συναντήσει εμπόδια, αντανακλάται από αυτά και επιστρέφει στο δέκτη (μικρόφωνο). Το ποσό του χρόνου που χρειάζεται για τη δέσμη του ήχου για να έρθει πίσω παρακολουθείται (από την έναρξη με ένα χρονόμετρο όταν η δέσμη παράγεται, και σταματώντας όταν ο ανακλώμενος ήχος επιστρέψει), και χρησιμοποιείται για να υπολογίσει την απόσταση που ο ήχος ταξίδεψε.

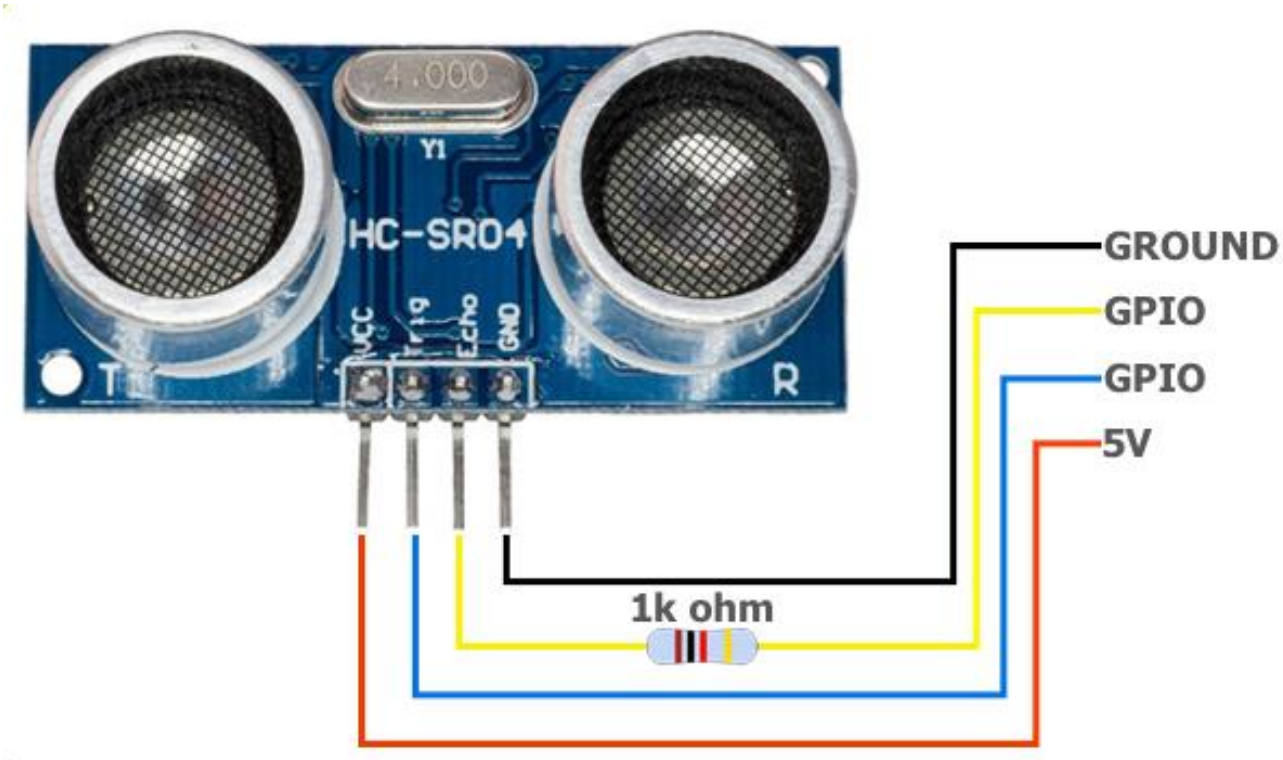
Οι αισθητήρες υπερήχων είναι ευρέως διαθέσιμοι και σχετικά φθηνοί, κυμαίνονται από 2 ευρώ έως 40 ευρώ, ανάλογα με την επιθυμητή κλίμακα. Κατά μέσο όρο το μέγιστο εύρος ενός αισθητήρα υπερήχων είναι μεταξύ 4 και 6 μέτρων. Σε αντίθεση με τις υπέρυθρες ή τους λέιζερ αισθητήρες, οι αισθητήρες υπερήχων έχουν επίσης μια ελάχιστη απόσταση ανίχνευσης, Αυτό οφείλεται στο γεγονός ότι οι μετρήσεις απόστασης γίνονται με βάση την ταχύτητα του ήχου, και σε πολύ μικρές αποστάσεις ο ήχος ταξιδεύει και γυρνάει πίσω πιο γρήγορα από ότι το κύκλωμα μπορεί να ανταποκριθεί. Αυτή η ελάχιστη απόσταση θα ποικίλει από αισθητήρα σε αισθητήρα, αλλά τυπικά είναι περίπου 2 έως 5 εκατοστά.

Ο συγκεκριμένος αισθητήρας που χρησιμοποιήσαμε είναι ο HC-SR04 και έχει 4 ακροδέκτες

VCC – +5V τροφοδοσία, TRIG – Trigger Pin , ECHO – Echo Pin , GND – γείωση

Η συνδεσμολογία του HC-SR04 με το Raspberry Pi γίνεται ως εξής:

Ο ακροδέκτης GND συνδέεται με την υποδοχή GROUND του Raspberry Pi και ο ακροδέκτης VCC με υποδοχή 5V. Ο ακροδέκτης TRIG συνδέεται με την GPIO 26 του Raspberry Pi ενώ ο ακροδέκτης ECHO αφού πρώτα συνδεθεί με αντίσταση 1k Ohm καταλήγει στην υποδοχή GPIO 23 του Raspberry Pi.



v. **Κάμερα**

Η μονάδα κάμερας Raspberry Pi είναι ικανή να εγγράψει full HD 1080p φωτογραφίες και βίντεο και μπορεί να ελεγχθεί μέσω προγραμματισμού**.**

Η συνδεσμολογία της κάμερας με το Raspberry Pi γίνεται με την καλωδιοταινία να εισάγεται στην υποδοχή μεταξύ της θύρας Ethernet και HDMI.

Η ρύθμιση του λογισμικού της κάμερας για την λειτουργία της με το Raspberry Pi γίνεται με τη γραμμή εντολών για να κατεβάσουμε και να εγκαταστήσουμε την τελευταία έκδοση του πυρήνα, firmware της GPU και εφαρμογές. Θα χρειαστούμε μια σύνδεση στο Internet για να λειτουργήσει σωστά.

Κατόπιν εκτελούμε τις εντολές :

sudo apt-get update

sudo apt-get upgrade

Έπειτα θα πρέπει να ενεργοποιήσουμε την υποστήριξη της κάμερας χρησιμοποιώντας το πρόγραμμα raspi-config το οποίο χρησιμοποιείται όταν ρυθμίζουμε για πρώτη φορά το Raspberry Pi.

Χρησιμοποιούμε τα πλήκτρα του κέρσορα για να μετακινηθούμε στην επιλογή κάμερα, και επιλέγουμε «enable». Ύστερα θα κάνουμε επανεκκίνηση. Η επιλογή «enable» θα εξασφαλίσει ότι κατά την επανεκκίνηση το σωστό firmware GPU θα τρέχει με το πρόγραμμα οδήγησης της κάμερας και ρύθμισης, καθώς και η κατανομή της μνήμης της GPU είναι επαρκής ώστε να επιτρέπει στην κάμερα να αποκτήσει αρκετή μνήμη για να τρέξει σωστά.

Για να ελέγξουμε ότι έχει εγκατασταθεί το σύστημα και λειτουργεί, εκτελούμε την παρακάτω εντολή:

raspistill -v -o test.jpg

Η οθόνη θα πρέπει να δείξει μια προεπισκόπηση των πέντε δευτερολέπτων από την κάμερα και στη συνέχεια να λάβει μια φωτογραφία, αποθηκεύοντας ένα test.jpg αρχείο, ενώ εμφανίζει διάφορα ενημερωτικά μηνύματα.

Ο Κώδικας σε Python είναι ο εξής

```
import picamera

from time import sleep

camera = picamera.PiCamera()

camera.capture('image.jpg')

camera.start_preview()

camera.vflip = True

camera.hflip = True

camera.brightness = 60


camera.start_recording('video.h264')

sleep(5)

camera.stop_recording()
```

ΚΕΦΑΛΑΙΟ 3^{Ο} : ΛΕΙΤΟΥΡΓΙΚΟ ΣΥΣΤΗΜΑ Linux Raspbian Jessy

3.1 Τί είναι το Linux;

Το Linux είναι ένα δωρεάν λειτουργικό σύστημα ανοιχτού κώδικα.

Η ιστορία του Linux ξεκίνησε το 1991 από τον, τότε φοιτητή του Πανεπιστημίου του Ελσίνκι, Linus Torvalds.

Από τότε ο πυρήνας (Kernel) του Linux χαρακτηρίζεται από συνεχή ανάπτυξη από τους υποστηρικτές του και από έναν μικρό αριθμό αρχείων βασισμένα σε κώδικα C έχει αναπτυχθεί σε ένα εμπορικό προϊόν με περισσότερες από 23εκ. γραμμές κώδικα.

3.2 Η δημιουργία του Linux

Το 1991 ο Linus Torvalds, παράλληλα με τις σπουδές του, ξεκίνησε ένα έργο που αργότερα έγινε ο πυρήνας του Linux. Έγραψε το πρόγραμμα ειδικά για το υλικό (hardware) που χρησιμοποιούσε και ήταν ανεξάρτητο από ένα λειτουργικό σύστημα καθώς ήθελε να χρησιμοποιήσει τις δικές του λειτουργίες στον υπολογιστή με επεξεργαστή 80836.

Η ανάπτυξή του έγινε στο MINIX με την χρήση του C μεταγλωτιστή (compiler) ο οποίος και εξακολουθεί να είναι ο κύριος τρόπος λειτουργίας/κατασκευής του Linux μέχρι σήμερα. Μπορεί επίσης να κατασκευαστεί με την χρήση άλλων compilers όπως για παράδειγμα ο Intel C.

3.3 Πως πήραν την ονομασία τους

Αρχικά ο Torvalds ήθελε να ονομάσει την εφεύρεσή του Freax (freak + x) ως υπαινιγμός στο UNIX. Τα πρώτα αρχεία μάλιστα του Project είχαν την ονομασία Freax. Ο Torvalds είχε ήδη σκεφθεί στην ονομασία Linux, αλλά πίστευε πως ήταν υπερβολικά εγωιστικό.

Εν τέλει το project ονομάστηκε Linux από τον Ari Lemmke που ήταν εθελοντής και διαχειριστής του FTP Server του Πανεπιστημίου. Ο Lemmke προχώρησε στην μετονομασία των αρχείων χωρίς να συμβουλευτεί τον Torvalds και τους άλλους εθελοντές. Αργότερα όλοι συμφώνησαν πως ήταν καλύτερο όνομα από το Freax και κατοχύρωσαν το Linux.

3.4 Η διαμάχη του ονόματος

Η ονομασία "Linux" χρησιμοποιήθηκε αρχικά από το Torvalds μόνο για τον πυρήνα του Linux. Ωστόσο, ο πυρήνας χρησιμοποιήθηκε συχνά μαζί με άλλο λογισμικό, ειδικά αυτό του έργου GNU.

Τον Ιούνιο του 1994 στις σελίδες του GNU, το Linux αναφερόταν ως "ελεύθερος κλώνος του UNIX" και το Debian άρχισε να αποκαλεί το προϊόν Debian GNU / Linux. Τον Μάιο του 1996, ο Richard Stallman δημοσίευσε τον επεξεργαστή Emacs 19.31, στον οποίο ο τύπος του συστήματος μετονομάστηκε από το Linux στο Lignux. Αυτή η ορθογραφία είχε ως στόχο να αναφέρεται ειδικά στον συνδυασμό του GNU και του Linux, αλλά σύντομα εγκαταλείφθηκε υπέρ του "GNU / Linux".

Αυτό το όνομα συγκέντρωσε διάφορες αντιδράσεις. Τα GNU και Debian χρησιμοποιούν το όνομα, αν και οι περισσότεροι χρησιμοποιούν απλώς τον όρο "Linux" για να αναφερθούν στον συνδυασμό.

3.5 Λειτουργικό Linux Debian Jessie

Το Debian είναι ένα λειτουργικό σύστημα ανοιχτού κώδικα που αναπτύσσεται από διάφορους εθελοντές/χρήστες ανά τον κόσμο. Βασίζεται στον κεντρικό πυρήνα του Linux και τα εργαλεία του GNU.

Η έκδοση που κάνουμε χρήση στο project μας είναι η ‘Jessie’ ή αλλιώς η έκδοση 8. Πήρε το όνομά της από τον ομώνυμο χαρακτήρα του Toy Story.

Για την δική μας ευκολία εγκαταστήσαμε την αντίστοιχη έκδοση που παρέχει το raspberry pi (RaSpbian Jessie) που είναι μια ποιο ‘ελαφριά’ έκδοση του Debian 8.

Το Raspbian jessie είναι εξίσου γνωστό για το σύστημα διαχείρισης των πακέτων (~43.000 έτοιμα προς εγκατάσταση) καθώς επίσης και για τις δυνατότητες που προσφέρει σε μια τόσο ελαφριά έκδοση έχοντας την υποστήριξη αρχιτεκτονικών όπως amd64, i386, armel, armhf, arm64, powerpc, ppc64el, mips, mipsel, s390x. Παράλληλα με την APT (advanced packging tool) μπορεί κάποιος εύκολα και γρήγορα να εγκαταστήσει ή να αναβαθμίσει τα πακέτα στο σύστημα του με ασφάλεια καθώς το Debian έχει υιοθετήσει αυστηρές πολιτικές ως προς την ποιότητα των πακέτων και των εκδόσεων του.

Το Debian, όπως και άλλα flavors του Linux και ανοιχτού λογισμικού, υποστηρίζονται από τις συνεχείς δωρεές που γίνονται από εταιρίες και οργανισμούς που κάνουν χρήση του λογισμικού τους.

Το τσιπ ARM11 στην καρδιά του Pi (μοντέλα πρώτης γενιάς) βασίζεται στην έκδοση 6 του ARM. Το πρωτεύον λειτουργικό σύστημα που υποστηρίζεται είναι Raspbian, αν και είναι συμβατά με πολλά άλλα. Η τρέχουσα έκδοση του Ubuntu υποστηρίζει το Raspberry Pi 2, ενώ το Ubuntu, και πολλές δημοφιλείς εκδόσεις του Linux, δεν υποστηρίζουν τα παλαιότερα Raspberry Pi 1 που τρέχει σε ARM11. Το Raspberry Pi 2 μπορεί επίσης να τρέξει Windows 10 με IoT πυρήνα, ενώ καμία έκδοση του Pi δεν μπορεί να τρέξει κανονικά Windows. Το Raspberry Pi 2 επι του παρόντος υποστηρίζει επίσης OpenELEC και RISC OS.

Ο διαχειριστής εγκατάστασης για το Raspberry Pi είναι το noobs. Τα λειτουργικά συστήματα που περιλαμβάνονται με noobs είναι:

- Arch Linux ARM
- OpenELEC

- OsmC (πρώην Raspbmc) και Kodi open source ψηφιακό κέντρο πολυμέσων
- Pidora (Fedora Remix)
- Puppy Linux
- RISC OS - είναι το λειτουργικό σύστημα του πρώτου ARM-based υπολογιστή .
- Raspbian (συνιστάται για Raspberry Pi 1) – βασίζεται στην Debian ARM (armhf) αρχιτεκτονική και είχε αρχικά σχεδιαστεί για ARMv7 και νεότερους επεξεργαστές (με Jazelle RCT / ThumbEE και VFPv3), και αργότερα συμπιέστηκε για το πιο περιορισμένο ARMv6 σύνολο εντολών του Raspberry Pi 1. απαιτείται ελάχιστο μέγεθος των 4 GB SD κάρτα για τις Raspbian διανομές που παρέχονται από το Ίδρυμα Raspberry Pi. Υπάρχει και Pi κατάστημα για την ανταλλαγή προγραμμάτων.
- Η Raspbian Server Edition είναι μια απογυμνωμένη έκδοση με λιγότερα πακέτα λογισμικού σε σύγκριση με το συνηθισμένη-σε υπολογιστή προσανατολισμένη Raspbian.
- PiBang Linux - προέρχεται από Raspbian
- Raspbian για Robots -. Είναι ένα παρακλάδι του Raspbian για έργα ρομποτικής με Lego, Grove, και Arduino

ΚΕΦΑΛΑΙΟ 4^{Ο} : Πρόσθετα του Raspberry Pi

4.1 Κάμερα

Η μονάδα κάμερας Raspberry Pi είναι ικανή να εγγράψει full HD 1080p φωτογραφίες και βίντεο και μπορεί να ελεγχθεί μέσω προγραμματισμού**.**

Η συνδεσμολογία της κάμερας με το Raspberry Pi γίνεται με την καλωδιοταινία να εισάγεται στην υποδοχή μεταξύ της θύρας Ethernet και HDMI

Σύνδεση κάμερας με Raspberry Pi

Η ρύθμιση του λογισμικού της κάμερας για την λειτουργιά της με το Raspberry Pi γίνεται με τη γραμμή εντολών για να κατεβάσουμε και να εγκαταστήσουμε την τελευταία έκδοση του πυρήνα, firmware της GPU και εφαρμογές. Θα χρειαστούμε μια σύνδεση στο Internet για να λειτουργήσει σωστά.

Κατοπιν εκτελούμε τις εντολές :

sudo apt-get update

sudo apt-get upgrade

Έπειτα θα πρέπει να ενεργοποιήσουμε την υποστήριξη της κάμερας χρησιμοποιώντας το πρόγραμμα raspi-config το όποιο χρησιμοποιείται όταν ρυθμίζουμε για πρώτη φορά το Raspberry Pi.

Χρησιμοποιουμε τα πλήκτρα του κέρσορα για να μετακινηθούμε στην επιλογή κάμερα, και επιλέγουμε «enable». Ύστερα θα κάνουμε επανεκκίνηση. Η επιλογή «enable». θα εξασφαλίσει ότι κατά την επανεκκίνηση το σωστό firmware GPU θα τρέχει με το πρόγραμμα οδήγησης της κάμερας και ρύθμισης, καθώς και η κατανομή της μνήμης της GPU είναι επαρκής ώστε να επιτρέπει στην κάμερα να αποκτήσει αρκετή μνήμη για να τρέξει σωστά.

Για να ελέγξουμε ότι έχει εγκατασταθεί το σύστημα και λειτουργεί, εκτελούμε την παρακάτω εντολή:

raspistill -v -o test.jpg

Η οθόνη θα πρέπει να δείξει μια προεπισκόπηση των πέντε δευτερολέπτων από την κάμερα και στη συνέχεια να λάβει μια φωτογραφία, αποθηκεύοντας ένα test.jpg αρχείο, ενώ εμφανίζει διάφορα ενημερωτικά μηνύματα.

Ο Κώδικας σε Python είναι ο εξής

```
import picamera
from time import sleep
camera = picamera.PiCamera()
camera.capture('image.jpg')
camera.start_preview()
```

```
camera.vflip = True
camera.hflip = True
camera.brightness = 60

camera.start_recording('video.h264')
sleep(5)
camera.stop_recording()
```

4.2 Web Interface Raspbery PI Camera

Το RPI Cam Interface Web είναι ένα web interface για την RPI κάμερα που μπορεί να ανοίξει σε οποιοδήποτε πρόγραμμα περιήγησης (περιλαμβάνεται smartphones)

Η εγκατάσταση του γίνεται με τα παρακάτω βήματα

1)Εγκατατασταση του Raspbian στο Raspberry Pi

2)Σύνδεση κάμερας και ενεργοποίηση υποστήριξης της κάμερας όπως περιγράφηκε παραπάνω

3)Αναβαθμιση του Raspberry PI με τις εντολές

- sudo apt-get update
- sudo apt-get dist-upgrade

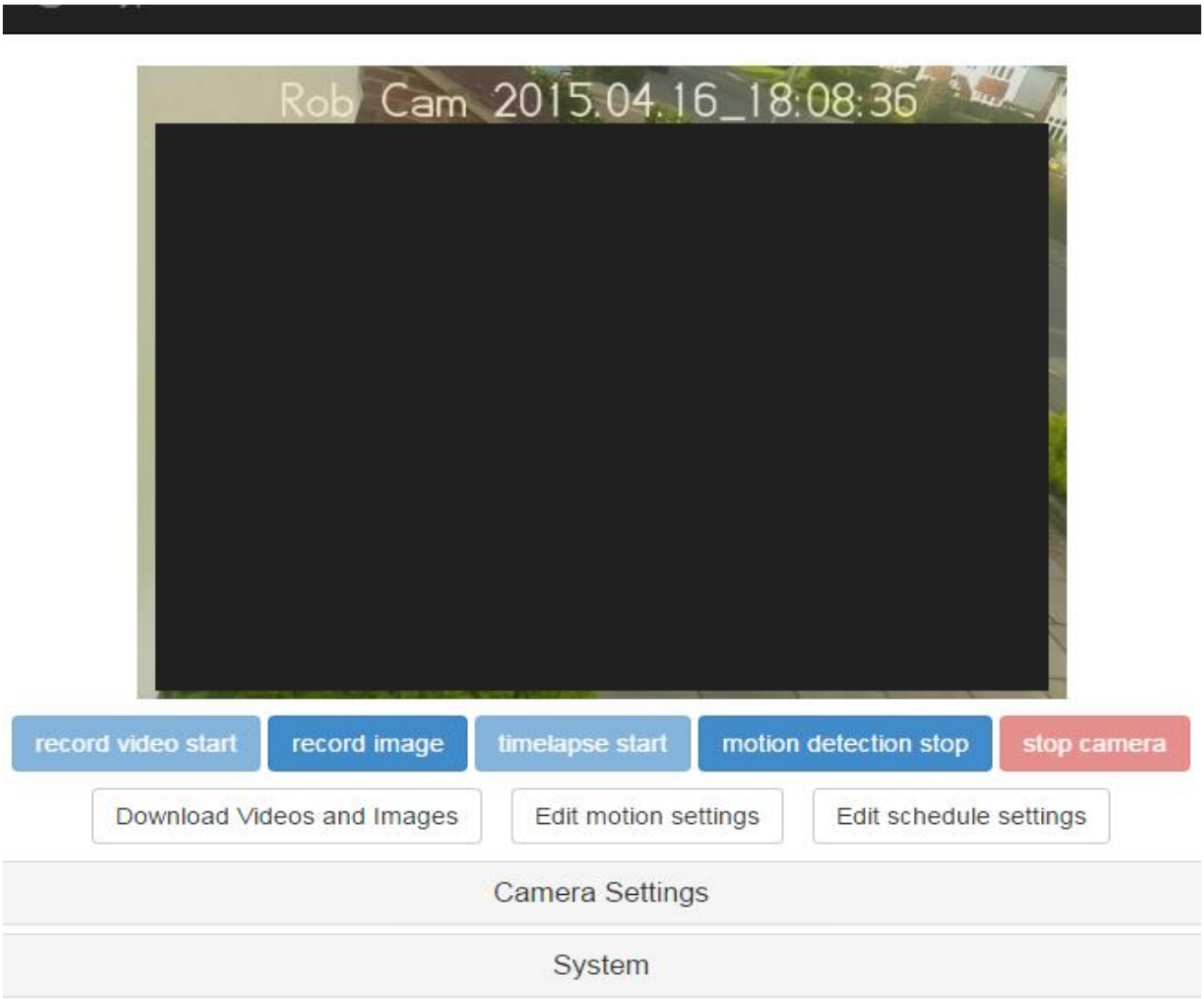
4)Για την έκδοση Jessie Lite εκτελούμε την εντολή sudo apt-get install git

Αντιγράφουμε τον κώδικα από το github και ενεργοποιούμε και τρέχουμε το script εγκατάστασης με τις ακόλουθες εντολές

- git clone https://github.com/silvanmelchior/RPi_Cam_Web_Interface.git
- cd RPi_Cam_Web_Interface

- chmod u+x RPi_Cam_Web_Interface_Installer.sh
- ./RPi_Cam_Web_Interface_Installer.sh install

Μετά το τέλος της εγκατάστασης είμαστε σε θέση να τρέξουμε το ./start.sh script για την εκκίνηση του Web Interface διαχείρισης της κάμερας



Έλεγχος κάμερας με RPi Web Interface

Τα βασικά χειριστήρια επιτρέπουν τη λήψη μεμονωμένων εικόνων και βίντεο. Μπορεί επίσης να τεθεί σε κατάσταση ανίχνευσης κίνησης, όπου οποιαδήποτε κίνηση θα προκαλέσει δραστηριότητα σαν μια εγγραφή βίντεο.

Τα κουμπιά κάτω από τα βασικά χειριστήρια επιτρέπουν τη λήψη και προεπισκόπηση οποιασδήποτε λήψης, καθώς και πρόσβαση σε ανίχνευση κίνησης και προγραμματισμούς εγγραφής η λειτουργίας

Κάτω από αυτό είναι οι ρυθμίσεις της κάμερας και ένα μια μπάρα ελέγχου του συστήματος.

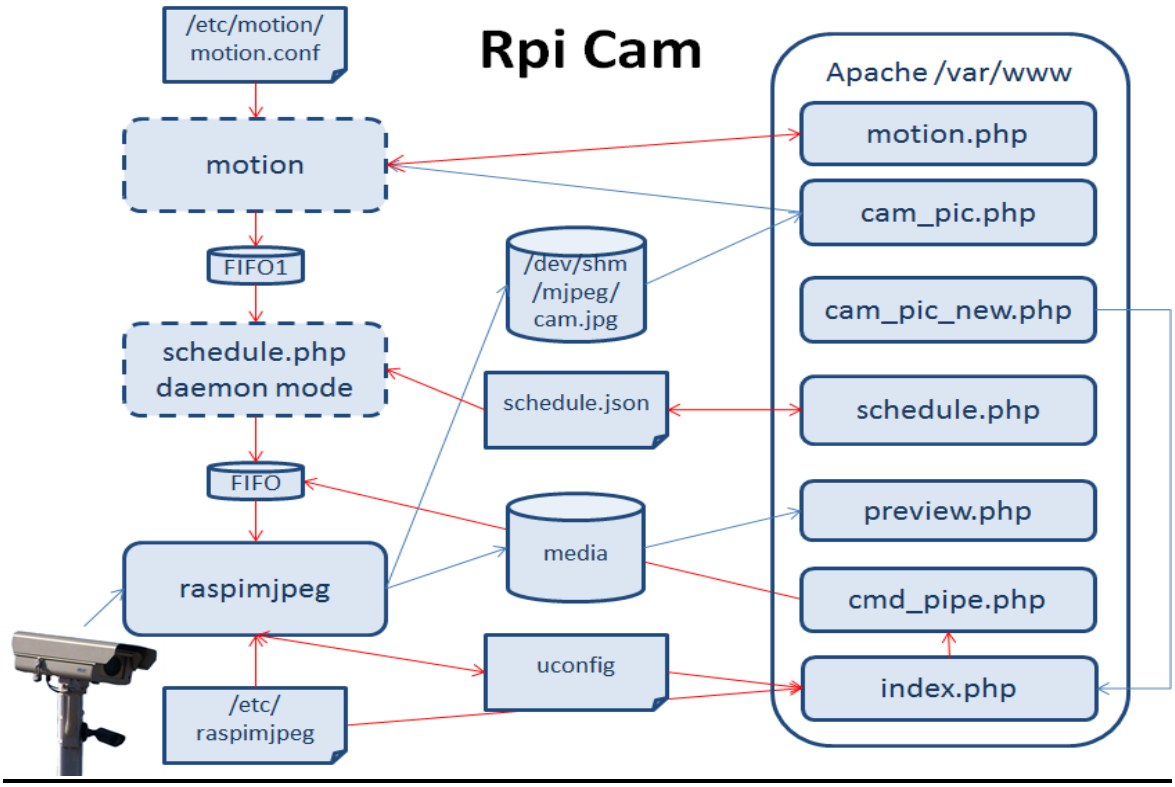
Κάνοντας κλικ στην εικόνα θα γίνει εναλλαγή μεταξύ κανονικής και πλήρους οθόνης. Εξ 'ορισμού ξεκινά σε κανονική λειτουργία, αλλά αυτό μπορεί να αλλάξει με τη χρήση της μεταβλητής fullscreen config.

Κάνοντας κλικ στο Ρυθμίσεις κάμερας δίνει πρόσβαση σε ένα ευρύ φάσμα επίλογων της φωτογραφικής μηχανής.

Η μπάρα ελέγχου του συστήματος επιτρέπει την επιλογή της ροής λειτουργίας, διακοπή λειτουργίας και επανεκκίνηση του συστήματος, επαναφορά ρυθμίσεων και την επιλογή των προσαρμοσμένων στυλ. Εξ ορισμού, το σύστημα δημιουργεί τη ζωντανή προεπισκόπηση ως μια συνεχής σειρά από εικόνες που έχουν ληφθεί, για να έχει μέγιστη συμβατότητα με το πρόγραμμα περιήγησης.

Η μπάρα προσαρμοσμένο στυλ δείχνει καμία επιπλέον στυλ που έχουν προστεθεί.

Η αρχιτεκτονική λειτουργιάς του προγράμματος είναι η ακόλουθη



Αρχιτεκτονική λειτουργίας του RPi Web Cam Interface

4.3 Πρόγραμμα απομακρυσμένης διαχείρισης του Raspberry Pi – NoIP

Το NO-IP είναι ένα πρόγραμμα διαχείρισης του Raspberry Pi και κατ' επέκταση του Ρομπότ απομακρυσμένα

Η διαδικασία εγκατάστασης του είναι η εξής. Ανοίγουμε ένα τερματικό κα πληκτρολογούμε τα παρακάτω πιέζοντας ENTER μετά από κάθε σειρά

- mkdir /home/pi/noip
- cd /home/pi/noip

Μετα την δημιουργία των φακέλων κατεβάζουμε το λογισμικό με τις εντολές

- wget http://www.no-ip.com/client/linux/noip-duc-linux.tar.gz

- tar vzxf noip-duc-linux.tar.gz

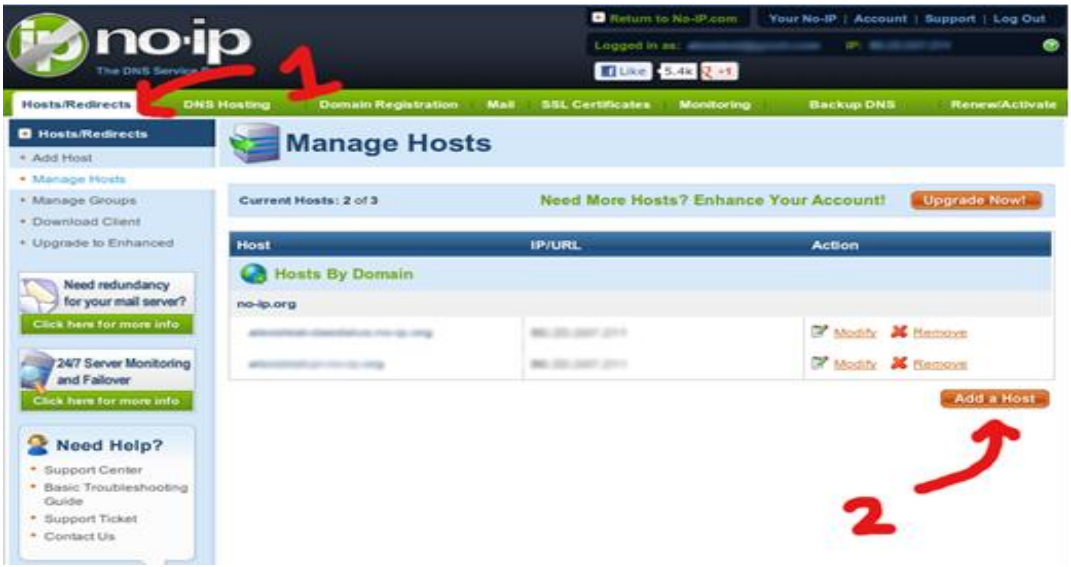
Ακολούθως πηγαίνουμε στον κατάλογο για να εντοπισουμε τα αρχεία που κατεβάσαμε

- cd noip-2.1.9-1

Αρχιζουμε την εγκατάσταση του προγράμματος

- sudo make
- sudo make install

Υστερα εισερχόμαστε στην βασική οθόνη λειτουργιάς του NO-IP



Σχ1.9 Web Interface της υπηρεσίας NO-IP

ΚΕΦΑΛΑΙΟ 5^{Ο} : ΓΛΩΣΣΑ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΥ PYTHON

1.1 Τι είναι η Python

Η Python είναι μια υψηλή γλώσσα προγραμματισμού η οποία δημιουργήθηκε το 1990 από τον Ολλανδό Guido van Rossum κι αρχικά χρησιμοποιούνταν στο λειτουργικό σύστημα Amoeba. Στόχος της είναι η αναγνωρισιμότητα του κώδικας της και η ευκολία χρήσης της και το συντακτικό της επιτρέπει στους προγραμματιστές να εκφράσουν έννοιες σε λιγότερες γραμμές κώδικα εν αντιθέσει της γλώσσας C++ και Java.

Οι διερμηνευτές της Python είναι διαθέσιμοι σε πολλά λειτουργικά συστήματα επιτρέποντας στην Python την εκτέλεση κώδικα σε ευρεία γκάμα συστημάτων. Η Python αναπτύσσεται ως open source(ανοιχτό λογισμικό) και η διαχείριση της γίνεται από τον μη κερδοσκοπικό οργανισμό Python Software Foundation.

Η γλώσσα χρησιμοποιεί compiler(μεταγλωττιστή) για την δημιουργία του εκτελέσιμου κώδικα και σχετίζεται με τις γλώσσες προγραμματισμού Tcl, Perl, Scheme, Java και Ruby. Ένα από τα πιο απλά προγράμματα είναι η εμφάνιση ενός γραπτού αποτελέσματος όπως πχ:

```
>>> print("Γεια σου, κόσμε!")
Γεια σου, κόσμε!
```

Ένα ιδιαίτερο χαρακτηριστικό της γλώσσας είναι η χρήση κενών διαστημάτων για τον διαχωρισμό των συντακτικών δομών του προγράμματος. Σε συνδυασμό με τη χρήση αγγλικών λέξεων στη θέση συμβόλων καθιστούν τον κώδικα της Python ευανάγνωστο από όσους έχουν βασική γνώση των αγγλικών όπως φαίνεται στο παρακάτω παράδειγμα.

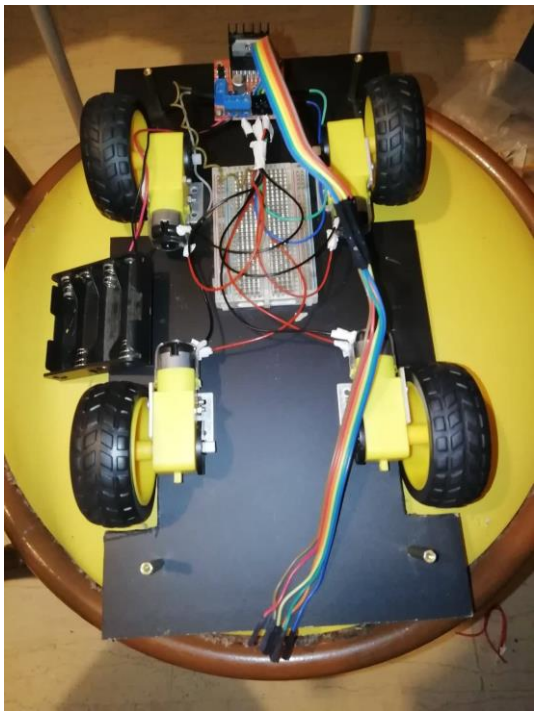
```
age = 21
if age >= 18:
    print("You vote")
else:
    print("You don't vote")
```

Η Python 2.0 κυκλοφόρησε στις 16 Οκτωβρίου του 2000. Στις 3 Δεκεμβρίου του 2008 κυκλοφόρησε η έκδοση 3.0. Πολλά από τα καινούργια χαρακτηριστικά αυτής της έκδοσης έχουν μεταφερθεί στις εκδόσεις 2.6 και 2.7.

Η Python 3.0 είναι ιστορικά η πρώτη γλώσσα προγραμματισμού που σπάει την προς τα πίσω συμβατότητα με προηγούμενες εκδόσεις ώστε να διορθωθούν κάποια λάθη που υπήρχαν σε προγενέστερες εκδόσεις και να καταστεί ακόμα πιο σαφής ο απλός τρόπος με τον οποίο μπορούν να γίνουν κάποια πράγματα.

ΚΕΦΑΛΑΙΟ 6^{Ο} : ΣΥΝΔΕΣΜΟΛΟΓΙΑ ΚΑΙ ΚΩΔΙΚΑΣ ΠΑΡΚΑΡΙΣΜΑΤΟΣ ΑΜΑΞΙΔΙΟΥ

6.1 Συνδεσμολογία 1^{ου} επιπέδου



Έξοδος 1 (OUT1) από τον LN298N καφέ καλώδιο είναι συνδεδεμένο με το μαύρο καλώδιο της μπροστά δεξιά ρόδας σε συνδυασμό με το κόκκινο καλώδιο της πίσω δεξιά ρόδας του αμαξιδίου.

Έξοδος 2 (OUT2) από τον LN298N γκρι καλώδιο είναι συνδεδεμένο με το μαύρο καλώδιο της πίσω δεξιά ρόδας σε συνδυασμό με το κόκκινο καλώδιο της μπροστά δεξιά ρόδας του αμαξιδίου.

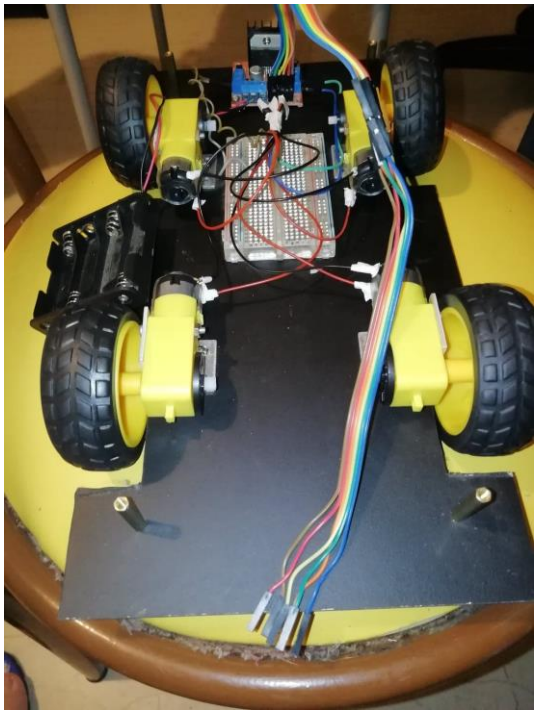
Έξοδος 3 (OUT3) από τον LN298N μπλε καλώδιο είναι συνδεδεμένο με το μαύρο καλώδιο της πίσω αριστερά ρόδας σε συνδυασμό με το κόκκινο καλώδιο της μπροστά αριστερά ρόδας του αμαξιδίου.

Έξοδος 4 (OUT4) από τον LN298N πράσινο καλώδιο είναι συνδεδεμένο με το μαύρο καλώδιο της μπροστά αριστερά ρόδας σε συνδυασμό με το κόκκινο καλώδιο της πίσω αριστερά ρόδας του αμαξιδίου.

Η θύρα +12V του LN298N είναι συνδεδεμένη με το κόκκινο καλώδιο που οδηγεί στην τροφοδοσία.

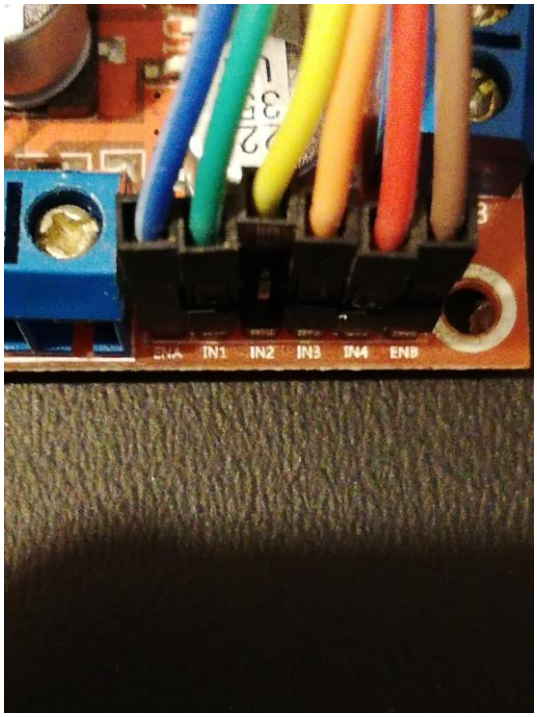
Η θύρα GND του LN298N είναι συνδεδεμένη με το μαύρο καλώδιο που οδηγεί στην τροφοδοσία.

Η θύρα GND του LN298N παρέχει παράλληλα γείωση στο breadboard που βρίσκεται στο 1ο επίπεδο του αμαξιδίου και από εκεί σε όλα τα παρελκόμενα εξαρτήματα και κυκλώματα που έχουμε ενσωματώσει στο αμαξίδιο.



Breadboard 1ου επιπέδου GND

- Θέση 22η μαύρο καλώδιο στην γείωση του οπίσθιου mini breadboard.
- Θέση 23η μαύρο καλώδιο στην γείωση του δεξιού mini breadboard.
- Θέση 24η μαύρο καλώδιο στην γείωση του αριστερού mini breadboard.
- Θέση 25η μωβ καλώδιο στην θύρα 6 του raspberry pi.



Η έξοδος ENA του LN298N είναι συνδεδεμένη το μπλε καλώδιο που οδηγεί στην θύρα του raspberry pi 7.

Η έξοδος IN1 του LN298N είναι συνδεδεμένη το πράσινο καλώδιο που οδηγεί στην θύρα του raspberry pi 11.

Η έξοδος IN2 του LN298N είναι συνδεδεμένη το κίτρινο καλώδιο που οδηγεί στην θύρα του raspberry pi 12.

Η έξοδος IN3 του LN298N είναι συνδεδεμένη το πορτοκαλί καλώδιο που οδηγεί στην θύρα του raspberry pi 13.

Η έξοδος IN4 του LN298N είναι συνδεδεμένη το κόκκινο καλώδιο που οδηγεί στην θύρα του raspberry pi 15.

Η έξοδος ENB του LN298N είναι συνδεδεμένη το καφέ καλώδιο που οδηγεί στην θύρα του raspberry pi 16.

6.2 Συνδεσμολογία 2ου επιπέδου

Ξεκινάμε με τη συνδεσμολογία του μπροστινού αισθητήρα υπερύθρων.

6.2.1 Συνδεσμολογία του μπροστινού αισθητήρα

Vcc (λευκό-καφέ): από την έξοδο 4 του raspberry στο mini breadboard θέση A1 και από εκεί με το καφέ καλώδιο της θέσης A4 προς την είσοδο Vcc του αισθητήρα.

Ground (καφέ): από την έξοδο 39 του rasberry στο mini breadboard θέση A10 και από εκεί με το πορτοκάλι καλώδιο της θέσης A8 προς την είσοδο Ground του αισθητήρα.

Echo (πράσινο - μωβ): από την έξοδο 37 του rasberry στο mini breadboard θέση C4 και στο ένα ποδαράκι της αντίστασης μας (B4) και στη συνέχεια από το άλλο ποδαράκι της αντίστασης (B8) με το μωβ καλώδιο στη θέση C8 πρός την είσοδο Echo του αισθητήρα.

Trigger (λευκό): από την έξοδο 38 του raspberry προς την είσοδο Trigger του αισθητήρα.

6.2.2 Συνδεσμολογία του δεξιού αισθητήρα

Vcc (πορτοκαλί-πράσινο): με το πορτοκαλί καλώδιο από τη θέση A4 του μποστινού mini breadboard προς την θέση G10 του δεξιού mini breadboard με πράσινο καλώδιο.

Ground (μαύρο): από την θέση 23 με μαύρο καλώδιο στην γείωση του δεξιού mini breadboard θέση F7.

Echo (μωβ-κίτρινο): από την έξοδο 21 του rasberry στο mini breadboard θέση C6 και στο ένα ποδαράκι της αντίστασης μας (B6) και στη συνέχεια από το άλλο ποδαράκι της αντίστασης (B12) με το κίτρινο καλώδιο στη θέση C12 πρός την είσοδο Echo του αισθητήρα.

Trigger (γκρι): από την έξοδο 22 του raspberry προς το mini breadboard θέση H9.

6.2.3 Συνδεσμολογία του πίσω αισθητήρα

Vcc (λευκό): από το mini breadboard του δεξιού αισθητήρα θέση F10 προς το mini breadboard του πίσω αισθητήρα θέση G10.

Ground (μαύρο): από τη θέση 22 με μαύρο καλώδιο στην γείωση του οπίσθιου mini breadboard θέση F7.

Echo (μωβ-μαύρο): από την έξοδο 18 του rasberry στο mini breadboard θέση C5 και στο ένα ποδαράκι της αντίστασης μας (B5) και στη συνέχεια από το άλλο ποδαράκι της αντίστασης (B11) με το μαύρο καλώδιο στη θέση C11 πρός την είσοδο Echo του αισθητήρα.

Trigger (γκρι): από την έξοδο 19 του raspberry προς το mini breadboard θέση H9.

6**.2.4** Συνδεσμολογία του αριστερού αισθητήρα

Vcc (λευκό): από το mini breadboard του μπροστινου αισθητήρα θέση A5 προς το mini breadboard του αριστερού αισθητήρα θέση G10.

Ground (μαύρο): από τη θέση 24 με μαύρο καλώδιο στην γείωση του οπίσθιου mini breadboard θέση F7.

Echo (μωβ-πράσινο): από την έξοδο 23 του rasberry στο mini breadboard θέση C6 και στο ένα ποδαράκι της αντίστασης μας (B6) και στη συνέχεια από το άλλο ποδαράκι της αντίστασης (B11) με το πράσινο καλώδιο στη θέση C11 προς την είσοδο Echo του αισθητήρα.

Trigger (γκρι): από την έξοδο 24 του raspberry προς το mini breadboard θέση H9.

6.3 Κώδικας αισθητήρα υπερύθρων

Για να δημιουργήσουμε τον παλμό ενεργοποίησης, θέτουμε τον ακροδέκτη του Trigger σε κατάσταση “high” για 10μS και στη συνέχεια ξανά σε “low”.

```
GPIO.output(TRIG, True)
time.sleep(0.00001)
GPIO.output(TRIG, False)
```

Ο αισθητήρας θέτει το ECHO σε “high” για το χρονικό διάστημα που χρειάζεται για να πάει ο παλμός και να επιστρέψει, οπότε ο κώδικας μας πρέπει να μετρήσει το **χρόνο** που ο ακροδέκτης ECHO παραμένει σε κατάσταση “high”. Χρησιμοποιούμε την συνάρτηση "while" για να βεβαιωθούμε ότι κάθε σήμα καταγράφεται με τη σωστή σειρά.

Η συνάρτηση time.time () θα καταγράψει την πιο πρόσφατη χρονική σήμανση. Εάν για παράδειγμα ένας ακροδέκτης πηγαίνει από κατάσταση “low” σε “high” και καταγράφουμε τη “low” κατάσταση, η καταγραφείσα χρονική σήμανση θα είναι η τελευταία κατά την οποία ο ακροδέκτης ήταν “low”.

Επομένως, το πρώτο μας βήμα πρέπει να είναι η καταγραφή της τελευταίας χαμηλής χρονικής σήμανσης για ECHO (pulse_start)

```
while GPIO.input(ECHO)==0:
  pulse_start = time.time()
```

Μόλις ληφθεί ένα σήμα, η τιμή αλλάζει από “low”(0) σε “high” (1) και το σήμα θα παραμείνει “high” για τη διάρκεια του παλμού. Επομένως, χρειαζόμαστε την τελευταία υψηλή χρονική σήμανση για ECHO (pulse_end).

```
while GPIO.input(ECHO)==1:
  pulse_end = time.time()
```

Τώρα μπορούμε να υπολογίσουμε τη διαφορά μεταξύ των δύο καταγεγραμμένων χρονικών σημείων (pulse_duration).

```
pulse_duration = pulse_end - pulse_start
```

Με το χρόνο που χρειάζεται για να μεταφερθεί το σήμα σε ένα αντικείμενο και πάλι πίσω, μπορούμε να υπολογίσουμε την απόσταση χρησιμοποιώντας τον ακόλουθο τύπο.

$$Speed = \frac{Distance}{Time}$$

Η ταχύτητα του ήχου είναι μεταβλητή, ανάλογα με το μέσο στο οποίο περνάει και τη θερμοκρασία αυτού του μέσου. Ωστόσο, γνωρίζουμε ότι η ταχύτητα του ήχου στη στάθμη της θάλασσας είναι 343m/s. Αν προσπαθήσουμε να μετρήσουμε την απόσταση μέσω του νερού πρέπει να σιγουρευτούμε ότι χρησιμοποιούμε τη σωστή ταχύτητα του ήχου!

Πρέπει επίσης να διαιρέσουμε το χρόνο μας κατά δύο επειδή αυτό που έχουμε υπολογίσει παραπάνω είναι στην πραγματικότητα ο χρόνος που χρειάζεται για τον παλμό να ταξιδέψει την απόσταση στο αντικείμενο και πάλι πίσω. Μπορούμε να απλουστεύσουμε τον υπολογισμό στο script μας ως εξής:

$$34300 = \frac{Distance}{Time/2}$$

$$17150 = \frac{Distance}{Time}$$

$$17150 \times Time = Distance$$

Μπορούμε να βάλουμε αυτόν τον υπολογισμό στο script μας:

distance = pulse_duration x 17150

Έπειτα να στρογγυλοποιήσουμε την απόσταση μας σε 2 δεκαδικά ψηφία για δική μας διευκόλυνση:

distance = round(distance, 2)

στη συνέχεια, εκτυπώνουμε την απόσταση σε “cm”

print "Distance:",distance,"cm"

Τέλος, καθαρίζουμε τους ακροδέκτες GPIO για να διασφαλίσουμε την επαναφορά όλων των εισόδων / εξόδων:

GPIO.cleanup()

6.3.1 Μπροστινός αισθητήρας

```
import RPi.GPIO as gpio
import time

def distance():

        gpio.setmode(gpio.BOARD)
        TRIG = 38
        ECHO = 37
        gpio.setup(TRIG,gpio.OUT)
        gpio.setup(ECHO,gpio.IN)
        gpio.output(TRIG, False)
        time.sleep(0.5)
        gpio.output(TRIG, True)
        time.sleep(0.00001)
        gpio.output(TRIG, False)
        while gpio.input(ECHO)==0:
                pulse_start = time.time()
        while gpio.input(ECHO)==1:
                pulse_end = time.time()
        pulse_duration = pulse_end - pulse_start
        distance = pulse_duration * 17150
        distance = round(distance, 2)
        gpio.cleanup()
        if distance < 1:
                distance()
                print "error distance",distance()
        else:
                return distance

print "Front Distance:",distance()
```

6.3.2 Πίσω αισθητήρας

```
import RPi.GPIO as gpio
import time

def distance():

        gpio.setmode(gpio.BOARD)
        TRIG = 19
        ECHO = 18
        gpio.setup(TRIG,gpio.OUT)
        gpio.setup(ECHO,gpio.IN)
        gpio.output(TRIG, False)
        time.sleep(1)
        gpio.output(TRIG, True)
        time.sleep(0.00001)
        gpio.output(TRIG, False)
        while gpio.input(ECHO)==0:
                pulse_start = time.time()
        while gpio.input(ECHO)==1:
                pulse_end = time.time()
        pulse_duration = pulse_end - pulse_start
        distance = pulse_duration * 17150
        distance = round(distance, 2)
        gpio.cleanup()
        return distance
print "Back Distance:",distance()
```

6.3.3 Αριστερός αισθητήρας

```
import RPi.GPIO as gpio
import time

def distance():

        gpio.setmode(gpio.BOARD)
        TRIG = 24
        ECHO = 23
        gpio.setup(TRIG,gpio.OUT)
        gpio.setup(ECHO,gpio.IN)
        gpio.output(TRIG, False)
        time.sleep(0.5)
        gpio.output(TRIG, True)
        time.sleep(0.00001)
        gpio.output(TRIG, False)
        while gpio.input(ECHO)==0:
                pulse_start = time.time()
        while gpio.input(ECHO)==1:
                pulse_end = time.time()
        pulse_duration = pulse_end - pulse_start
        distance = pulse_duration * 17150
        distance = round(distance, 2)
        gpio.cleanup()
        return distance

print "Left Distance:",distance()
```

6.3.4 Δεξιός αισθητήρας

```
import RPi.GPIO as gpio
import time

def distance():

        gpio.setmode(gpio.BOARD)
        TRIG = 22
        ECHO = 21
        gpio.setup(TRIG,gpio.OUT)
        gpio.setup(ECHO,gpio.IN)
        gpio.output(TRIG, False)
        time.sleep(0.5)
        gpio.output(TRIG, True)
        time.sleep(0.00001)
        gpio.output(TRIG, False)
        while gpio.input(ECHO)==0:
                pulse_start = time.time()
        while gpio.input(ECHO)==1:
                pulse_end = time.time()
        pulse_duration = pulse_end - pulse_start
        distance = pulse_duration * 17150
        distance = round(distance, 2)
        gpio.cleanup()
        return distance
```

```
print "Right Distance:",distance()
```

6.4 Κώδικας παρκαρίσματος

Για την χρήση του παρκαρίσματος χρειαζόμαστε τις αποστάσεις και από τις 4 πλευρές του αμαξιδίου. Στην συνάρτηση `findAndPark` ορίζουμε από ποιους αισθητήρες θα λάβει μέτρηση το πρόγραμμά μας κατά την διάρκεια εκτέλεσής του. Έτσι μπορούμε να ορίσουμε από την αρχή αν θέλουμε το αμαξίδιο να παρκάρει από τα αριστερά ή δεξιά ανάλογα με την φορά. Θα πρέπει αντίστοιχα να χρησιμοποιήσουμε άλλη μέθοδο (`Park`).

Αρχικοποιούμε τις εισόδους από το raspberry με την `def init` ως εξής :

7 = INA του Motor Controller LN298. Όταν είναι σε κατάσταση '1' τότε οι IN1, IN2 ελέγχουν την κίνηση στις ρόδες που βρίσκονται στην δεξιά πλευρά του αμαξιδίου.

11 = IN1 για κίνηση μπροστά στις ρόδες από την δεξιά πλευρά του αμαξιδίου.

12 = IN2 για κίνηση πίσω στις ρόδες από την δεξιά πλευρά του αμαξιδίου.

13 = IN3 για κίνηση πίσω στις ρόδες από την αριστερή πλευρά του αμαξιδίου.

15 = IN4 για κίνηση μπροστά στις ρόδες από την αριστερή πλευρά του αμαξιδίου.

16 = INB του Motor Controller LN298. Όταν είναι σε κατάσταση '1' τότε οι IN3, IN4 ελέγχουν την κίνηση στις ρόδες που βρίσκονται στην αριστερή πλευρά του αμαξιδίου.

Έπειτα δημιουργούμε τις συναρτήσεις για την κίνηση του αμαξιδίου. Στην ουσία θέτοντας τους απαραίτητους ακροδέκτες, που περιγράφουμε παραπάνω, σε κατάσταση "high" ή "low" δίνουμε την κίνηση που επιθυμούμε στο αμαξίδιο.

`halt` = Συνάρτηση για το σταμάτημα της κίνησης.

`forward_always` = Συνάρτηση για εμπρόσθια κίνηση η οποία χρησιμοποιείται μόνο στις συνθήκες ελέγχου απόστασης. Η διαφορά της με την go_front είναι ότι δεν χρήζει ανάγκης κάποιου tf(time frame) και η κίνηση της διακόπτεται MONO όταν κάποια από τις συνθήκες ελέγχου πληροί τις προϋποθέσεις που έχουμε ορίσει στο πρόγραμμα.

`go_back` = Συνάρτηση οπίσθια κίνηση.

`go_front` = Συνάρτηση για εμπρόσθια κίνηση.

`pivot_left` = Συνάρτηση για στροφή προς τα αριστερά.

`pivot_right` = Συνάρτηση για στροφή προς τα δεξιά.

Εφόσον έχουμε ορίσει τις εισόδους μας αλλά και τις συναρτήσεις για την κίνηση του αμαξιδίου, απομένει να δημιουργήσουμε την συνάρτηση με την οποία το αμαξίδιο θα παρκάρει στο χώρο.
Επομένως δημιουργήσαμε την συνάρτηση `findAndPark`. Έχοντας υπόψη ότι το αμαξίδιο βρίσκεται σε στάση, πραγματοποιούμε εμπρόσθια κίνηση. Όσο το

αμαξίδιο κινείται προς τα εμπρός γίνεται έλεγχος απόστασης από τους αισθητήρες. Ο 1ος έλεγχος που γίνεται με την συνθήκη :

```
if right() < 35 and front() > 5:
```

Ορίζει την εμπρόσθια κίνηση του αμαξιδίου για όσο ο δεξιός αισθητήρας έχει απόσταση μικρότερη από 35 εκατοστά και ο μπροστινός μεγαλύτερη από 5 εκατοστά. Το αμαξίδιο θα συνεχίσει να κινείται μπροστά μέχρις ότου ένας από τους 2 ελέγχους πάψει να ισχύει. Όταν ο δεξιός αισθητήρας επιστρέψει τιμή μεγαλύτερη από 35 εκατοστά, τότε ενεργοποιείται ο 2ος έλεγχος με την συνθήκη :

```
elif right() > 35 and front() > 5:
```

Τότε σημαίνει ότι ο χώρος στα δεξιά του αμαξιδίου είναι αρκετός για να ξεκινήσει την διαδικασία λήψης και ελέγχου αποστάσεων. Η διαδικασία αυτή ορίζει τα εξής :

```
time.sleep(2)
start = front()
```

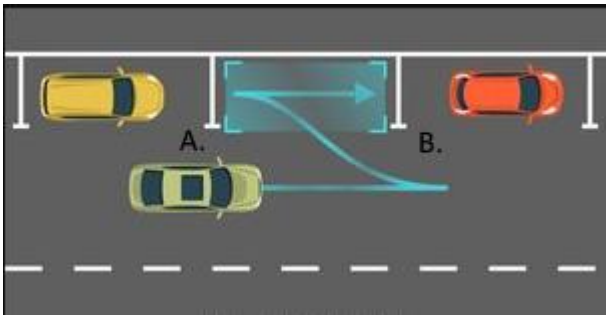
Το αμαξίδιο έχοντας 2 δευτερόλεπτα χρόνο θα λάβει μια μέτρηση από τον μπροστινό αισθητήρα την οποία θα περάσουμε σε μια μεταβλητή start. Την μεταβλητή start θα την χρειαστούμε για τον έλεγχο που θα κάνουμε στην επόμενη συνθήκη. Με την 3η λοιπόν συνθήκη ελέγχουμε πόση απόσταση θα διανύσει το αμαξίδιο από το σημείο Α (μεταβλητή start) μέχρι το σημείο Β (μεταβλητή dist) και αν αυτή η απόσταση είναι αρκετή έτσι ώστε το αμαξίδιο να ξεκινήσει την διαδικασία παρκαρίσματος. Αλλά ας δούμε παρακάτω τι ακριβώς κάνουμε :

```
while right() > 35 and front() > 5:
```

Έχοντας ξεκινήσει την διαδικασία παρκαρίσματος και για όσο ισχύει η συνθήκη while, λαμβάνουμε την απόσταση από τον μπροστινό αισθητήρα και αποθηκεύουμε την τιμή στην μεταβλητή now. Στην συνέχεια αφαιρούμε την απόσταση που έχουμε λάβει, στην παραπάνω συνθήκη από την μεταβλητή start με την μεταβλητή now και το αποτέλεσμά τους το εισάγουμε στην μεταβλητή dist :

```
now = front()
print "now",now
forward_always()
dist = start - now
```

Η παραπάνω πράξη γίνεται όσες φορές χρειαστεί για όσο ισχύει η συνθήκη μας. Έτσι με αυτόν τον τρόπο μπορούμε να γνωρίζουμε την απόσταση μεταξύ 2 σημείων (Α και Β) που βρίσκονται στα δεξιά του αμαξιδίου μας.



Η συνθήκη while λοιπόν εκτελείται για όσο οι αισθητήρες μας έχουν τις αποστάσεις που τους έχουμε ορίσει και όσο η μεταβλητή dist είναι μικρότερη από 30 εκατοστά :

```
if dist > 30:
```

Μόλις η μεταβλητή dist επιστρέψει αποτέλεσμα μεγαλύτερο από 30 εκατοστά τότε το αμαξίδιο μπαίνει στην τελική φάση παρκαρίσματος εκτελώντας την συνάρτηση `def Park():.` Η συνάρτηση αυτή εκτελεί μεθοδικές κινήσεις που έχουν οριστεί από εμάς και συντελούν έτσι ώστε το αμαξίδιο να εισέλθει στο χώρο παρκαρίσματος που εντόπισε κατά την εκτέλεση της προηγούμενης συνάρτησης. Προς το παρόν δεν έχει οριστεί να γίνεται κάποιος προηγμένος έλεγχος.

```
import RPi.GPIO as gpio
import time
import sys
from range_sensor_right import distance as right
from range_sensor_left import distance as left
from range_sensor_front import distance as front
from range_sensor_back import distance as back

def init():
        gpio.setmode(gpio.BOARD)
        gpio.setup(7, gpio.OUT)
        gpio.setup(11, gpio.OUT)
        gpio.setup(12, gpio.OUT)
        gpio.setup(13, gpio.OUT)
        gpio.setup(15, gpio.OUT)
        gpio.setup(16, gpio.OUT)

def halt():
        init()
        gpio.output(7, False)
        gpio.output(11, False)
        gpio.output(12, False)
        gpio.output(13, False)
        gpio.output(15, False)
        gpio.output(16, False)
        print "Wheels stopped"

def forward_always():
        init()
        gpio.output(7, True)
        gpio.output(11, True)
        gpio.output(12, False)
        gpio.output(13, False)
        gpio.output(15, True)
        gpio.output(16, True)

def go_back(tf):
        init()
        gpio.output(7, True)
        gpio.output(11, False)
        gpio.output(12, True)
        gpio.output(13, True)
        gpio.output(15, False)
        gpio.output(16, True)
        time.sleep(tf)
        gpio.cleanup()

def go_front(tf):
        init()
        gpio.output(7, True)
        gpio.output(11, True)
        gpio.output(12, False)
        gpio.output(13, False)
        gpio.output(15, True)
        gpio.output(16, True)
```

```
        time.sleep(tf)
        gpio.cleanup()

def pivot_left(tf):
        init()
        gpio.output(7, True)
        gpio.output(11, True)
        gpio.output(12, False)
        gpio.output(13, True)
        gpio.output(15, False)
        gpio.output(16, True)
        time.sleep(tf)
        gpio.cleanup()

def pivot_right(tf):
        init()
        gpio.output(7, True)
        gpio.output(11, False)
        gpio.output(12, True)
        gpio.output(13, False)
        gpio.output(15, True)
        gpio.output(16, True)
        time.sleep(tf)
        gpio.cleanup()

def Park():
        print "park"
        init()
        go_back(0.2)
        time.sleep(1)
        pivot_left(1.5)
        time.sleep(1)
        go_back(0.8)
        time.sleep(1)
        pivot_right(1.6)
        time.sleep(1)
        go_front(0.1)
        halt()
        gpio.cleanup()

def findAndPark():
        init()
        forward_always()
        if right() < 35 and front() > 5:
                forward_always()
                findAndPark()
        elif right() > 35 and front() > 5:
                print "Start Measuring Parking Space"
                time.sleep(2)
                start = front()
                print "start",start
                while right() > 35 and front() > 5:
                        now = front()
                        print "now",now
                        forward_always()
                        dist = start - now
                        print "print distance before",dist
                        if dist > 30:
                                Park()
```

```
        else:
                halt()
                gpio.cleanup()

try:
        findAndPark()
except KeyboardInterrupt:
        gpio.cleanup()

print "End of programm"
```

ΚΕΦΑΛΑΙΟ 7^{Ο} : ΒΙΒΛΙΟΓΡΑΦΙΑ

1. Liden, Daniel. "What Is a Driverless Car?". *WiseGeek*. Retrieved 11 October 2013.
2. Paul Goodman. Advantages-and-Disadvantages-of-Driverless-Cars AxleAddict . August 11, 2015.
3. Efa2. Raspberry_Pi_B%2B_rev_1.2 Wikimedia. 17 January 2015
4. modmypi installing-the-raspberry-pi-camera-board modmypi. May 22, 2013
5. micropik SG90Servo micropik. May 26,2015
6. *Boris Landoni* computer-vision-with-raspberry-pi-and-the-camera-pi-module open-electronics. *October 10, 2014*
7. Intorobotics 20-hand-picked-raspberry-pi-tutorials-in-computer-vision intorobotics. August 3, 2015
8. AdrianRosebrock home-surveillance-and-motion-detection-with-the-raspberry-pi-python-and-opencv pyimagesearch. June 1, 2015
9. AdrianRosebrock how-to-install-opencv-3-on-raspbian-jessie pyimagesearch. October 26, 2015
10. Harrison user-control-raspberry-pi-car pythonprogramming. 2016
11. plotlygraphs Software-Setup-Run-all-of-these-commands-in-your-T instructables. 2014
12. Raspberry Pi - Τι Είναι και Γιατί θα Θέλατε Ένα https://www.pcsteps.gr/52828-%CF%84%CE%B9-%CE%B5%CE%AF%CE%BD%CE%B1%CE%B9-%CF%84%CE%BF-raspberry-pi/
13. Το μέλλον των μεταφορών 2020 - 2030 και πέρα! http://www.futuremobility.gr/tech-news/events/to-mellon-twn-metaforon-2020-2030-kai-pera
14. Οι προκλήσεις ασφαλείας των αυτόνομων οχημάτων https://www.nrso.ntua.gr/geyannis/wp-content/uploads/geyannis-cp219.pdf