

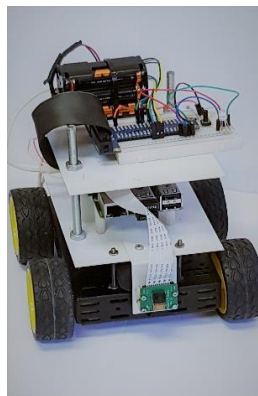


Ανώτατο Εκπαιδευτικό Ίδρυμα Πειραιά
Τεχνολογικού Τομέα
Τ.Ε.Ι. ΠΕΙΡΑΙΑ

Πτυχιακή Εργασία

Θέμα:

**«Ανίχνευση και παρακολούθηση αντικειμένων από μηχανοκίνητο
όχημα με τεχνικές μηχανικής όρασης»**



Φοιτητής: Τσαμούταλης Παναγιώτης

A.M.:36302

Επιβλέπων : ΝΙΚΟΛΑΟΥ ΓΡΗΓΟΡΙΟΣ

ΚΑΘΗΓΗΤΗΣ ΕΦΑΡΜΟΓΩΝ

ΑΙΓΑΛΕΩ 2017

ΔΗΛΩΣΗ ΣΥΓΓΡΑΦΕΑ ΠΤΥΧΙΑΚΗΣ ΕΡΓΑΣΙΑΣ

Ο / Η κάτωθι υπογεγραμμένος / η Τσαμούταλης
Παναγώτης.....,

του Νικολάου....., με αριθμό μητρώου

36302..... φοιτητής / τρια του Τμήματος **Μηχανικών Αυτοματισμού Τ.Ε.** του
Α.Ε.Ι. Πειραιά Τ.Τ. πριν αναλάβω την εκπόνηση της Πτυχιακής Εργασίας μου, δηλώνω ότι
ενημερώθηκα για τα παρακάτω:

«Η Πτυχιακή Εργασία (Π.Ε.) αποτελεί προϊόν πνευματικής ιδιοκτησίας τόσο του συγγραφέα,
όσο και του Ιδρύματος και θα πρέπει να έχει μοναδικό χαρακτήρα και πρωτότυπο
περιεχόμενο.

Απαγορεύεται αυστηρά οποιοδήποτε κομμάτι κειμένου της να εμφανίζεται αυτούσιο ή
μεταφρασμένο από κάποια άλλη δημοσιευμένη πηγή. Κάθε τέτοια πράξη αποτελεί προϊόν
λογοκλοπής και εγείρει θέμα Ηθικής Τάξης για τα πνευματικά δικαιώματα του άλλου
συγγραφέα. Αποκλειστικός υπεύθυνος είναι ο συγγραφέας της Π.Ε., ο οποίος φέρει και την
ευθύνη των συνεπειών, ποινικών και άλλων, αυτής της πράξης.

Πέραν των όποιων ποινικών ευθυνών του συγγραφέα σε περίπτωση που το Ίδρυμα του έχει
απονείμει Πτυχίο, αυτό ανακαλείται με απόφαση της Συνέλευσης του Τμήματος. Η
Συνέλευση του Τμήματος με νέα απόφασης της, μετά από αίτηση του ενδιαφερόμενου, του
αναθέτει εκ νέου την εκπόνηση της Π.Ε. με άλλο θέμα και διαφορετικό επιβλέποντα
καθηγητή. Η εκπόνηση της εν λόγω Π.Ε. πρέπει να ολοκληρωθεί εντός τουλάχιστον ενός
ημερολογιακού δμήνου από την ημερομηνία ανάθεσης της.

Κατά τα λοιπά εφαρμόζονται τα προβλεπόμενα στο άρθρο 18, παρ. 5 του ισχύοντος
Εσωτερικού Κανονισμού.»

Ο Δηλών

Ημερομηνία

Περιεχόμενα

Κεφάλαιο 1 ^ο – Εισαγωγή.....σελ.5	
1.1 Μηχανική όραση.....σελ. 5	
1.2 Περιγραφή λογισμικού – υλικού.....σελ.8	
1.3 Δομή πτυχιακής εργασίας.....σελ.8	
Κεφάλαιο 2 ^ο – Προγενέστερη ερευνητική δραστηριότητα.....σελ.10	
Κεφάλαιο 3 ^ο – Μηχανική Όραση.....σελ.13	
3.1 Εφαρμογές για τη δημιουργία του ανιχνευτή προτύπων.....σελ.13	
3.2 Τοπικά Δυναδικά Πρότυπα.....σελ.14	
3.3 Χαρακτηριστικά Haar.....σελ.21	
3.4 Haar wavelet.....σελ.28	
3.5 Παρακολούθηση και Ανίχνευση αντικειμένων.....σελ.30	
Κεφάλαιο 4 – Υλικό.....σελ.33	
4.1 Raspberry Pi.....σελ.33	
4.2 DC DUAL MOTOR DRIVER 30V 4A V2.....σελ.40	
4.3 DC MOTORS.....σελ.43	
Κεφάλαιο 5ο-Υλοποίηση αυτονόμου οχήματος.....σελ.45	
5.1 Περιγραφή υλικού.....σελ.45	
5.2 ΣΥΝΑΡΜΟΛΟΓΗΣΗ ΑΥΤΟΝΟΜΟΥ ΟΧΗΜΑΤΟΣ σελ.47	
5.3 Διασύνδεση.....σελ.53	
Κεφάλαιο 6 – Λογισμικό.....σελ.57	
6.1 Λειτουργικό σύστημα Rasbian Jessie σελ.57	
6.2 Γλώσσα Python.....σελ.59	
6.3 OpenCV.....σελ.73	

Κεφάλαιο 7 ^ο – Ανάπτυξη αλγορίθμου ακολούθησες αντικειμένου.....σελ.87	
7.1 Περιγραφή- Οδηγίες Λειτουργίας..... σελ.88	
7.2 Μηχανή πεπερασμένων καταστάσεων..... σελ.90	
7.3 Αναλυτική περιγραφή της κατάστασης SD..... σελ.92	
7.4 κινήσεις οχήματος για εντοπισμό μπάλας..... σελ.94	
7.5 Ανάλυση και επεξήγηση κώδικα..... σελ.94	
Κεφάλαιο 8 ^ο - Στόχος και βελτίωση της εργασίας..... σελ.101	
8.1 Επίτευξη στόχου σελ.101	
8.2 Βελτιώσεις σελ.102	
 ΒΙΒΛΙΟΓΡΑΦΙΑ.....σελ.105	
 ΚΩΔΙΚΑΣ.....σελ.111	
 ΠΑΡΑΡΤΗΜΑ σελ.122	

Κεφάλαιο 1^ο - Εισαγωγή

1.1 Μηχανική όραση

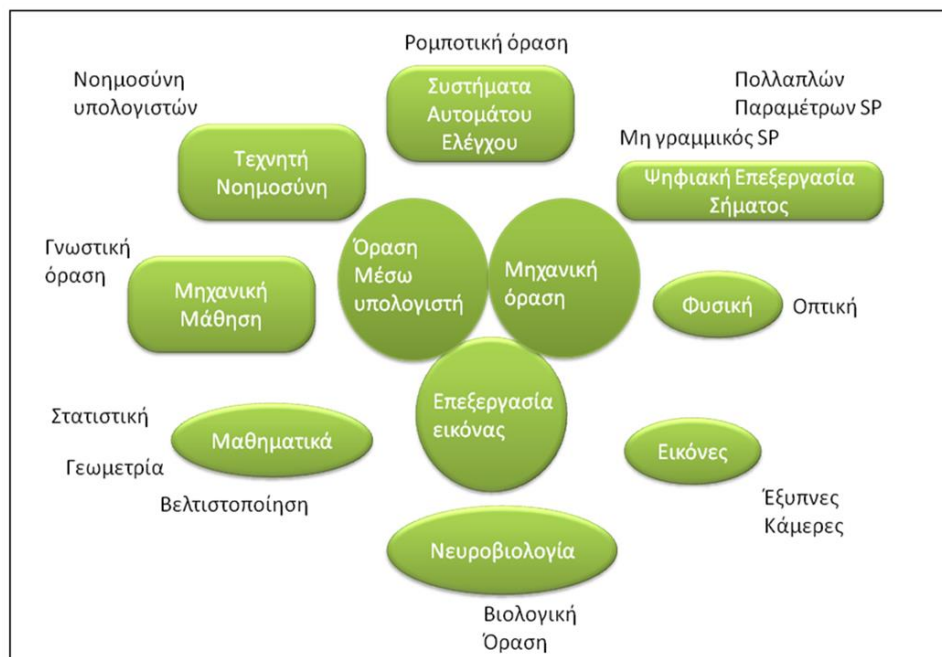
Η μηχανική όραση είναι ένα πεδίο το οποίο περιλαμβάνει μεθόδους για επεξεργασία, ανάλυση και κατανόηση των εικόνων και γενικά δεδομένων μεγάλων διαστάσεων από τον πραγματικό κόσμο έτσι ώστε να παραχθούν αριθμητικά ή συμβολικά δεδομένα ως είδος πληροφορίας[1][2][3][4]. Η διαδικασία αυτή προσομοιώνει την ανθρώπινη όραση με ηλεκτρονικά μέσα, έτσι ώστε η εικόνα να γίνεται αντιληπτή και κατανοητή από υπολογιστικά συστήματα[5]. Η χρήση γεωμετρίας, φυσικής, στατιστικής καθώς και της θεωρίας μάθησης λειτουργούν συνδυαστικά για την αποσαφήνιση της πληροφορίας που περιέχει η εικόνα[6].

1.1.1 Σκοπός πτυχιακής εργασίας

Σκοπός της παρούσας πτυχιακής εργασίας είναι η μελέτη των τεχνικών μηχανικής όρασης με χρήση ηλεκτρονικού μικροϋπολογιστή(raspberry pi 2) και πως αυτή μπορεί να εφαρμοστεί στην ανίχνευση και παρακολούθηση αντικειμένων σε συστήματα πραγματικού χρόνου. Οι καινοτομίες που πηγάζουν από τον χώρο της μηχανικής όρασης βρίσκουν ευρεία εφαρμογή σε πολλούς τομείς της καθημερινής μας ζωής. Η τάση για πλήρη αυτοματοποίηση διεργασιών, που μέχρι πρότινος απαιτούσαν την ανθρώπινη παρουσία, έχει δώσει τεράστια ώθηση στη βελτιστοποίηση της μηχανικής όρασης. Πιλοτικά προγράμματα βρίσκονται σε πλήρη εξέλιξη στην Ολλανδία και στην Ιταλία[7][8] και αφορούν τη μεταφορά επιβατών με μέσα, είτε σταθερής τροχιάς όπως μετρό, τραμ, είτε συμβατικά όπως ταξί λεωφορεία χωρίς την παρουσία οδηγού. Γίνεται άμεσα αντιληπτό ότι η παρουσία ενός συστήματος αυτόματης ανίχνευσης αντικειμένων σε πραγματικό χρόνο αποτελεί την καρδιά των παραπάνω συστημάτων. Στην Ολλανδία αναπτύχθηκε ένα σύστημα με όνομα Phileas με σκοπό την

εξυπηρέτηση επιβατών σε περιοχές με λίγους κατοίκους. Αυτό επιτεύχθηκε με χρήση τραμ, ώστε να καλύψει το κόστος από τη συντήρηση του οχήματος και να μειώσει τη χρήση καυσίμων και ικανοποιεί την ανάγκη των ανθρώπων για μετακίνηση. Η λειτουργία του βασίζεται σε αισθητήρες που λαμβάνουν τις πληροφορίες, τις αποθηκεύουν στον υπολογιστή και βάσει αυτού καθορίζουν την πορεία του. Επίσης, στην Ιταλία δημιουργήθηκε αυτόνομο όχημα χωρίς την ανάγκη οδηγού. Το σύστημα πέτυχε σημαντικά αποτελέσματα κατά τη χρήση του σε ποικίλες περιπτώσεις, όπως σε κίνηση και σε μεγάλους αυτοκινητόδρομους, καθώς δεν ελέγχεται από απόσταση αλλά αντιλαμβάνεται το δρόμο και τη πορεία που θα ακολουθήσει.

1.1.2 Επιστημονικοί τομείς



Σχήμα 1.1 Πεδία εφαρμογών της μηχανικής όρασης.

Στο χώρο της τεχνητής νοημοσύνης ρομποτικά συστήματα λειτουργούν αυτόνομα με χρήση υπολογιστών, οι οποίοι λειτουργούν σαν οπτικοί αισθητήρες και παρέχουν

πληροφορίες για την πλοήγηση τους στο χώρο. Η τεχνητή νοημοσύνη και η μηχανική όραση μοιράζονται επιπλέον διαδικασίες, όπως την αναγνώριση προτύπων και τις τεχνικές μάθησης. Με τη μηχανική όραση ασχολείται η νευροβιολογία όπου χρησιμοποιεί υπολογιστές για να μελετήσει τις διεργασίες πίσω από την οπτική αντίληψη των ανθρώπων αλλά και των ζώων. Επίσης, σχετίζεται η επεξεργασία εικόνας, η ανάλυση εικόνας και η όραση μέσω συστημάτων. Τα πεδία της μηχανικής όρασης και της όρασης μέσω υπολογιστή είναι παρόμοια έχοντας βασικές τεχνικές που χρησιμοποιούνται σε παρόμοιες εφαρμογές.

Η μηχανική όραση με κάποιο τρόπο είναι το αντίστροφο των γραφικών. Ενώ τα γραφικά στον υπολογιστή παράγουν δεδομένα εικόνων από τρισδιάστατα μοντέλα, η μηχανική όραση παράγει τρισδιάστατα μοντέλα από δεδομένα εικόνων. Επιπλέον, υπάρχει μια τάση για συνδυασμό αυτών των δυο για τη διερεύνηση της τεχνολογίας Augmented Reality ή αλλιώς επαυξημένη πραγματικότητα ή αλλιώς ενισχυμένη πραγματικότητα, η οποία είναι μια τεχνολογία που χρησιμοποιείται στα κινητά τηλέφωνα. Η τεχνολογία Augmented Reality επιτρέπει την προβολή ενός φυσικού περιβάλλοντος του οποίου όμως η πραγματικότητα είναι επαυξημένη με την προβολή πληροφοριών αλλά και εικονικών προσώπων ή χώρων σχεδιασμένων μέσα από έναν ηλεκτρονικό υπολογιστή. Ο συνδυασμός της κάμερας με το σύστημα GPS ενός κινητού τηλεφώνου επιτρέπουν την προβολή επιπλέον πληροφοριών για ένα γεωγραφικό σημείο, διαμορφώνοντας ένα επαυξημένο πληροφοριακά τελικό αποτέλεσμα. Παρέχονται πληροφορίες που περιλαμβάνουν κείμενα, ήχους και video και αφορούν ειδικά τη γεωγραφική θέση που βρίσκεται ο χρήστης και στοχεύει η κάμερα του. Οι προβολές δεδομένων είναι δυνατές είτε από τις οθόνες κινητών είτε από ειδικά γυαλιά προβολής Augmented Reality.

Η όραση μέσω υπολογιστή καλύπτει την κύρια τεχνολογία της αυτόματης ανάλυσης της εικόνας, η οποία χρησιμοποιείται σε αρκετούς τομείς. Παραδείγματα τέτοιων εφαρμογών περιλαμβάνουν ηλεκτρονικά ή βιομηχανικά συστήματα, όπως ο έλεγχος διαδικασιών(βιομηχανικά ρομπότ), η πλοήγηση(αυτοκίνητα χωρίς οδηγό), η ανίχνευση γεγονότων(παρακολούθηση από κάμερες ή καταμέτρηση ανθρώπων), οργάνωση πληροφοριών, η μοντελοποίηση αντικειμένων(για ιατρικές περιπτώσεις), η αλληλεπίδραση με ανθρώπους και οι αυτοματοποιημένοι έλεγχοι. Μια προεξέχουσα εφαρμογή χρησιμοποιείται στις ιατρικές εικόνες για εξαγωγή πληροφοριών με σκοπό την διάγνωση και θεραπεία του ασθενή.

1.2 Περιγραφή λογισμικού - υλικού

Στόχος της εργασίας αυτής είναι η ανίχνευση και παρακολούθηση αντικειμένων από μηχανοκίνητο όχημα με τεχνικές μηχανικής όρασης. Για την ανίχνευση αντικειμένων σε εικόνες πραγματικού χρόνου χρησιμοποιήσαμε τη βιβλιοθήκης ανοιχτού κώδικα Open Source Computer Vision (*Open CV*) σε γλώσσα Python[9][10][11]. Το μικροϋπολογιστικό σύστημα που αποτελεί και την καρδιά της υλοποίησης είναι το Raspberry Pi2[12] με προσαρτημένη την κάμερα raspberry pi camera module και το μηχανοκίνητο όχημα είναι το Multi-Chassis - 4WD Kit [13]. Διακρίνουμε τις εξής φάσεις στην πορεία ανάπτυξης της παρούσας πτυχιακής: τη φάση της συναρμολόγησης του οχήματος, τη φάση της διασύνδεσης οχήματος μικροϋπολογιστή και τερματικών συσκευών(οθόνη, πληκτρολόγιο, ποντίκι, σερβοκινητήρες, κάμερα) και τέλος τη φάση της ανάπτυξης της εφαρμογής μας για την ανίχνευση του αντικειμένου(επιλέξαμε ένα μπαλάκι χρώματος μπλε).

1.3Δομή πτυχιακής εργασίας

Η παρούσα πτυχιακή εργασία αποτελείται από οκτώ κεφάλαια. Στο τρέχον κεφάλαιο έγινε μία σύντομη αναφορά στον τομέα της μηχανικής όρασης και στην ιστορία των αυτοκινούμενων οχημάτων. Στο δεύτερο κεφάλαιο γίνεται περιγραφή σε προγενέστερες ερευνητικές δραστηριότητες στον τομέα της μηχανικής όρασης. Το επόμενο κεφάλαιο περιγράφει το θεωρητικό υπόβαθρο που για την αναγνώριση προτύπων στον τομέα της μηχανικής όρασης. Το τέταρτο κεφάλαιο ασχολείται με τους μικροϋπολογιστές και συγκεκριμένα με το μοντέλο Raspberry Pi2 το οποίο είναι και η καρδιά του οχήματος μας. Στο πέμπτο κεφάλαιο γίνεται αναφορά στα διάφορα μέρη από τα οποία αποτελείται το όχημα μας, τον τρόπο της τελικής συναρμολόγησης τους καθώς και λεπτομέρειες για τα τεχνικά χαρακτηριστικά του. Το λογισμικό που αναπτύξαμε για την εκτέλεση του αλγορίθμου βρίσκεται στο έκτο κεφάλαιο. Σε αυτό το κεφάλαιο επίσης γίνεται περιγραφή του λειτουργικού συστήματος Rasbian, ο τρόπος εγκατάστασης του και η διαδικασία εκτέλεσης

προγράμματος σε γλώσσα python. Στο έκτο κεφάλαιο περιγράφουμε τον αλγόριθμο ακολούθησης του αντικειμένου και αναλύεται ο κώδικας λειτουργίας του οχήματος. Η πτυχιακή ολοκληρώνεται με το όγδοο και τελευταίο κεφάλαιο στο οποίο περιγράφουμε κατά πώς και πόσο επιτεύχθηκε ο στόχος μας, αλλά και πίες βελτιώσεις μπορούν να κάνουν την εργασία μας καλύτερη.

Κεφάλαιο 2^ο – Προγενέστερη ερευνητική δραστηριότητα

Οι Navneet Dalal και Bill Triggs[14] μελέτησαν το ζήτημα των χαρακτηριστικών σετ για την αναγνώριση αντικειμένων και συγκεκριμένα δοκιμάζοντας έναν γραμμικό SVM βασισμένο στην αναγνώριση ανθρώπων. Αφού μελέτησαν τους περιγραφείς, που χρησιμοποιούσαν τις άκρες και τη κλίση μάς έδειξαν πειραματικά ότι οι HOG περιγραφείς ξεπερνούν σε σημαντικό βαθμό τις υπάρχουσες μεθόδους για αναγνώριση ανθρώπων. Μελέτησαν κάθε βήμα του υπολογισμού για την απόδοση συμπεραίνοντας ότι έχοντας συγκεκριμένη κλίση, καλό προσανατολισμό, καλή χωρική τοποθέτηση και καλής ποιότητας ρύθμιση στην αντίθεση, λαμβάνουμε καλύτερα αποτελέσματα. Η νέα προσέγγισή τους οδηγεί σε σχεδόν τέλεια διαχώριση μεταξύ της αρχικής βάσης δεδομένων των πεζών από το MIT και για το λόγο αυτό εισήγαγαν μια μεγαλύτερη, η οποία περιλαμβάνει πάνω από 1800 εικόνες που συμπεριλαμβάνουν ανθρώπους σε ένα μεγάλο εύρος από πόζες και παρασκήνια.

Οι Constantine Papageorgiou και Tomaso Poggio[15] στην εργασία τους παρουσιάζουν γενικά συστήματα για αναγνώριση αντικειμένων σε διάφορες εικόνες. Τα συστήματα αυτά λαμβάνουν τον μεγαλύτερο όγκο πληροφοριών τους από μια παρουσίαση η οποία περιγράφει μια κλάση αντικειμένου με χρήση τοπικών, προσανατολισμένων και διαφορετικών εντάσεων μεταξύ διαφορετικών περιοχών, που μπορούν να υπολογιστούν όπως ένας μετασχηματισμός wavelet[16]. Αυτή η προσέγγιση της εκμάθησης παράγει ένα μοντέλο εκπαιδεύοντας ένα Μηχανισμό Διανυσμάτων Υποστήριξης και χρησιμοποιώντας μεγάλη ποικιλία από θετικές και αρνητικές εικόνες. Οι ερευνητές παρουσιάζουν τα αποτελέσματα σε πρόσωπα, ανθρώπους και αυτοκίνητα χρησιμοποιώντας την ίδια αρχιτεκτονική. Επί πρόσθετα, σύγκριναν το πώς η εργασία τους επηρεάζει την επίδοση της ανίχνευσης λαμβάνοντας υπόψη διαφορετικές εφαρμογές που χρησιμοποιούν σαν κύρια χαρακτηριστικά τα εικονοστοιχεία. Επιπλέον, περιγράφουν μια εφαρμογή του συστήματος σε πραγματικό χρόνο για την ανίχνευση ατόμων προς βοήθεια των οδηγών.

Οι Henry Schneiderman και Takeo Kanade[17] στην εργασία τους περιγράφουν ένα σύστημα που ανιχνεύει αντικείμενα και χρησιμοποιείται σε πρόσωπα και αυτοκίνητα οποιουδήποτε σχήματος, με οποιαδήποτε πόζα ή σε οποιαδήποτε τοποθεσία. Για να αντιμετωπίσουν την ευρύτητα στον προσανατολισμό των αντικειμένων ο ανιχνευτής χρησιμοποιεί πολλαπλούς ταξινομητές, που ο καθένας εκτείνεται σε διαφορετικό εύρος των προσανατολισμών των αντικειμένων και καθένας από αυτούς καθορίζει εάν το αντικείμενο βρίσκεται στο προκαθορισμένο παράθυρο της εικόνας. Για να ανιχνεύσουν το αντικείμενο

που μπορεί να υπάρχει με οποιοδήποτε μέγεθος ή σε οποιαδήποτε θέση οι ταξινομητές χρειάζεται να τη σαρώσουν όλη την εικόνα.

Κάθε ταξινομητής βασίζεται στα στατιστικά των μερών. Κάθε κομμάτι είναι ένας μετασχηματισμός από ένα υποσύνολο wavelet που αντιστοιχεί σε ένα διακριτό σύνολο τιμών. Αυτά τα κομμάτια δημιουργήθηκαν για την χρήση διάφορων συνδυασμών στο χώρο, τη συχνότητα και το προσανατολισμό. Στη δημιουργία κάθε ταξινομητή συνέλλεξαν τιμές από τα αντιπροσωπευτικά δείγματα των εικόνων που περιλαμβάνουν το αντικείμενο και από αυτές που δε το περιέχουν. Εκπαίδευσαν κάθε ταξινομητή, ώστε να έχει το μικρότερο σφάλμα στο σετ της εκπαίδευσης χρησιμοποιώντας το Adaboost με το Confidence-Weighted Prediction(Shapire και Singer,1999).Στην ανίχνευση κάθε ταξινομητής υπολογίζει κάθε μέρος των τιμών που περιλαμβάνονται στο παράθυρο της εικόνας και ψάχνει για τις σχετικές πιθανότητες συνθηκών της κλάσης. Ο ταξινομητής, έπειτα, αποφασίζει εφαρμόζοντας πιθανότητες. Για αποτελεσματικότητα ο ταξινομητής αξιολογεί αυτή τη πιθανότητα σε στάδια. Σε κάθε στάδιο συγκρίνει κάθε κομμάτι της με ένα κατώφλι και παίρνει την απόφαση για το αν θα τερματίσει την λειτουργία-επισημαίνοντας ότι η εικόνα δε περιέχει το αντικείμενο-ή συνεχίζοντας την περαιτέρω λειτουργία. Ο ανιχνευτής ορίζει τα στάδια να ξεκινήσουν από χαμηλής ποιότητας αξιολόγηση. Ο ανιχνευτής αντικειμένων που δημιούργησαν πετυχαίνει αξιόπιστη και επαρκή ανίχνευση ανθρώπινων προσώπων και ανθρώπων μέσα σε αυτοκίνητα.

Οι Paul Viola, Michael J. Jones και Daniel Snow[18] στην εργασία τους για την ανίχνευση πεζών περιγράφουν ένα σύστημα που συνδυάζει τις τιμές των πληροφοριών της εικόνας με τις πληροφορίες της κίνησης. Χρησιμοποιούν έναν αλγόριθμο ανίχνευσης, όπου σαρώνει με έναν ανιχνευτή κάθε 2 πλαίσια του βίντεο. Ο ανιχνευτής δημιουργήθηκε μέσω του Adaboost δίνοντάς τους το πλεονέκτημα να λαμβάνουν πληροφορίες για τη κίνηση και την εμφάνιση ώστε να ανιχνεύσουν έναν κινούμενο πεζό. Παλαιότερες εργασίες χρησιμοποιούν, είτε τις πληροφορίες από τη κίνηση είτε τις πληροφορίες από την εμφάνιση αλλά η δική τους εργασία είναι η πρώτη που συνδυάζει και τις δυο πηγές πληροφοριών σε έναν ανιχνευτή. Η εφαρμογή λειτουργεί στα 4 πλαίσια το δευτερόλεπτο, ανιχνεύει πεζούς σε μικρή κλίμακα(περίπου 20x15 pixels) και έχει μικρό ρυθμό σφάλματος.

Η προσέγγισή των ερευνητών στηρίζεται στη εργασία για ανίχνευση των Viola και Jones. Οι νέες συνεισφορές σε αυτήν την εργασία περιέχουν την ανάπτυξη της παρουσίας της κινούμενης εικόνας, η οποία είναι εξαιρετικά αποτελεσματική και την εφαρμογή ενός

συστήματος για ανίχνευση πεζών, το οποίο λειτουργεί σε χαμηλής ποιότητας εικόνες κάτω από δύσκολες συνθήκες(όπως η βροχή και το χιόνι).

Κεφάλαιο 3^ο – Μηχανική Όραση

3.1 Εφαρμογές για τη δημιουργία του ανιχνευτή προτύπων

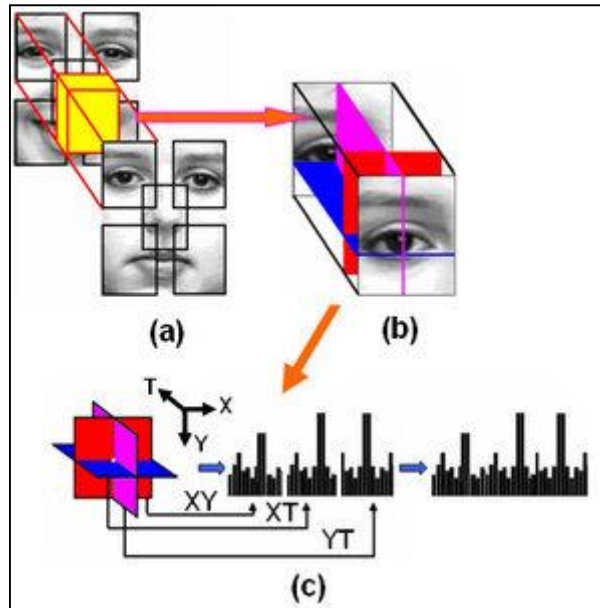
Η διαδικασία για τη δημιουργία του ανιχνευτή προτύπων περιλαμβάνει δύο κύρια στάδια: την εκπαίδευση και την ανίχνευση. Στο OpenCV υπάρχουν δύο εφαρμογές για τη δημιουργία του ανιχνευτή. Η πρώτη εφαρμογή είναι το *opencv_haartraining* και η δεύτερη το *opencv_traincascade*. Η δεύτερη εφαρμογή, που είναι και η πιο πρόσφατη και υποστηρίζει ταυτόχρονα τα τοπικά δυαδικά πρότυπα (Local Binary Patterns- LBP) [19] και τα Haar[20] (βλέπε Κεφ.3.2 , 3.3) χαρακτηριστικά.

Η διαδικασία της εκπαίδευσης του ανιχνευτή περιλαμβάνει βήματα τα οποία εξαρτώνται αποκλειστικά από το αντικείμενο που πρόκειται να ανιχνευτεί. Στο πρώτο βήμα συγκεντρώνουμε εικόνες που περιέχουν το αντικείμενο ανίχνευσης και εικόνες που δε το περιέχουν. Κατόπιν, αποθηκεύουμε σε διαφορετικούς φακέλους αυτές που το περιλαμβάνουν(θετικές) και σε έναν άλλο φάκελο αυτές που δε το περιέχουν(αρνητικές). Η ευστοχία στην ανίχνευση του αντικειμένου αυξάνεται προφανώς με το πλήθος των αρχείων εικόνας που έχουμε συγκεντρώσει. Στη συνέχεια με ένα απλό πρόγραμμα επεξεργασίας εικόνας όπως *imageclipper*, *objectmarker*, κ.α. επισημαίνουμε στις θετικές φωτογραφίες την περιοχή ενδιαφέροντος που θέλουμε να γίνει η ανίχνευση. Κατόπιν, δημιουργούμε ένα ανυσματικό(*vector*) αρχείο βασισμένο στις φωτογραφίες, όπου έχει βρεθεί η περιοχή ενδιαφέροντος. Τέλος, αρχίζει η διαδικασία εκπαίδευσης της οποίας ο χρόνος εκτέλεσης ποικίλει ανάλογα με το πλήθος των φωτογραφιών που έχουμε στη διάθεσή μας, παράγοντας ένα αρχείο *xml* το οποίο και καλούμε μέσα από την εφαρμογή μας για την τελική επιθυμητή ανίχνευση.

3.2 Τοπικά Δυαδικά Πρότυπα

Τα τοπικά δυαδικά πρότυπα (ΤΔΠ) είναι ένα είδος χαρακτηριστικού που χρησιμοποιούμε στην ταξινόμηση για την μηχανική όραση. Παρατηρήθηκε ότι είναι σημαντικό για την αναγνώριση της σύστασης και επί πρόσθετα συνδυάστηκε με περιγραφείς που συνδυάζουν τα Ιστογράμματα Προσανατολισμένης Κλίσης, οι οποίοι βελτίωσαν την απόδοση σε σύνολα δεδομένων.

Τα ΤΔΠ στην απλούστερη μορφή τους δημιουργούνται χωρίζοντας το παράθυρο της εικόνας σε κελιά. Έπειτα, γίνεται σύγκριση κάθε εικονοστοιχείου στο κελί με καθένα διπλανό του με φορά του ρολογιού ή και αντίστροφα. Εάν η τιμή του κεντρικού εικονοστοιχείου είναι μεγαλύτερη από αυτήν του διπλανού του τότε σημειώνουμε 1. Εάν είναι μικρότερη τότε σημειώνουμε 0. Αυτό μας δίνει έναν 8bit δυαδικό αριθμό, τον οποίο μπορούμε να μετατρέψουμε σε δεκαδικό προς διευκόλυνσή μας. Υπολογίζουμε, ύστερα, το ιστόγραμμα του κελιού έχοντας υπόψη την συχνότητα που εμφανίζεται ο κάθε αριθμός. Προαιρετικά μπορούμε να κανονικοποιήσουμε το ιστόγραμμα και τελικά ενώνουμε τα κανονικοποιημένα και μη ιστογράμματα όλων των κελιών. Αυτό μας δίνει ως αποτέλεσμα το χαρακτηριστικό διάνυσμα του παραθύρου. Μπορούμε να το επεξεργαστούμε χρησιμοποιώντας τα SVM ή κάποιο άλλο παρόμοιο αλγόριθμο. Στο σχήμα 3.1 βλέπουμε μια περιγραφή των χαρακτηριστικών προσώπου με χρήση τοπικών δυαδικών προτύπων.



Σχήμα 3.1- Σχηματική περιγραφή σε χαρακτηριστικά προσώπου με χρήση τοπικών δυαδικών προτύπων.

3.2.1 Τοπικά δυαδικά πρότυπα στο χώρο

Η βασική ιδέα για την ανάπτυξη των χαρακτηριστικών ήταν η χρήση δισδιάστατου χώρου τον οποίο περιγράφουμε με δυο συμπληρωματικές μετρήσεις: με τα τοπικά χωρικά πρότυπα και την αντίθεση σε κλίμακα του γκρι. Ο πρωτότυπος τελεστής των ΤΔΠ[21] έχοντας αρχικά γειτονικά εικονοστοιχεία διαστάσεων 3×3 συμβολίζει τα εικονοστοιχεία της εικόνας, περνώντας τη κεντρική τιμή κάθε εικονοστοιχείου από ένα κατώφλι και δίνει αποτέλεσμα έναν δυαδικό αριθμό. Το ιστόγραμμα αυτών των $2^8 = 256$ διαφορετικών συμβολισμών μπορούμε να το χρησιμοποιήσουμε σαν ένα περιγραφέα της σύστασης[22].

Ο τελεστής ΤΔΠ προεκτάθηκε για να χρησιμοποιεί γειτονικά τμήματα διαφορετικών μεγεθών[23]. Χρησιμοποιώντας γειτονικό τμήμα κυκλικού μεγέθους με τιμές διγραμμικής παρεμβολής σε μη ακέραιες τιμές συντεταγμένων μας επέτρεψε την ύπαρξη οποιασδήποτε γωνίας ή αριθμού εικονοστοιχείων στο τμήμα αυτό. Οι διακυμάνσεις στη κλίμακα του γκρι του τοπικού τμήματος μπορούν να

χρησιμοποιηθούν ως μια συμπληρωματική μέτρηση της αντίθεσης. Στα ακόλουθα σύμβολα (P,R), τα οποία χρησιμοποιούνται για τα εικονοστοιχεία του γειτονικού τμήματος, θεωρούμε ότι έχουμε P δείγματα σε κύκλο ακτίνας R.

Άλλη επέκταση του αρχικού τελεστή είναι ο ορισμός των αποκαλούμενων ομοιόμορφων προτύπων, τα οποία χρησιμοποιούνται στη μείωση του μήκους του χαρακτηριστικού διανύσματος και εφαρμόζουν έναν απλό περιστροφικό περιγραφέα. Ένα τοπικό δυαδικό πρότυπο καλείται ομοιόμορφο, εάν περιέχει το πολύ δυο μεταβάσεις bits από 0 σε 1 και το αντίθετο. Για παράδειγμα, τα πρότυπα 01110000 και 11001111 ανήκουν στην κατηγορία των ομοιόμορφων, μιας και σε αυτά παρατηρούνται δύο μεταβάσεις. Αντίθετα, δεν ανήκουν στην κατηγορία των ομοιόμορφων πρότυπα τύπου 00000000, στα οποία δεν εμφανίζονται μεταβάσεις και πρότυπα όπου εμφανίζονται 4 και 6 μεταβάσεις, όπως στα 11001001 και 01010010.

Ο Ojala[24] παρατήρησε στα πειράματά του με τις εικόνες σύστασης ότι στα ομοιόμορφα πρότυπα λαμβάνουμε υπόψη λιγότερο από 90% των προτύπων, όταν χρησιμοποιούμε γειτονικό τμήμα (8,1) και περίπου το 70% σε γειτονικό τμήμα(16,2). Κάθε θέση(LBP code) μπορεί να θεωρηθεί ως μια μικροδομή. Αρχικά, κωδικοποιούμε αυτές τις θέσεις οι οποίες περιλαμβάνουν διαφορετικά είδη γωνιών, σημείων κτλ.

Οι ακόλουθοι συμβολισμοί χρησιμοποιούνται από τον τελεστή: $LBP_{P,R}^{u2}$. Ο δείκτης αντιπροσωπεύει το γειτονικό τμήμα(P,R). Ο δείκτης u2 αντιπροσωπεύει τα ομοιόμορφα πρότυπα και όλα τα εναπομείναντα που έχουν έναν συμβολισμό. Αφού λάβουμε την εικόνα μας $f_1(x,y)$ ορίζουμε το ιστόγραμμα:

$$H_i = \sum_{x,y} I\{f_1(x,y) = i\}, i = 0, \dots, n-1$$

στο οποίο το n είναι ο αριθμός των διαφορετικών συμβολισμών που παράγονται από τον χειριστή και $I\{A\}$ είναι 1 αν το A είναι αληθές και 0 αν είναι ψευδές.

Όταν τα τμήματα της εικόνας των οποίων τα ιστογράμματα που θα συγκριθούν έχουν διαφορετικά σχήματα, τότε τα κανονικοποιούμε με τον ακόλουθο τρόπο:

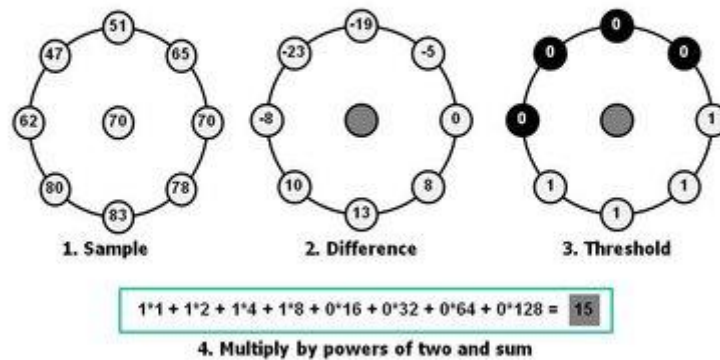
$$N_i = \frac{H_i}{\sum_{j=0}^{n-1} H_j}$$

Στο σχήμα 3.2 έχουμε ένα παράδειγμα του υπολογισμού των τοπικών δυαδικών προτύπων.

$$LBP_{P,R} = \sum_{p=0}^{P-1} 8(g_p - g_c)2^p \quad \begin{matrix} s(x) = 1, x \geq 0 \\ s(x) = 0, \text{άλλου} \end{matrix}$$

The value of the LBP code of a pixel (x_c, y_c) is given by:

$$LBP_{P,R} = \sum_{p=0}^{P-1} s(g_p - g_c)2^p \quad s(x) = \begin{cases} 1, & \text{if } x \geq 0; \\ 0, & \text{otherwise.} \end{cases}$$



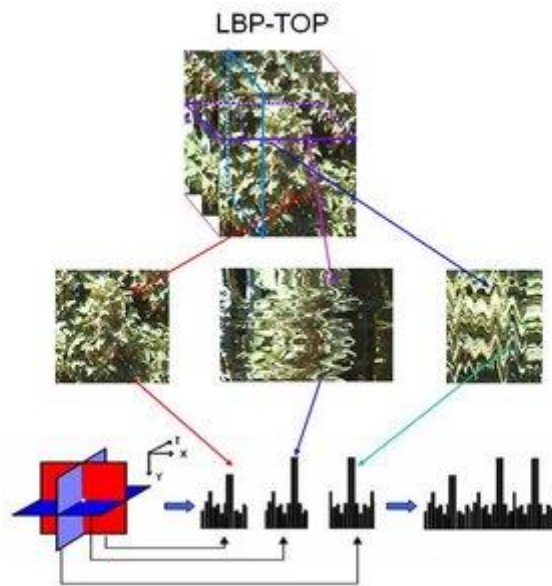
Σχήμα 3.2- Σχήμα παραδείγματος υπολογισμού των LBP

3.2.2 Χωροχρονικά τοπικά δυαδικά πρότυπα

Ο αρχικός τελεστής ορίστηκε για να λειτουργεί με πληροφορίες του χώρου. Για αυτόν τον σκοπό προτάθηκε ο αυτοαποκαλούμενος τελεστής VLBP (Volume Local Binary Pattern) [24]. Η ιδέα πίσω από αυτόν είναι να βλέπουμε τη δυναμική σύσταση των στοιχείων σαν ένα μέρος συνόλου στο τρισδιάστατο χώρο με άξονες (X,Y,Z) όπου X και Y υποδηλώνουν τις συντεταγμένες του χώρου και Z υποδηλώνει τον δείκτη του πλαισίου. Επομένως, η γειτονιά του κάθε εικονοστοιχείου ορίζεται από έναν τρισδιάστατο χώρο. Έπειτα παρόμοια με τα ΤΔΠ στο πεδίο του χώρου μπορούμε να ορίσουμε το σύνολο των μικροδομών και να τον εξάγουμε σε ιστογράμματα.

Για να απλουστεύσουμε τον υπολογισμό των VLBP προτάθηκε ένας τελεστής ο οποίος βασίζεται στις εμφανίσεις σε έναν τριορθογώνιο χώρο(LBP-TOP) των τοπικών δυαδικών προτύπων. Το LBP-TOP θεωρεί τρεις ορθογώνιους χώρους: XY, XT και YT και συνενώνει τα τοπικά δυαδικά πρότυπα συνδυάζοντας στατιστικές σε αυτές τις τρεις κατευθύνσεις. Τα κυκλικά γειτονικά τμήματα γενικεύονται σε ελλειπτική δειγματοληψία για να τα προσαρμόσουμε στα στατιστικά του χωροχρόνου.

Το σχήμα 3.4 δείχνει ένα παράδειγμα για μια εικόνα τριών επιπέδων. Το XY επίπεδο αντιπροσωπεύει τις πληροφορίες εμφάνισης ενώ το XT δίνει μια οπτική εικόνα της σειράς συναρτήσεως της αλλαγής του χρόνου και το YT περιγράφει την κίνηση της στήλης στο χρόνο. Οι ΤΔΠ κώδικες εξάγονται από όλα τα εικονοστοιχεία των επιπέδων XY, XT, YT όπου και υπολογίζονται και ενώνονται σε ένα ιστόγραμμα. Με αυτήν την αναπαράσταση η δυναμική σύσταση κωδικοποιείται από την εμφάνιση (XY-LBP) και τις δυο χωροχρονικές (XT-LBP και YT-LBP) .



Σχήμα 3.4 - ΤΔΠ από τρία ορθογώνια επίπεδα

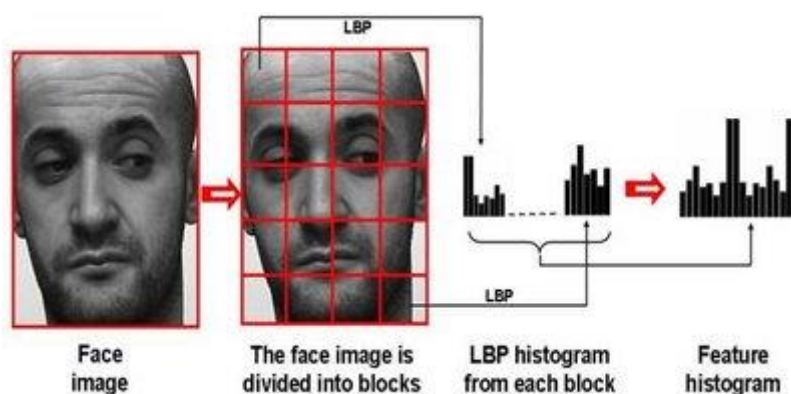
Άρα, χρειάζεται να ρυθμίσουμε διαφορετικές παραμέτρους στο χώρο και στο χρόνο. Στα επίπεδα των XT και YT μπορούμε να προσδιορίσουμε διαφορετικές ακτίνες, έτσι ώστε να δειγματοληπτήσουμε γειτονικά σημεία στο χώρο και στο χρόνο. Γενικότερα, οι ακτίνες

στους άξονες X, Y και T και ο αριθμός των γειτονικών σημείων στα επίπεδα XY ,XT και YT μπορούν να είναι διαφορετικά.

3.2.3 Αναγνώριση προσώπου χρησιμοποιώντας τα ΤΔΠ

Στην προσέγγιση των ΤΔΠ για αναγνώριση της σύστασης οι εμφανίσεις του ΤΔΠ κώδικα της εικόνας συλλέγονται σε ένα ιστόγραμμα. Η αναγνώριση έπειτα γίνεται υπολογίζοντας απλές ομοιότητες μεταξύ των ιστογραμμάτων. Παρόλα αυτά, έχοντας παρόμοια προσέγγιση για αναπαράσταση προσώπων σε εικόνες έχουμε σαν αποτέλεσμα την απώλεια χωρικής πληροφορίας και ως εκ τούτου χρειάζεται να κωδικοποιηθεί η πληροφορία των στοιχείων. Ένας τρόπος για να το πετύχουμε είναι η χρήση των ΤΔΠ περιγραφών για να κατασκευάσουμε περιγραφείς προσώπου αφού αρχικά χωρίσουμε την εικόνα σε τμήματα και συνδυάσουμε τους περιγραφείς του κάθε τμήματος σε έναν γενικό περιγραφέα.

Η βασική μεθοδολογία των ΤΔΠ, που βασίζεται στην αναγνώριση προσώπου προτάθηκε από τον Ahonen[25] και είναι η ακόλουθη: η εικόνα προσώπου χωρίζεται σε τοπικές περιοχές και οι ΤΔΠ περιγραφείς εξάγονται από κάθε περιοχή ξεχωριστά και έπειτα ενώνονται για να σχηματίσουν έναν γενικό περιγραφέα του προσώπου, όπως φαίνεται στο σχήμα 3.5.



Σχήμα 3.5 - Αναγνώριση προσώπου μέσω των ΤΔΠ

Το ιστόγραμμα περιέχει την περιγραφή του προσώπου σε τρία διαφορετικά επίπεδα: έχουμε το επίπεδο με τους συμβολισμούς των ΤΔΠ που περιέχουν τη πληροφορία για τα πρότυπα, έχουμε τους συμβολισμούς ενός τμήματος όπου παράγονται οι πληροφορίες των τοπικών τμημάτων της εικόνας και έχουμε τα τοπικά ιστογράμματα που ενώνονται για σχηματίσουν έναν γενικό περιγραφέα του προσώπου.

Σημειώνουμε, ότι, όταν χρησιμοποιούμε μεθόδους βασισμένες σε ιστογράμματα, οι περιοχές δε χρειάζεται να είναι τετραγωνικές ούτε χρειάζεται να είναι του ίδιου σχήματος ή του ίδιου μεγέθους. Επίσης, δεν είναι απαραίτητο να καλύπτουν όλη την εικόνα ενώ είναι δυνατόν τοπικά τμήματα να καλύπτουν το ένα το άλλο. Η περιγραφή προσώπων περιγράφεται από τη μέθοδο σε χωροχρονικό πεδίο[26].

Από την δημοσίευση των ΤΔΠ για αναγνώριση προσώπων η μεθοδολογία έχει ήδη καθιερωθεί στις έρευνες και εφαρμογές της αναγνώρισης και ανάλυσης προσώπων. Ένα αξιοσημείωτο παράδειγμα είναι το σύστημα αναγνώρισης προσώπων που δεν επηρεάζεται από τον φωτισμό ο οποίος προτάθηκε από τον Li[27]. Ο Zhang[28] πρότεινε την εξαγωγή των ΤΔΠ χαρακτηριστικών από τις εικόνες φιλτράροντας την εικόνα προσώπου με 40 Gabor φίλτρα διαφορετικής κλίμακας και προσανατολισμού τα οποία μας παρέχουν εκπληκτικά αποτελέσματα. Οι Hadid και Pietikainen[29] χρησιμοποίησαν χωροχρονικά ΤΔΠ για αναγνώριση προσώπου και φύλου από ακολουθίες βίντεο ενώ ο Zhao[30] υιοθέτησε την προσέγγιση ΤΔΠ-TOP για οπτική αναγνώριση του λόγου πετυχαίνοντας εξαιρετικά αποτελέσματα χωρίς να είναι επιρρεπές σε λάθη κατά τη κατάτμηση στα κινούμενα χείλη.

3.2.4 Επεκτάσεις και εφαρμογές

Η μεθοδολογία των ΤΔΠ έχει επιφέρει σημαντική πρόοδο στην ανάλυση των στοιχείων της εικόνας. Χρησιμοποιείται ευρέως από όλον τον κόσμο και ερευνητικά αλλά και σε εφαρμογές. Λόγω της διακριτικής του ισχύος και της υπολογιστικής του απλότητας, η μέθοδος είναι πετυχημένη σε πολλά ζητήματα της μηχανικής όρασης τα οποία η επιστημονική κοινότητα δε τα θεωρούσε καν ζητήματα σε επίπεδο στοιχείων της εικόνας, όπως είναι η ανάλυση αντικειμένων και κίνησης[31][32].

Προς αύξηση της χρήσης των ΤΔΠ, προτάθηκαν ποικίλες επεκτάσεις και τροποποιήσεις. Για παράδειγμα ο Liao[33] πρότεινε ΤΔΠ, τα οποία χρησιμοποιούν τα πιο συχνά μοτίβα των ΤΔΠ για να βελτιώσουν την ευστοχία της αναγνώρισης. Η χρήση των περιγραφών περιοχών (SIFT βλ κεφ 4.4.2) σε ποικίλα ζητήματα της μηχανικής όρασης έχουν λάβει μεγάλο ενδιαφέρον. Για αυτόν τον σκοπό, ένας νέος περιγραφέας συνδυάζοντας την ευχέρεια του SIFT[34][35] και των ΤΔΠ προτάθηκε από τον Heikkila[36] στο οποίο ένας συμμετρικά κεντραρισμένο ΤΔΠ χρησιμοποιήθηκε για να αντικαταστήσει τον χειριστή κλίσεων που χρησιμοποιείται από τους SIFT. Οι Maenpaa και Pietikainen[37] πρότειναν έναν χρωματικό ΤΔΠ και ερεύνησαν τη χρήση του χρώματος και των στοιχείων στην ταξινόμηση. Ο συνδυασμός των ΤΔΠ με τα Gabor χαρακτηριστικά διερευνήθηκαν από τους Tan[38] και Wang[39].

Η πρώτη μέθοδος βασισμένη στα στοιχεία για εξαγωγή του παρασκήνιου προτάθηκε από τους Heikkila-Pietikainen[40]. Κάθε εικονοστοιχείο μοντελοποιείται ως μια ομάδα προσαρμοστικών ιστογραμμάτων ΤΔΠ, τα οποία υπολογίζονται πάνω από μια κυκλική περιοχή γύρω από το εικονοστοιχείο. Η μέθοδος αυτή φάνηκε να μην επηρεάζεται από τις αλλαγές του φωτισμού, την ποικιλία του παρασκήνιου και από την εισαγωγή ή την εξαγωγή αντικειμένων στο παρασκήνιο και είναι ικανή για επεξεργασία σε πραγματικό χρόνο. Ένας αλγόριθμος βασισμένος στα ΤΔΠ αναπτύχθηκε για να αντιμετωπίσει αλλαγές στο φωτισμό σε συστήματα ταυτοποίησης προσώπων[41]. Η χρήση των ΤΔΠ στην αναγνώριση ηλικίας μέσω προσώπου ερευνήθηκε από τον Wang[42]. Άλλες παρόμοιες εφαρμογές των ΤΔΠ για επίλυση αυτών των ζητημάτων έχουν προταθεί πρόσφατα.

Επί πρόσθετα στην αναγνώριση προσώπων και εκφράσεων τα ΤΔΠ χρησιμοποιούνται σε εφαρμογές, όπως της βιομέτρησης, δηλαδή τον καθορισμό της ίριδας του ματιού και τα δαχτυλικά αποτυπώματα.

3.3 Χαρακτηριστικά Haar

Τα Haar χαρακτηριστικά είναι γνωρίσματα της ψηφιακής εικόνας τα οποία χρησιμοποιούνται στην αναγνώριση αντικειμένων. Οφείλουν το όνομά τους στην ομοιότητα

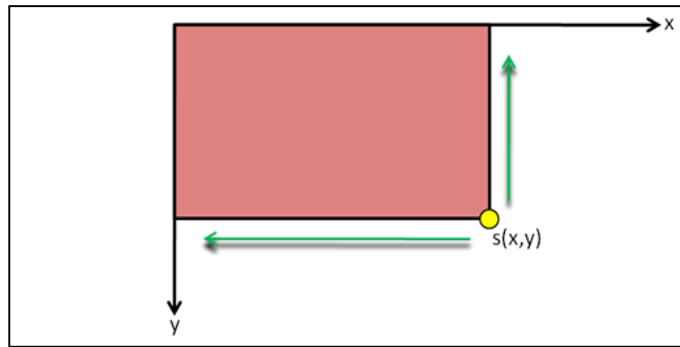
με τα Haar wavelets[16] και χρησιμοποιήθηκαν στην πρώτη εφαρμογή[43] ανίχνευσης προσώπων σε πραγματικό χρόνο.

Ο πυρήνας της βάσης ενός ανιχνευτή Haar για αναγνώριση αντικειμένων είναι τα χαρακτηριστικά Haar. Αυτά αντί να χρησιμοποιούν τις τιμές έντασης του εικονοστοιχείου, χρησιμοποιούν τις αλλαγές στις τιμές της αντίθεσης μεταξύ διπλανών ορθογώνιων ομάδων εικονοστοιχείων. Οι διακυμάνσεις της αντίθεσης μεταξύ αυτών των ομάδων χρησιμοποιούνται για να καθορίσουν τα τμήματα με τις υψηλές τιμές και τα τμήματα με τις χαμηλές. Δυο ή τρεις διπλανές περιοχές από ομάδες εικονοστοιχείων με μια σχετική διακύμανση σε τιμές αντίθεσης, διαμορφώνουν τα χαρακτηριστικά Haar. Τέτοια χαρακτηριστικά φαίνονται στα σχήματα του επόμενου υποκεφαλαίου και χρησιμοποιούνται για ανίχνευση σε εικόνα[44]. Στην ομάδα εικονοστοιχείων την οποία μελετάμε μπορούμε να τροποποιήσουμε τα Haar χαρακτηριστικά της αυξάνοντας ή μειώνοντας το μέγεθος, ώστε να είναι δυνατόν να αναγνωρίσουμε αντικείμενα διάφορων μεγεθών.

3.3.1 Ολοκλήρωμα εικόνας

Η ακέραιη εικόνα ή ο αθροιστικός πίνακας δημοσιεύτηκε το 1984 άλλα η χρήση του στην όραση μέσω υπολογιστή παρουσιάστηκε από τους Viola και Jones. Η ακέραιη εικόνα είναι ένας γρήγορος και αποτελεσματικός τρόπος για να υπολογίσουμε το άθροισμα των τιμών των εικονοστοιχείων σε μια δεδομένη εικόνα ή σε ένα ορθογώνιο κομμάτι της. Η κύρια χρήση της είναι να υπολογίζουμε τη μέση αντίθεση.

Για να δημιουργήσουμε το ολοκλήρωμα εικόνας, αρχικά δημιουργούμε τον αθροιστικό πίνακα. Οποιοδήποτε σημείο x,y στον πίνακα αντιπροσωπεύεται από την αντίστοιχη τιμή του. Αυτή η τιμή είναι ενδιαφέρουσα, διότι αποτελείται από το άθροισμα όλων των τιμών των εικονοστοιχείων προς τη κατεύθυνση x και τη κατεύθυνση y συμπεριλαμβανομένης και της τιμής στη θέση x,y , όπως φαίνεται στο παρακάτω σχήμα 3.6.

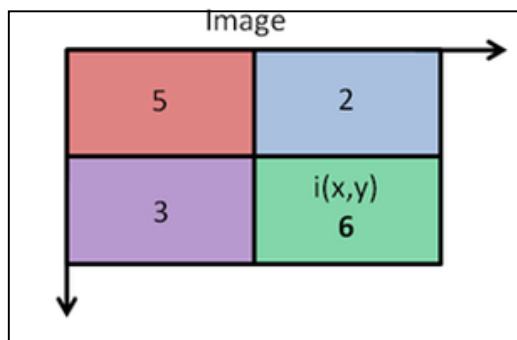


Σχήμα 3.6 Αρχικό σημείο για υπολογισμό του ολοκληρώματος εικόνας

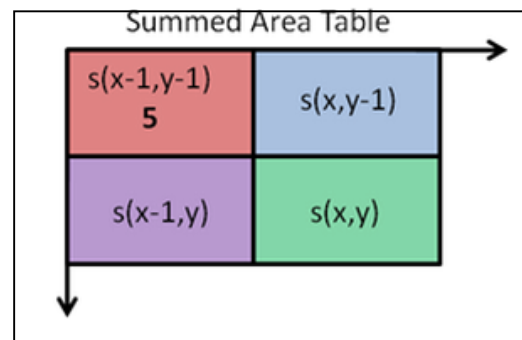
Το πλεονέκτημα του αθροιστικού πίνακα είναι ότι μπορούμε να το δημιουργήσουμε με μόνο ένα πέρασμα της δεδομένης εικόνας. Η τιμή στη θέση x,y του αθροιστικού πίνακα υπολογίζεται από την ακόλουθη εξίσωση:

$$s[x,y] = i[x,y] + s[x-1,y] + s[x,y-1] - s[x-1,y-1]$$

Παίρνουμε την αρχική τιμή του εικονοστοιχείου $i(x,y)$ της εικόνας και προσθέτουμε τις τιμές που βρίσκονται ακριβώς από πάνω και ακριβώς αριστερά από αυτό το εικονοστοιχείο του αθροιστικού πίνακα στα $s(x-1,y)$ και $s(x,y-1)$ και τελικά αφαιρούμε τη τιμή που βρίσκεται ακριβώς πάνω αριστερά του $i(x,y)$ του αθροιστικού πίνακα στο $s(x-1,y-1)$.



Σχήμα 3.6α Τιμές εικονοστοιχείων
της εικόνας



Σχήμα 3.6β Αθροιστικός πίνακας

Η θέση $s(x-1, y)$ αντιπροσωπεύεται από την τιμή 8. Είναι το άθροισμα που αντιπροσωπεύεται από τις τιμές που βρίσκονται αριστερά, πάνω και συμπεριλαμβανομένης αυτής της θέσης,

όπως φαίνεται και στη παρακάτω εικόνα. Πιο συγκεκριμένα, στην περίπτωση της τιμής 8 της θέσης $s(x-1,y)$

έχουμε $5+3=8$. Η ίδια διαδικασία ακολουθείται για τη θέση $s(x, y-1)$, όπου έχουμε $5+2=7$. Έτσι, εμφανίζεται ολοκληρωμένος ο πίνακας ως εξής:

Image	
5	2
3	$i(x,y)$ 6

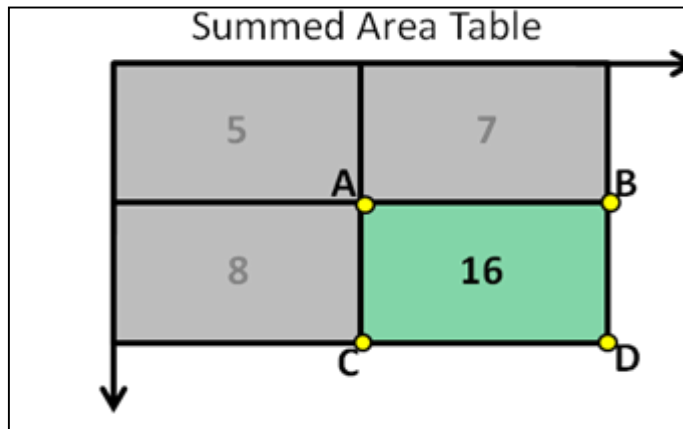
Σχήμα 3.6c Τιμές των εικονοστοιχείων

Summed Area Table	
$s(x-1,y-1)$ 5	$s(x,y-1)$ 7
$s(x-1,y)$ 8	$s(x,y)$ 16

Σχήμα 3.6d Αθροιστικός πίνακας
με υπολογισμένες τιμές

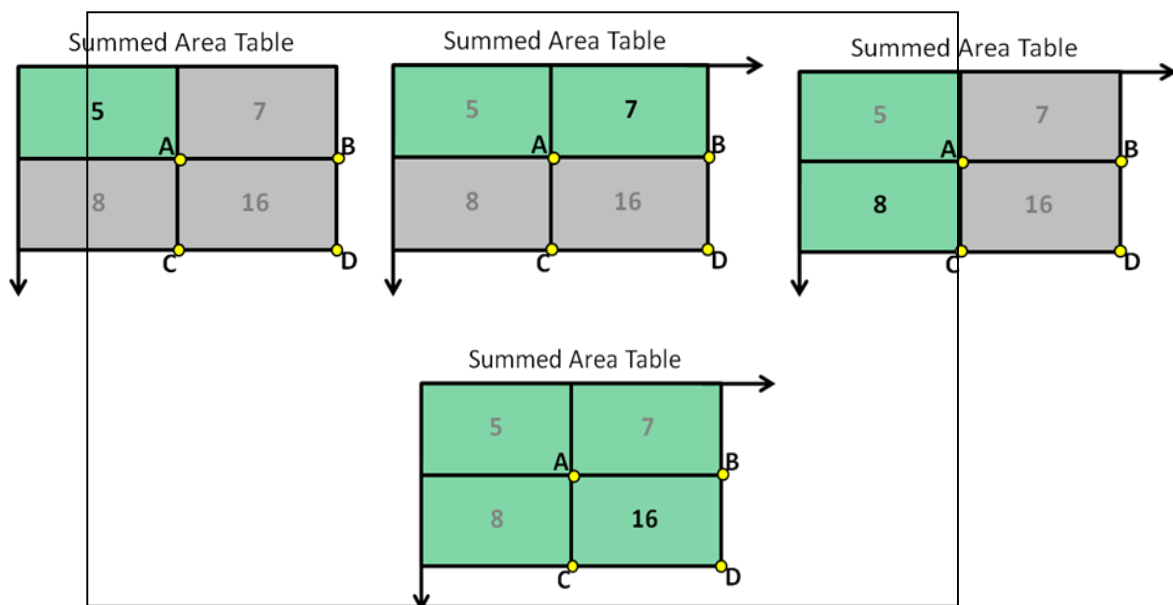
Για την θέση $s(x,y)$ έχουμε $6+8+7-5=16$. Και αν προσθέσουμε τις τέσσερις τιμές των εικονοστοιχείων της εικόνας μας έχουμε $5+2+3+6=16$.

Έτσι τώρα μπορούμε να υπολογίσουμε σχεδόν αμέσως το άθροισμα των εικονοστοιχείων σε μια ορθογώνια περιοχή που είναι το υποσύνολο της αρχικής εικόνας. Χρησιμοποιούμε τέσσερις τιμές από τον αθροιστικό πίνακα, δηλαδή τέσσερις περιοχές που αναφέρονται στον πίνακα και χρησιμοποιούμε την εξίσωση $i(x',y')=s(A) + s(D) -s(B)-s(C)$. Στο παρακάτω τμήμα του πίνακα θέλουμε να υπολογίσουμε τη μαρκαρισμένη πράσινη περιοχή.



Σχήμα 3.6e Αθροιστικός πίνακας

Η τιμή 16 είναι το άθροισμα όλων των τετραγώνων αλλά εμείς θέλουμε μόνο τη πράσινη περιοχή. Αρχικά, έχουμε για το $s(A)$ το οποίο έχει τιμή 5. Το $s(B)$ έχει τιμή 7. Παρόμοια και ο $s(C)$ έχει 8 και τελικά για το $s(D)$ είναι το άθροισμα όλων των τιμών(16).



Σχήμα 3.6f Πίνακες για τον υπολογισμό του κάθε στοιχείου

Έτσι, εφαρμόζοντας τον τύπο $i(x',y')=s(A) + s(D) -s(B)-s(C)$ έχουμε $5+16-7-8=6$.

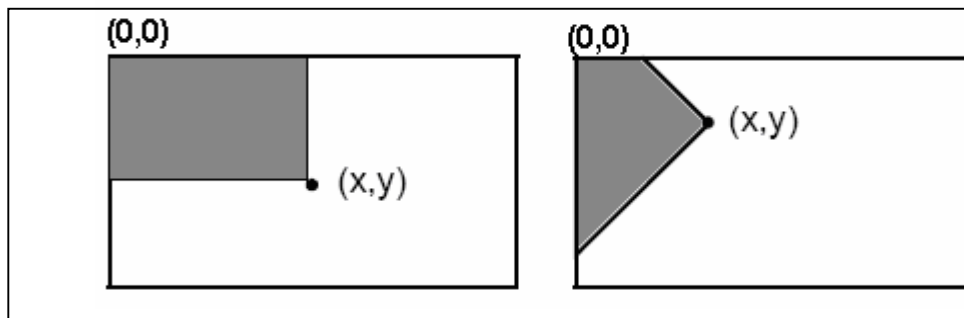
Επομένως, πηγαίνουμε στην αρχική μας εικόνα με τις τιμές των εικονοστοιχείων και βλέπουμε ότι η κάτω δεξιά τιμή είναι 6, οπότε οι υπολογισμοί μας έγιναν σωστά.

Εάν συμβολίσουμε με $A[x,y]$ την αρχική μας εικόνα και $AI[x,y]$ το ολοκλήρωμά της, τότε το $AI[x,y]$ υπολογίζεται όπως φαίνεται από την παρακάτω εξίσωση.

$$AI[x,y] = \sum_{x' \leq x, y' \leq y} A(x', y')$$

Τα χαρακτηριστικά, τα οποία περιστρέφονται κατά 45 μοίρες, προτάθηκαν από τους Lienhard-Maydt και χρειάζονται μια ενδιάμεση παρουσίαση που ονομάζεται περιστρεμμένο ολοκλήρωμα εικόνας ή περιστρεφόμενη αθροιστική βοηθητική εικόνα[9]. Η περιστρεμμένη εικόνα υπολογίζεται βρίσκοντας το άθροισμα της έντασης των εικονοστοιχείων, τα οποία βρίσκονται σε γωνία 45 μοιρών στα αριστερά πάνω από την x τιμή και κάτω από την y . Έτσι, αν $A[x,y]$ είναι η αρχική μας εικόνα και $AR[x,y]$ το περιστρεμμένο ολοκλήρωμα εικόνας, τότε αυτή υπολογίζεται όπως απεικονίζεται στην παρακάτω εξίσωση και απεικονίζεται από στο σχήμα 3.8.

$$AR[x,y] = \sum_{x' \leq x, x' = |y-y'|} A(x', y')$$



Σχήμα 3.7 αθροισμένη περιοχή του ολοκληρώματος εικόνας και αθροισμένη περιοχή του περιστρεμμένου ολοκληρώματος εικόνας

Χρειάζεται ένας υπολογισμός για κάθε μία εικόνα ξεχωριστά. Χρησιμοποιώντας την κατάλληλη εικόνα και λαμβάνοντας υπόψη τη διαφορά, μπορούμε να υπολογίσουμε το χαρακτηριστικό οποιουδήποτε μεγέθους. Έτσι, ο υπολογισμός του χαρακτηριστικού είναι γρήγορος και επαρκής. Αυτό σημαίνει επιπλέον, ότι υπολογίζοντας χαρακτηριστικά ποικίλων

μεγεθών, έχουμε την ίδια διαδικασία που εκτελείται για ένα χαρακτηριστικό δύο ή τριών εικονοστοιχείων. Η ανίχνευση διάφορων μεγεθών του ίδιου αντικειμένου εγείρει ίση προσπάθεια και ίσο χρόνο μ' αυτόν που είναι αναγκαίος και σε αντικείμενα παρόμοιου μεγέθους, διότι η κλιμάκωση δεν απαιτεί επιπρόσθετες ενέργειες[45].

3.3.2 Διαδοχικοί ανιχνευτές

Παρότι ο υπολογισμός ενός χαρακτηριστικού είναι εξαιρετικά επαρκής και γρήγορος, ο υπολογισμός όλων των χαρακτηριστικών (180.000) που περιέχονται σε ένα τμήμα 24x24 μιας εικόνας είναι πρακτικά αδύνατος[46][47]. Ευτυχώς, απαιτείται ένα μικρό τμήμα αυτών των χαρακτηριστικών για να καθορίσουμε, εάν το τμήμα αυτό της εικόνας περιέχει το επιθυμητό αντικείμενο[48]. Για να εξαλείψουμε, όσο το δυνατόν περισσότερα τμήματα της εικόνας, χρησιμοποιούμε μόνο μερικά από τα χαρακτηριστικά τα οποία είναι ικανά να προσδιορίσουν τα αντικείμενα που θέλουμε να ανιχνεύσουμε. Η διαδικασία συνεχίζεται αυξάνοντας τον αριθμό των χαρακτηριστικών, τα οποία χρησιμοποιούνται για την ανάλυση των τμημάτων της εικόνας σε κάθε βήμα.

Η διαδοχή των ανιχνευτών επιτρέπει να αναλυθούν τμήματα της εικόνας με τη μεγαλύτερη πιθανότητα για όλα τα χαρακτηριστικά Haar, τα οποία διακρίνουν τα αντικείμενα. Επίσης, επιτρέπει την διαφοροποίηση της ακρίβειας του ανιχνευτή. Ο ρυθμός σφάλματος είναι δυνατόν να αυξηθεί μειώνοντας τον αριθμό των βημάτων και αντιστρόφως. Οι Viola και Jones κατάφεραν να επιτύχουν 95% ακρίβεια για την ανίχνευση ανθρώπινου προσώπου χρησιμοποιώντας μόνο 200 δείγματα χαρακτηριστικών[45]. Χρησιμοποιώντας υπολογιστή 2GHz, ο Haar ταξινομητής μπορούσε να ανιχνεύσει ανθρώπινα πρόσωπα με ρυθμό τουλάχιστον 5 πλαίσια ανά δευτερόλεπτο[43].

3.4 Haar wavelet

Στα μαθηματικά τα Haar wavelet είναι μια ακολουθία απλουστευμένων τετραγωνικών συναρτήσεων οι οποίες αν συνδυαστούν δημιουργούν τη βάση των wavelet. Η ανάλυση σε κυμάτια παρουσιάζει ομοιότητες με την ανάλυση Fourier[49], η οποία αναπαριστά μία συνάρτηση ορισμένη σ' ένα διάστημα με όρους μίας ορθοκανονικής συνάρτησης βάσης. Η Σειρά Haar προτάθηκε το 1909 από τον Alfred Haar[50], ο οποίος χρησιμοποίησε αυτές τις συναρτήσεις για να δώσει ένα παράδειγμα ενός ορθοκανονικού συστήματος σχετικά με τον χώρο των τετραγωνικά ολοκληρώσιμων συναρτήσεων στο διάστημα $[0,1]$.

Το Haar wavelet είναι η απλούστερη γενική μορφή ενός wavelet. Το μειονέκτημα του είναι ότι δεν είναι συνεχής συνάρτηση (σχήμα 3.9) και ως εκ τούτου δεν είναι διαφορίσιμη. Μας δίνει τη δυνατότητα να ανιχνεύσουμε σφάλματα σε συστήματα με απότομες μεταβολές[51].

Η πρωταρχική συνάρτηση του Haar wavelet ορίζεται ως:

$$\psi(t) = 1, 0 \leq t \leq 1/2$$

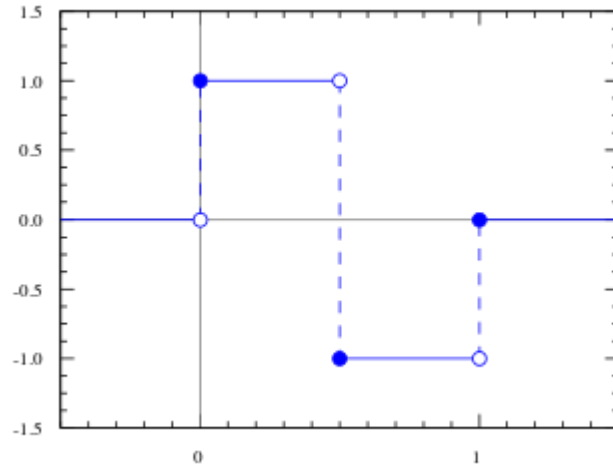
$$\psi(t) = -1, 1/2 \leq t \leq 1$$

$$\psi(t) = 0, \text{ αλλού}$$

Η κλιμακωτή συνάρτηση ορίζεται ως:

$$\psi(t) = 1, 0 \leq t \leq 1$$

$$\psi(t) = 0, \text{ αλλού}$$



Σχήμα 3.9 Haar wavelet

Ένα παράδειγμα υλοποίησης και χρήσης των wavelets αυτών είναι οι σύγχρονες κάμερες οι οποίες μπορούν να παράγουν εικόνες με ανάλυση πάνω από 10Mpixel και για την αποθήκευσή τους απαιτούν συμπίεση. Για τη διαδικασία της συμπίεσης αυτής χρησιμοποιούμε τα Haar wavelet. Η βασική ιδέα είναι να μεταφέρουμε την εικόνα μας σε έναν πίνακα στον οποίο κάθε στοιχείο του να αντιπροσωπεύει τα εικονοστοιχεία της εικόνας. Για παράδειγμα, αν έχουμε μια εικόνα μεγέθους 256x256 pixels, την αποθηκεύουμε σε έναν πίνακα αντίστοιχων διαστάσεων. Οι jpeg εικόνες μπορούν να συμπιεστούν χωρίζοντας την αρχική εικόνα σε μικρότερες με τη χρήση του δύο διαστάσεων μετασχηματισμού Haar. Η εξίσωση αυτή είναι η $B_n = H_n A_n H_n^T$ όπου A_n είναι ο $n \times n$ πίνακας και H_n είναι ο μετασχηματισμός n -σημείων. Η αντίστροφη διαδικασία ορίζεται από τον τύπο $A_n = H_n^T B_n H_n$.

Δε χρειαζόμαστε πολλαπλασιασμούς με τη χρήση του μετασχηματισμού αλλά μόνο προσθέσεις. Ακόμη, στον πίνακα Haar, υπάρχουν πολλά μηδενικά στοιχεία, κάτι που έχει σαν αποτέλεσμα να μειώνουμε το χρόνο του υπολογισμού. Τα μήκη της εισόδου και της εξόδου είναι τα ίδια. Παρόλα αυτά, θέλουμε να είναι δύναμη του 2 δηλαδή $N = 2^k$.

3.5 Παρακολούθηση και Ανίχνευση αντικειμένων

Η ανίχνευση αντικειμένων είναι μια ουσιαστική και σημαντική διαδικασία σε οποιοδήποτε ευφύες σύστημα μηχανικής όρασης καθώς παρέχει βασικές πληροφορίες για τη σημασιολογική κατανόηση των εικόνων. Βρίσκει εφαρμογή σε μια μεγάλη γκάμα προϊόντων.

3.5.1 Δυσκολίες στην ανίχνευση

Συναντάμε πολλές προκλήσεις στην ανίχνευση αντικειμένων μέσω μηχανικής όρασης οι οποίες και απαιτούν περαιτέρω ανάπτυξη των συστημάτων. Αναφορικά μπορούμε να πούμε ότι τα μακρινά αντικείμενα σε φωτογραφίες φαίνονται μικροί σε σχέση με τη πραγματικότητα. Αυτό σημαίνει ότι έχουμε να δουλέψουμε με λιγότερα εικονοστοιχεία στην εικόνα μας οπότε αυξάνεται το σφάλμα στην ανίχνευσή τους. Επιπλέον η ανίχνευση πρέπει να γίνεται γρήγορα. Επίσης το περιβάλλον στο οποίο υπάρχουν τα αντικείμενα επηρεάζει το αποτέλεσμα μας. Παρά τις δυσκολίες η ανίχνευση αντικειμένων παραμένει ενεργή σε ερευνητικό επίπεδο στην μηχανική όραση και αρκετές λύσεις έχουν προταθεί για τα ζητήματα που αναφέρθηκαν.

3.5.2 Λειτουργία ανίχνευσης

Στον τομέα της εκμάθησης μηχανής[52] ο στόχος της ταξινόμησης είναι η χρησιμοποίηση των χαρακτηριστικών του αντικειμένου δηλαδή τις χρωματικές τιμές της εικόνας(Red-Green-Blue) για να πετύχουμε την αναγνώριση του. Ένας γραμμικός ταξινομητής το πετυχαίνει αυτό βασισμένος στον γραμμικό συνδυασμό των χαρακτηριστικών

του αντικειμένου ώστε να τοποθετηθεί σε μια κατηγορία. Τα χαρακτηριστικά αυτά είναι γνωστά σαν χαρακτηριστικές τιμές και γενικά παρουσιάζονται σαν διάνυσμα με ονομασία διάνυσμα τιμών. Έτσι για την επίλυση αυτού χρησιμοποιούμε τα Ιστογράμματα Προσανατολισμένης Κλίσης τα οποία είναι περιγραφείς που χρησιμοποιούνται στην όραση μέσω υπολογιστή και επεξεργασία της εικόνας για την ανίχνευση αντικειμένων. Η τεχνική αυτή αθροίζει τις εμφανίσεις του προσανατολισμού των κλίσεων σε περιοχές της φωτογραφίας. Υπάρχουν παρόμοιοι μέθοδοι αλλά διαφοροποιούνται στο ότι υπολογίζονται σε ομοιόμορφα κελιά(περιοχές) και χρησιμοποιούν τοπική κανονικοποίηση της αντίθεσης για να βελτιώσουν την ακρίβεια.

Η ουσιώδης σκέψη στους HOG περιγραφείς είναι ότι η εμφάνιση του αντικειμένου οποιουδήποτε σχήματος στην εικόνα μπορεί να περιγραφεί από την κατανομή της έντασης των κλίσεων ή από την κατεύθυνση των γωνιών. Η εφαρμογή αυτών μπορεί να επιτευχθεί χωρίζοντας την εικόνα σε μικρότερα τμήματα που αναφέρονται ως κελιά και σε κάθε ένα από αυτά να δημιουργούμε ένα ιστόγραμμα κλίσης σύμφωνα με την κατεύθυνση των γωνιών των εικονοστοιχείων(pixel). Για βελτιωμένη ακρίβεια τα τοπικά ιστογράμματα μπορούν να κανονικοποιηθούν κατά αντίθεση υπολογίζοντας την ένταση σε ένα μεγαλύτερο τμήμα της εικόνας το οποίο ονομάζουμε μπλοκ και μετά χρησιμοποιώντας την τιμή αυτή να κανονικοποιήσουμε όλα τα κελιά μέσα στο μπλοκ. Αυτή η κανονικοποίηση έχει σαν αποτέλεσμα μικρότερες αλλοιώσεις στις αλλαγές του φωτισμού ή των σκιών.

Οι HOG περιγραφείς διατηρούν σημαντικά πλεονεκτήματα σε σχέση με άλλες μεθόδους. Εφόσον ο HOG λειτουργεί σε κελιά η μέθοδος κρατάει αναλλοίωτους τους γεωμετρικούς και φωτομετρικούς μετασχηματισμούς εκτός από την κατεύθυνση των αντικειμένων. Τέτοιες αλλαγές εμφανίζονται μόνο σε μεγαλύτερες περιοχές. Επιπλέον, η χωρική δειγματοληψία, η καλή δειγματοληψία κατεύθυνσης και η κανονικοποίηση φωτομέτρησης αγνοεί την μεμονωμένη κίνηση του σώματος εφόσον διατηρεί μια ορθή στάση. Οι HOG περιγραφείς είναι επομένως κατάλληλοι για ανίχνευση ανθρώπων σε εικόνες.

Η χρήση Ιστογραμμάτων Προσανατολισμένης Κλίσης εφαρμόζονται εάν αρχικά έχουμε λάβει το πρώτο καρέ από το βίντεο μας ώστε να ξεκινήσει η διαδικασία ανίχνευσης. Αρχικά εφαρμόζεται η διαδικασία της κανονικοποίησης γάμμα η οποία μας ελέγχει την κωδικοποίηση και την αποκωδικοποίηση των τιμών φωτεινότητας. Στην ίδια περίπτωση έχουμε και την κανονικοποίηση χρώματος η οποία επιτρέπει αναγνώριση με βάση το χρώμα για να αντισταθμίσει τυχόν μεταβολές από τον φωτισμό. Στο επόμενο στάδιο έχουμε την

κλίση της εικόνας όπου έχουμε μια κατευθυντική αλλαγή στην ένταση ή στο χρώμα της. Έπειτα στην όλη διαδικασία έχουμε την δημιουργία κελιών ιστογράμματος. Κάθε εικονοστοιχείο (pixel) μέσα στο κελί τοποθετείται σε ένα κανάλι ιστογράμματος σύμφωνα με τον προσανατολισμό του που βασίζεται στις τιμές που υπολογίστηκαν από τη διαβάθμιση κλίσης. Στη συνέχεια ακολουθεί το ιστόγραμμα ισοστάθμισης της εικόνας. Και τελικά έχουμε τον Μηχανισμό Διανυσμάτων Υποστήριξης(Support Vector Machine) όπου είναι μοντέλο το οποίο συσχετίζεται με αλγόριθμους οι οποίοι μας βοηθάνε να αναλύουμε δεδομένα και να αναγνωρίζουμε μοντέλα που χρησιμοποιούνται στην ταξινόμηση.

Κεφάλαιο 4 – Υλικό

Βασικός σκοπός της παρούσας πτυχιακής εργασίας είναι ο σχεδιασμός και η υλοποίηση ενός ενσωματωμένου συστήματος βασισμένο σε ένα σύγχρονο υπολογιστικό σύστημα. Ως πρώτο στάδιο μπορούμε να διακρίνουμε την επιλογή του υλικού (hardware) που αποτελεί και το ενσωματωμένο σύστημα. Το στάδιο αυτό είναι ιδιαίτερα σημαντικό καθώς εδώ επιλέγονται όλα τα επιμέρους τμήματα που θα πρέπει στη συνέχεια να προγραμματιστούν και να λειτουργήσουν μαζί. Η επιλογή του υλικού αποτελεί το πρώτο βήμα στην διαδικασία σχεδιασμού οποιουδήποτε ενσωματωμένου συστήματος.

4.1 Raspberry Pi

Το ρόλο του μικροελεγκτή (microcontroller) στο αυτόματο μηχανοκίνητο όχημα μας παίζει το raspberry pi2 (Σχήμα 4.1). Το raspberry pi2 αποτελεί έναν υπολογιστή σε μέγεθος πιστωτικής κάρτας. Αναπτύχθηκε από την Raspberry Pi Foundation και σκοπό είχε την προώθηση της διδασκαλίας της επιστήμης των υπολογιστών σε σχολεία. Είναι μια συσκευή η οποία παρουσιάζει πολλές δυνατότητες και είναι ικανή για λειτουργίες που συνήθως γίνονται από ένα προσωπικό ηλεκτρονικό υπολογιστή, όπως προβολή ταινιών, παιχνίδια, περιήγηση στο διαδίκτυο. Λόγω του μεγέθους και των δυνατοτήτων του χρησιμοποιείται σε πολλά project [53]. Ο λόγος που επιλέξαμε αυτήν το μικροϋπολογιστικό σύστημα είναι η πληθώρα των λειτουργιών που μπορεί να εκτελέσει αλλά και το χαμηλό κόστος του.

4.1.1 Ιστορική αναδρομή

Το 2006, στο πανεπιστήμιο του Cambridge, μια ομάδα εργαζομένων στο τμήμα Computer Laboratory ανησυχούσε για το μειωμένο ενδιαφέρον των φοιτητών στην πληροφορική, καθώς και τις περιορισμένες γνώσεις τους. Η ομάδα αυτή, που περιλάμβανε μεταξύ άλλων τους Eben Upton, Rob Mullins, Jack Lang, και Alan Mycroft, σκέφτηκε πως η λύση θα ήταν ένας εξαιρετικά μικρός και προσιτός υπολογιστής. Με ένα τέτοιο οικονομικό σύστημα, θα ήταν δυνατόν να διδάσκεται πρακτικά η πληροφορική στα σχολεία, και να έχει

μεγαλύτερο ενδιαφέρον για τους μαθητές. Ενώ η ομάδα σχεδίασε αρκετά αρχικά πρωτότυπα του Raspberry Pi, πρέπει να αναλογιστούμε πως το 2006 υπήρχαν σημαντικοί περιορισμοί, λόγω του υψηλού κόστους και χαμηλής ισχύος των επεξεργαστών για mobile συσκευές. Σταδιακά, όμως, με την κυκλοφορία του πρώτου iPhone το 2007 και την μετέπειτα επέλαση των smartphones, το κόστος της τεχνολογίας άρχισε να μειώνεται αρκετά για να γίνει βιώσιμη η υλοποίηση του Raspberry Pi.

Το 2008, με τη συνεργασία των Pete Lomas και David Braben δημιουργήθηκε το φιλανθρωπικό ίδρυμα Raspberry Pi Foundation[54]. Τρία χρόνια αργότερα κυκλοφόρησε το πρώτο Raspberry Pi, με τα Model A, Model A+ και Model B. Τα μοντέλα αυτά διέθεταν επεξεργαστή ARMv6k στα 700 MHz, 256MB RAM, κάρτα γραφικών Broadcom VideoCore IV, και κατανάλωση από 1 έως 3.5 watt, ενώ η αποθήκευση των δεδομένων γινόταν σε κάρτες SD, SDHC και MicroSD. Μόνο το Raspberry Pi Model B, μέσα σε δύο χρόνια από όταν βγήκε στη μαζική παραγωγή, πούλησε πάνω από δύο εκατομμύρια κομμάτια. Στη συνέχεια κυκλοφόρησαν οι εκδόσεις Model B rev 2 και Model B+ με 512MB RAM. Τον Οκτώβριο του 2014, οι συνολικές πωλήσεις προσέγγιζαν τα 4 εκατομμύρια.

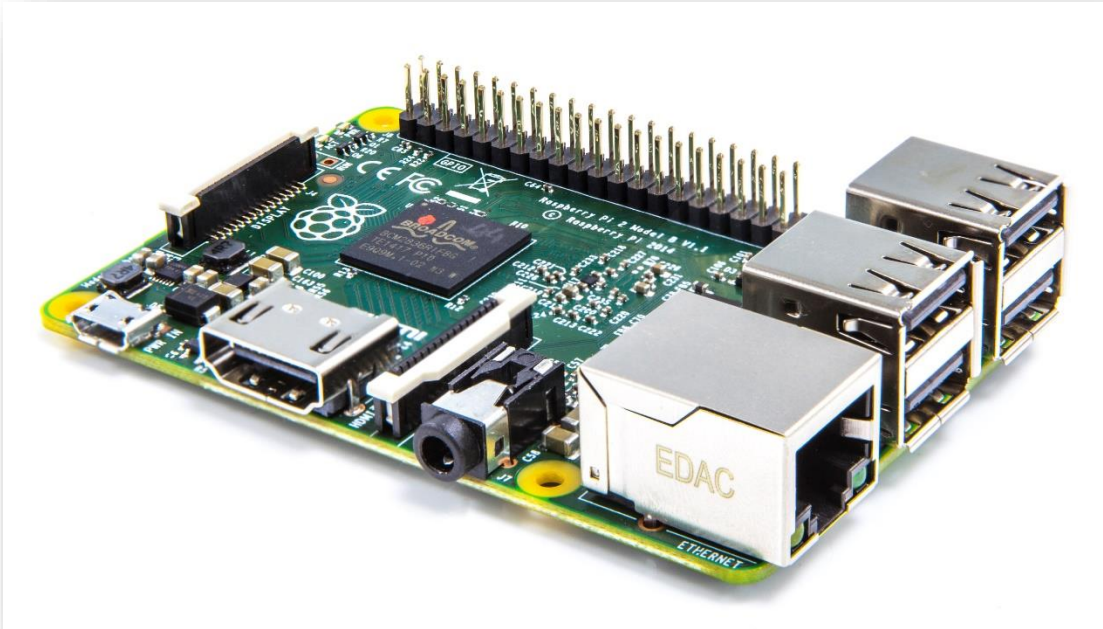
4.1.2 Raspberry Pi2

Η παρούσα έκδοση κυκλοφόρησε το Φεβρουάριο του 2015 με το όνομα Raspberry PiGeneration 2 Model B. Το νέο αυτό μοντέλο έχει αναβαθμισμένη μνήμη RAM μεγέθους 1GB και εξαπλάσια ταχύτητα σε σχέση με τον προκατόχο του, με τον τετραπύρηνο Cortex-A7 (ARMv7), ενώ έχει και βελτιωμένη διπύρηνη κάρτα γραφικών.

Το Raspberry Pi2 (σχήμα τάδε) εκτός από τις παραπάνω βελτιώσεις διαθέτει:

- 4 θύρες USB
- 40 ακροδέκτες εισόδου-εξόδου(GPIO)
- θύρα διασύνδεσης πολυμέσων υψηλής ευκρίνειας (High Definition Multimedia Interface – HDMI)
- υποδοχή για δικτύωση μέσω καλωδίου Ethernet
- μικτή έξοδο υποδοχής 3.5mm για ήχο και για εικόνα.
- διεπαφή για υποδοχή κάμερας (CSI)

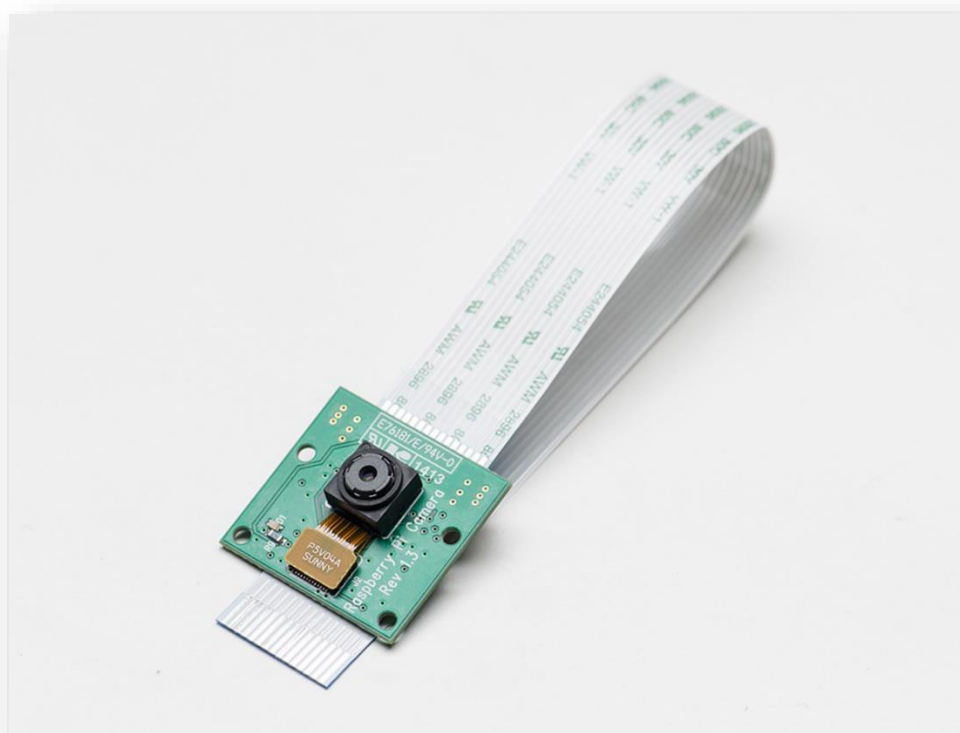
- διεπαφή για σύνδεση εξωτερικής οθόνης (DSI)
- υποδοχή για κάρτα μνήμης τύπου Micro SD



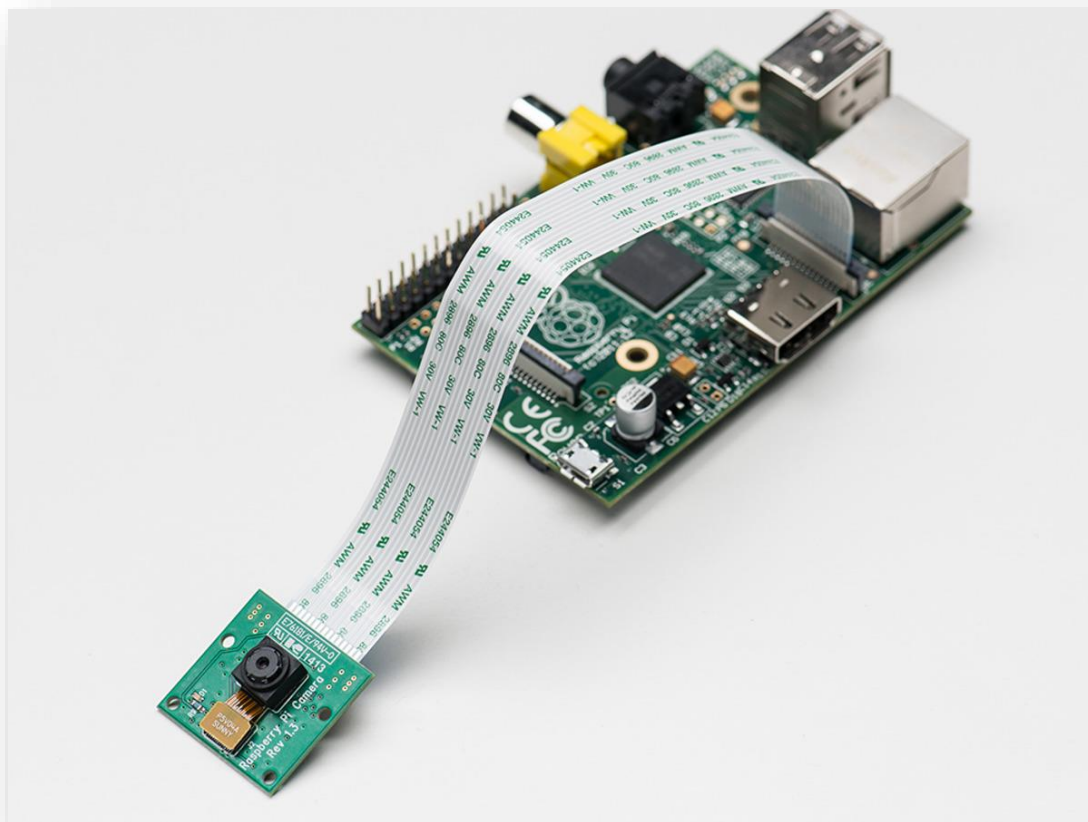
Σχήμα 4.1 - Raspberry Pi 2

4.1.3 Raspberry Pi Camera Board

Στα σημαντικότερα τμήματα του υλικού μας συγκαταλέγεται η κάμερα η οποία αποτελεί τα «μάτια» του αυτόνομου μηχανοκίνητου οχήματος μας. Όπως φαίνεται και στο σχήμα 4.2 είναι μία μικρών διαστάσεων κάμερα (25x24x9mm) βάρους μόλις 3g. Υποστηρίζει τη λήψη βίντεο αναλύσεων από 480p μέχρι και 1080p με ρυθμούς σάρωσης από 30 έως 60 καρέ το δευτερόλεπτο (fps). Διαθέτει αισθητήρα με ανάλυση στα 2592x1944 ιχνοστοιχεία – pixels. Η σύνδεση της γίνεται μέσω μίας απλής καλωδιωταινίας όπως φαίνεται και στο σχήμα 4.3.



Σχήμα 4.2 - Raspberry Pi Camera board



Σχήμα 4.3 - Σύνδεση Raspberry Pi2 και Camera board

4.1.4 Σύγκριση raspberry pi 2 με άλλα μοντέλα

	Model B+	Raspberry Pi 2
System-on-a-chip (SoC)	Broadcom BCM2835	Broadcom BCM2836
CPU:	700 MHz ARM11 ARM1176JZF-S core	900MHz quad-core ARM Cortex-A7
GPU:	Broadcom VideoCore IV,OpenGL ES 2.0,OpenVG 1080p30 H.264 high-profile encode/decode	
Memory (SDRAM)iB	512 MiB	1024 MiB

USB ports:	4 USB 2.0 (via integrated USB hub in LAN9514)	
Video outputs:	HDMI Composite video requires 4 Pole Adapter	
Audio outputs	TRS connector 3.5 mm jack, HDMI	
Onboard Storage:	Micro Secure Digital / MicroSD slot	
Onboard Network	None	10/100 wired Ethernet RJ45
Low-level peripherals	40 General Purpose Input/Output (GPIO) pins, Serial Peripheral Interface Bus (SPI), I ² C, I ² S, I2C IDC Pins, Universal asynchronous receiver/transmitter (UART)	
Power ratings:	~650 mA, (3.0 W)	
Power source	5 V (DC) via Micro USB type B or GPIO header	
Size:	85.0 x 56.0 mm x 17mm	
Weight	40g	

4.1.5 Raspberry Pi GPIO

Το πιο ισχυρό χαρακτηριστικό του Raspberry Pi2 είναι η διάταξη των ακροδεκτών εισόδου-εξόδου GPIO(general purpose input/output)(Σχήμα 4.4) η οποία αποτελεί ένα βασικό εξάρτημα για τη διασύνδεση του Raspberry Pi2 με εξωτερικές συσκευές όπως διακόπτες, μεταγωγούς, κινητήρες, αντιστάσεις είναι το Adafruit GPIO T-Cobbler (Σχήμα 4.5). Πρόκειται για μία πλακέτα η οποία μεταφέρει τις θύρες εισόδου εξόδου του Raspberry Pi2 σε δικούς τις ακροδέκτες-υποδοχείς διευκολύνοντας με αυτόν τον τρόπο τη διαδικασία σύνδεσης συσκευών. Στην περίπτωση της δικής μας εργασίας έγινε χρήση του για τη σύνδεση των ακροδεκτών του στο breadboard.



Σχήμα 4.4 – GPIO (General Purpose Input/Output)



4.2 DC DUAL MOTOR DRIVER 30V 4A V2

Το DC Dual Motor Driver 30V 4A V2 επιτρέπει να οδηγείτε ανεξάρτητα δύο κινητήρες συνεχούς ρεύματος, ελέγχει τόσο την ταχύτητα όσο και την κατεύθυνση. Βασίζεται στο διάσημο ολοκληρωμένο κύκλωμα L298, κατασκευασμένο από την STMicroelectronics, το L298 είναι ένα ολοκληρωμένο μονολιθικό κύκλωμα σε ένα πακέτο PowerSO20. Είναι ένα υψηλής τάσης, υψηλού ρεύματος οδηγός διπλού fullbridge σχεδιασμένος για να δέχεται τα επίπεδα λογικής TTL και να οδηγεί επαγωγικά φορτία όπως ρελέ, πηνία, DC και κλιμακωτοί κινητήρες.

Η ελάχιστη τάση τροφοδοσίας που επιτρέπεται είναι 7V, έτσι ώστε να μπορείτε επίσης να χρησιμοποιήσετε τις μπαταρίες LiPo δύο-κυττάρων (7.4V) που χορηγεί μικρές διαστάσεις και χαρακτηριστικά χαμηλού βάρους. Η μέγιστη τάση τροφοδοσίας που υποστηρίζεται από αυτό το σύστημα είναι 30V.

Για να ενεργοποιήσετε ή να απενεργοποιήσετε κάθε κανάλι ανεξάρτητα από τα σήματα εισόδου παρέχονται δύο εισοδοί ενεργοποίησης (E1, E2) τοποθετημένοι στο μπροστινό μέρος της πλακέτας, η ρύθμιση της ταχύτητας του κινητήρα επιτυγχάνεται εφαρμόζοντας σε αυτές τις καρφίτσες ένα σήμα PWM με 20KHz max. συχνότητα.

Αυτή η πλακέτα παρέχει επίσης δείκτες κατεύθυνσης LED και για τα δύο κανάλια. Αυτό είναι πολύ χρήσιμο κατά τη διάρκεια του σταδίου εγκατάστασης για να ελέγξει τη συμπεριφορά του firmware.

ΟΔΗΓΙΕΣ

Δύο ακροδέκτες είναι οι δύο εξοδοι για κινητήρες (M1 και M2). Οι 3 ακίδες δίπλα στον ακροδέκτη ισχύος, χρησιμοποιείται για να ελέγχει τα 2 κανάλια του DC Dual Motor Driver 30V 4A V2 (Σχήμα 4.6). Κάθε ένα από αυτά έχει σήματα όπως αναφέρεται στο πίνακα 1.

Πίνακας 1:

Channel 1		Channel 2	
<i>Name</i>	<i>Function</i>	<i>Name</i>	<i>Function</i>
1A	Input A of ch.1 (TTL input)	2A	Input A of ch.2 (TTL input)
1B	Input B of ch.1 (TTL input)	2B	Input B of ch.2 (TTL input)
E1	Enable ch.1 (TTL input)	E2	Enable ch.2 (TTL input)

Για να καταλάβουμε το νόημα αυτών των σημάτων και τη χρήση τους μπορείτε να διαβάσετε τον παρακάτω πίνακα (Πιν.2), όπου αναφέρονται όλες οι προϋποθέσεις. Σημειώστε ότι έχουν αναφερθεί μόνο τις προϋποθέσεις για το κανάλι 1 επειδή οι συνθήκες για το κανάλι 2 είναι ακριβώς όμοια τους

Πίνακας 2:

Inputs			M1+ and M1- output
E1	1A	1B	
1	1	1	HIGH state for both output (motor stopped)
1	0	0	LOW state for both output (motor stopped)
1	1	0	Current flows from M1+ to M1- (direction 1)
1	0	1	Current flows from M1- to M1+ (direction 2)
0	X	X	High impedance (motor is in free running)

Σχετικά με τα σήματα E1 και E2, έχουν αντιστάσεις pull-up έτσι ώστε σε ορισμένες εφαρμογές δεν χρειάζεται να τους οδηγούν και χρειαζόμαστε μόνο 2 σήματα για τον έλεγχο των οχημάτων.

Χαρακτηριστικά:

Supply voltage	7 - 30V
Supply current (logic)	24mA typ. (38mA max.)
Output current	4A (2A for each channel)
Data I/O voltage	TTL standard
Dimensions	37x36x13mm (connectors included)
Weight	12.5g / 0.44oz
Operating temperature	-25 - 130°C



Σχήμα 4.6 – DC Dual Motor Driver 30V 4A V2

4.3 DC MOTORS

Πρόκειται για γωνιακούς κινητήρες (Σχήμα 4.7)μοτέρ DAGU. Αυτοί οι ηλεκτρομειωτήρες είναι οι ίδιοι που χρησιμοποιούνται στο Chassis Magician και το RedBot Kit και είναι κατασκευασμένοι στην Κίνα . Λόγω του ότι οι κινητήρες αυτοί είναι σε ορθή γωνία, θα μπορούμε επίσης να παρακολουθείτε και να ελέγχουμε την περιστροφή των τροχών μας.

Κάθε ηλεκτρομειωτήρας απαιτεί τάση 3-6V με ρεύμα χωρίς φορτίο 200mA ενώ διαθέτει ρεύμα στάσης 3A 6V (1.5A 3V), λόγο κιβωτίου ταχυτήτων 48: 1 και ταχύτητα τροχού 65 rpm σε 3V χωρίς φορτίο .

Χαρακτηριστικά:

Τάση: 3V - 6V

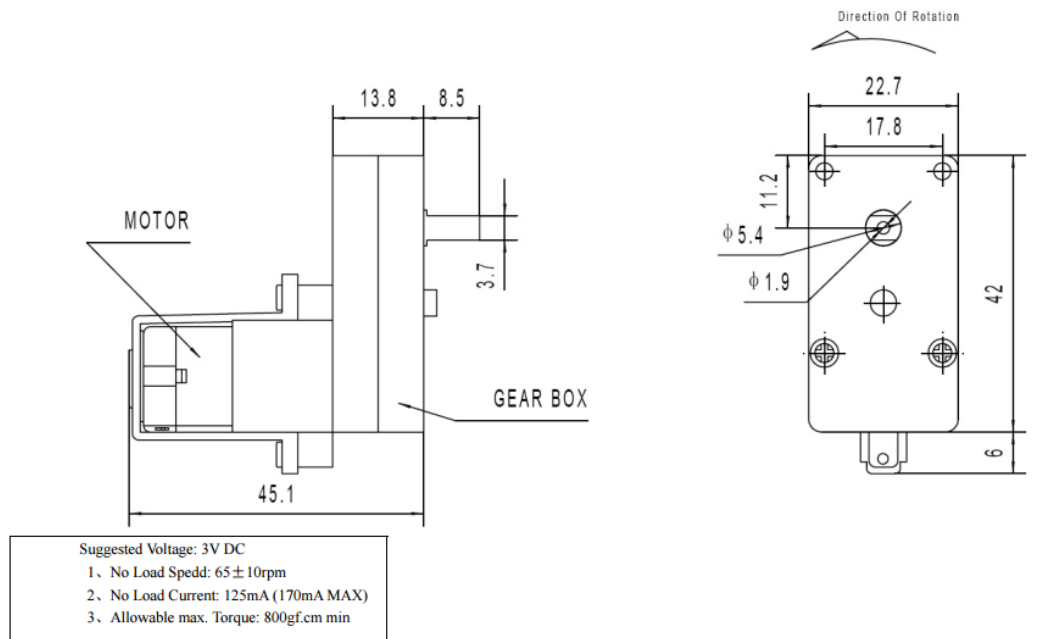
Δεν υπάρχει φορτίο ρεύματος: 200mA

Ρεύμα στάσης: 3A 6V (1,5A 3V)

Αναλογία κιβωτίου ταχυτήτων: 48: 1

Ταχύτητα τροχού: 65rpm στα 3V χωρίς φόρτωση

DG02S-A130GEARMOTOR



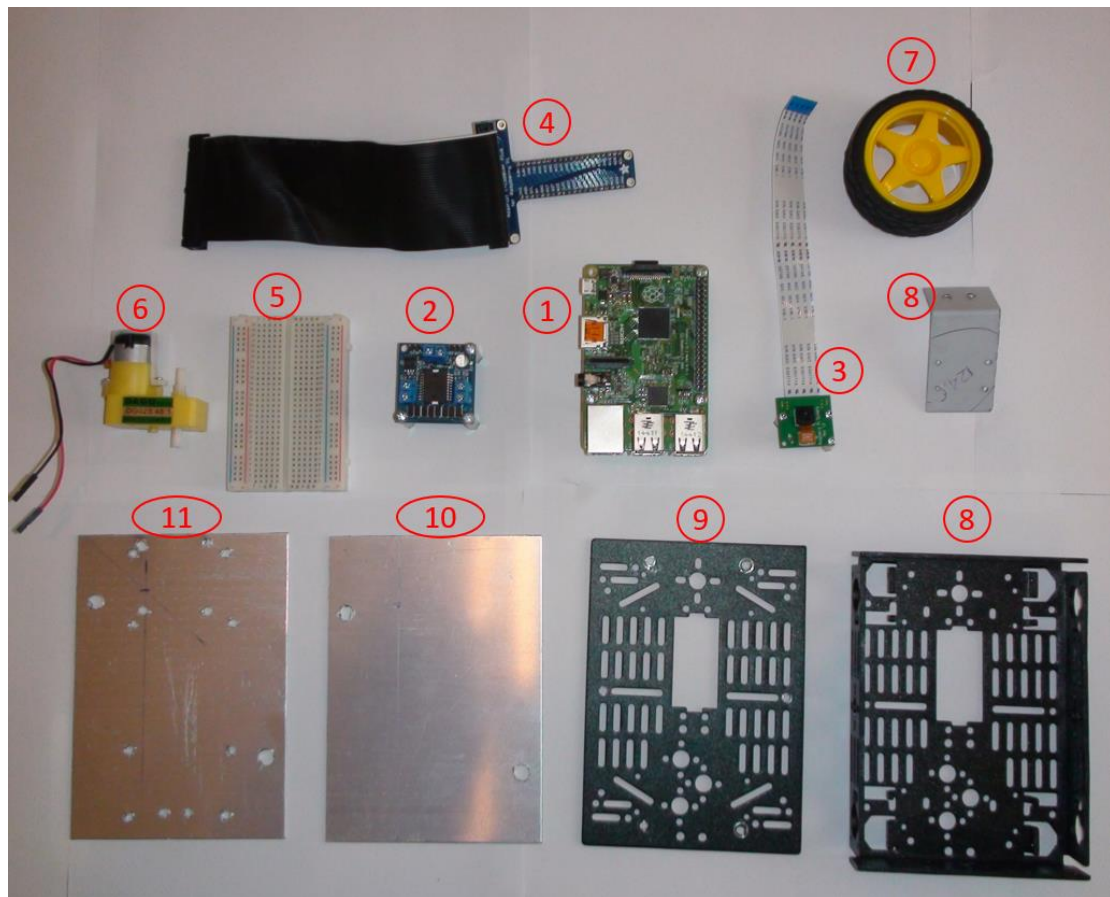
Σχήμα 4.7 – DC Motors

Κεφάλαιο 5^ο-ΥΛΟΠΟΙΗΣΗ ΑΥΤΟΝΟΜΟΥ ΟΧΗΜΑΤΟΣ

5.1 ΠΕΡΙΓΡΑΦΗ ΥΛΙΚΟΥ

Στη πορεία εξέλιξης της παρούσας πτυχιακής εργασίας διακρίνουμε τρεις φάσεις. Στην πρώτη φάση η οποία προηγείται και χρονολογικά ασχοληθήκαμε με τη θεωρητική μελέτη των μικροϋπολογιστών και συγκεκριμένα του Raspberry pi2, τις δυνατότητες συνδεσιμότητας του με διατάξεις όπως LEDs, κινητήρες, κάμερα. Η δεύτερη φάση που αποτελεί και μελέτη του τρέχοντος κεφαλαίου είναι το κατασκευαστικό τμήμα της εργασίας με σκοπό την υλοποίηση του αυτόνομου οχήματος, ενώ τέλος στην τρίτη φάση αναφέρεται στην κατασκευή του αλγορίθμου και πώς αυτό προσαρμόζεται στο μηχανικό μας μοντέλο ώστε να έχουμε το επιθυμητό αποτέλεσμα της οδήγησης του.

Τα επιμέρους εξαρτήματα από τα οποία αποτελείται το αυτοκινούμενου οχήματος μας είναι τα εξής (Σχήμα 5.1)



Σχήμα 5.1 - Κύρια τμήματα αυτόνομου οχήματος.

1. Μικροϋπολογιστής Raspberry Pi2
2. Driver συνεχούς ρεύματος για τη λειτουργία των κινητήρων
3. Raspberry Pi Camera Board
4. Raspberry Pi GPIO, διεπαφή σύνδεσης περιφερειακών συσκευών με τον μικροϋπολογιστή
5. Breadboard
6. Κινητήρας συνεχούς ρεύματος
7. Ρόδα οχήματος
8. Σασί μηχανοκίνητου οχήματος
9. Καπάκι του σασί
10. Πλαίσιο βάσης στήριξης του breadboard
11. Πλαίσιο βάσης στήριξης του Raspberry Pi2

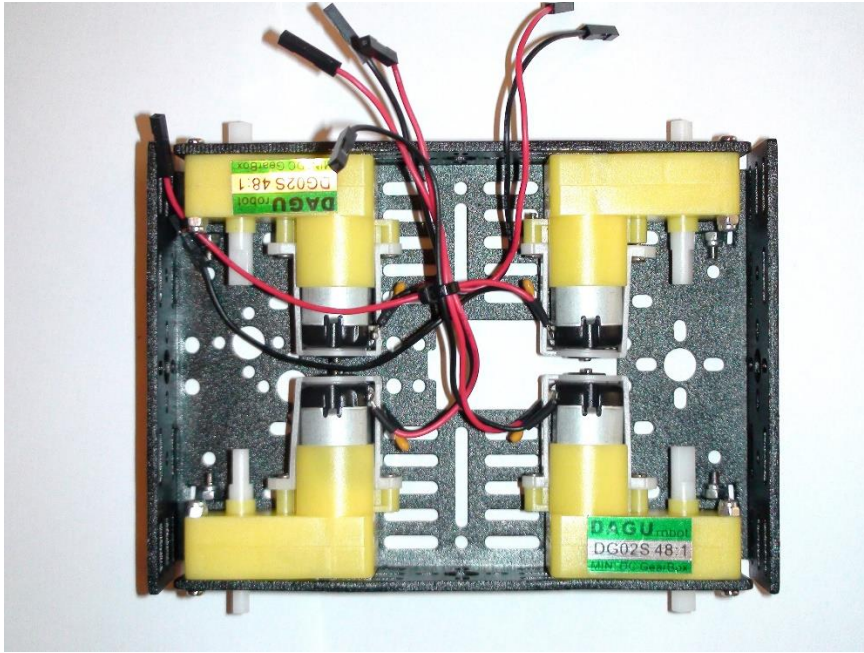
Εκτός από τα παραπάνω κύρια τμήματα του οχήματος υπάρχουν και δευτερεύοντα εξαρτήματα για τη συναρμολόγηση και διασύνδεση του. Κατάλληλες βίδες προσαρμόζουν και συνδέουν τα αντίστοιχα πλαίσια μαζί με ντίζες (βλέπε Παράρτημα Σχήμα 1), διάφορα μεγέθη μεταλλικών παξιμαδιών (βλέπε Παράρτημα Σχήμα 2) καθώς και πλαστικοί αποστάτες για τη στήριξη και μόνωση των ηλεκτρονικών εξαρτημάτων (βλέπε Παράρτημα Σχήμα 3).

Η αυτονομία του οχήματος παρέχεται από δύο πηγές ηλεκτρικής ενέργειας. Ως πρώτη πηγή ηλεκτρικής ενέργειας χρησιμοποιούμε ένα Power Bank (βλέπε Παράρτημα Σχήμα 4) με χαρακτηριστικά τάσης 5V και χωρητικότητα 10400mAh, το οποίο τροφοδοτεί το Raspberry Pi2, ενώ η δεύτερη πηγή είναι 6 μπαταρίες τύπου πρωτοβάθμιου αλκαλικού μαγγανίου τον 1.5 Volts που συνδέονται σε σειρά(βλέπε Παράρτημα Σχήμα 4).

Για τον τερματισμό του προγράμματος μας χρησιμοποιήσαμε ένα button και δύο αντιστάσεις των 10KΩ και του 1KΩ αντίστοιχα με τη συνδεσμολογία του φαίνεται στο σχήμα 5(βλέπε Παράρτημα)

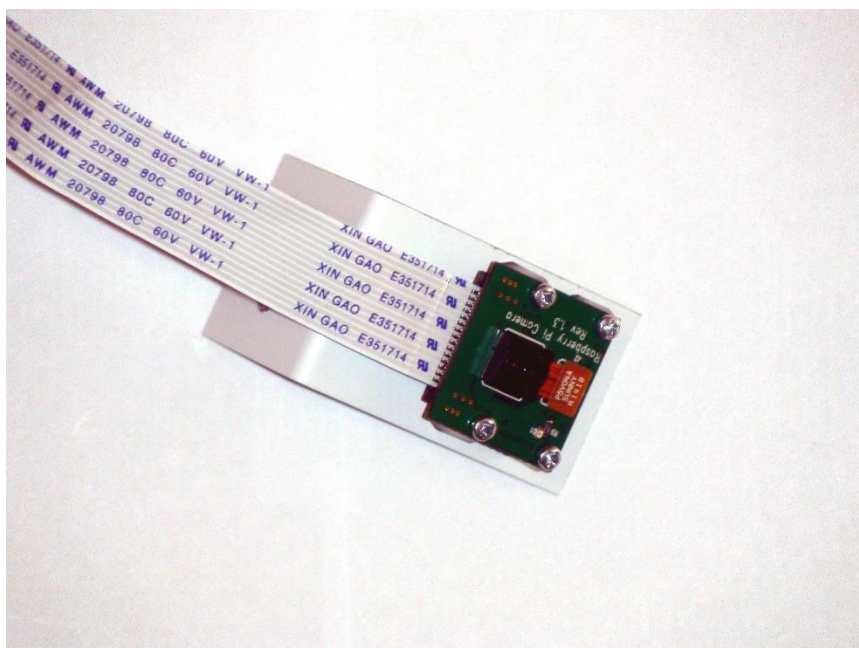
5.2 ΣΥΝΑΡΜΟΛΟΓΗΣΗ ΑΥΤΟΝΟΜΟΥ ΟΧΗΜΑΤΟΣ

Προσαρμόζουμε με την χρήση βιδών τους κινητήρες συνεχούς ρεύματος μέσα στο σασί του μηχανοκίνητου οχήματος (Σχήμα 5.2), στη συνέχεια τοποθετούμε το καπάκι του σασί πάνω στο σασί του μηχανοκίνητου οχήματος και το ενώνουμε με 2 βίδες όπως παρουσιάζεται στο σχήμα 6 (βλέπε Παράρτημα)

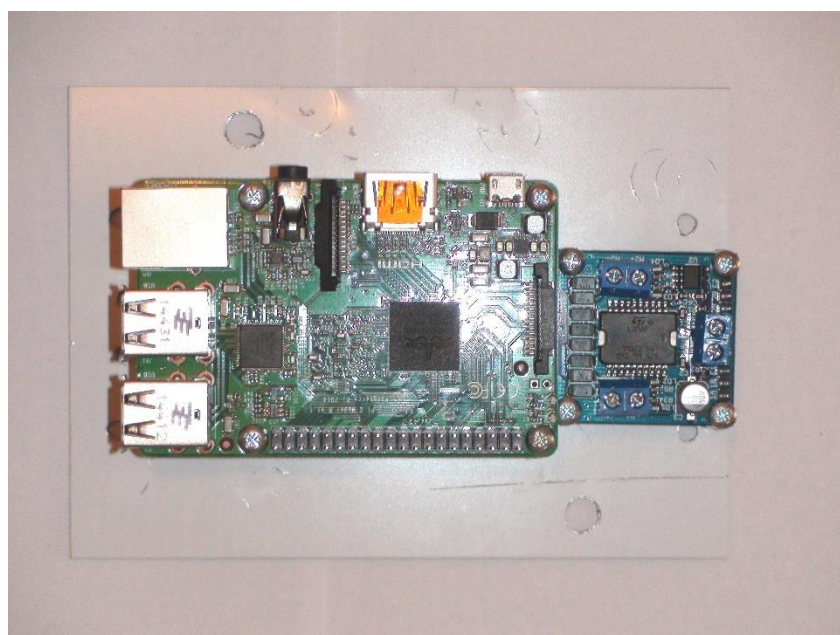


Σχήμα 5.2 – Προσαρμογή κινητήρων

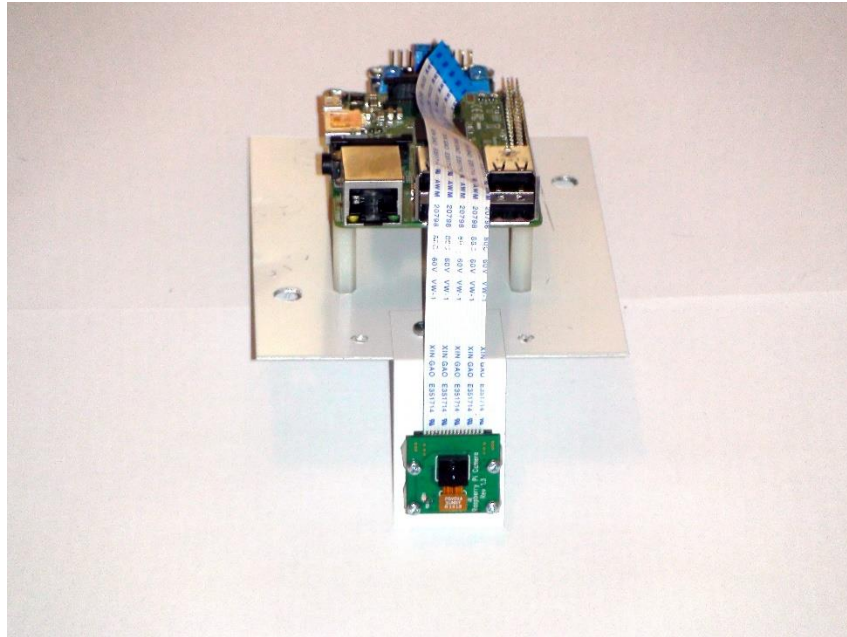
Με χρήση των πλαστικών αποστατών τοποθετούμε τα βασικά μέρη πάνω στο όχημα μας. Με τους μεγάλους αποστάτες σταθεροποιούμε το Raspberry Pi 2, με τους μεσαίους το κύκλωμα οδήγησης των κινητήρων συνεχούς ρεύματος και με τους μικρούς την κάμερα (βλέπε Παράρτημα Σχήμα 7). Τοποθετούμε την κάμερα πάνω στην ειδική βάση της και στην συνέχεια την ενώνουμε με το πλαίσιο βάσης στήριξης του Raspberry Pi 2, με την βοήθεια 2 βιδών. Έπειτα τοποθετούμε πάνω στην βάση στήριξης του Raspberry το Raspberry Pi 2 και τον οδηγό κινητήρων συνεχούς ρεύματος. Συνδέουμε τη Raspberry Pi Camera Board στη αντίστοιχη υποδοχή του μικροϋπολογιστή μας. Προσαρμόζουμε το GPIO T-Cobbler (βλέπε Παράρτημα Σχήμα 8) στις GPIO του Raspberry Pi 2 και το σταθεροποιούμε με χρήση του breadboard. Συνδέουμε τις εξόδους του οδηγού κινητήρων συνεχούς ρεύματος με τους κινητήρες συνεχούς ρεύματος και τοποθετούμε καλώδια στις εισόδους και στην τροφοδοσία του οδηγού. (Σχήμα 5.3 – 5.6)(βλέπε Παράρτημα Σχήμα 9 - 10).



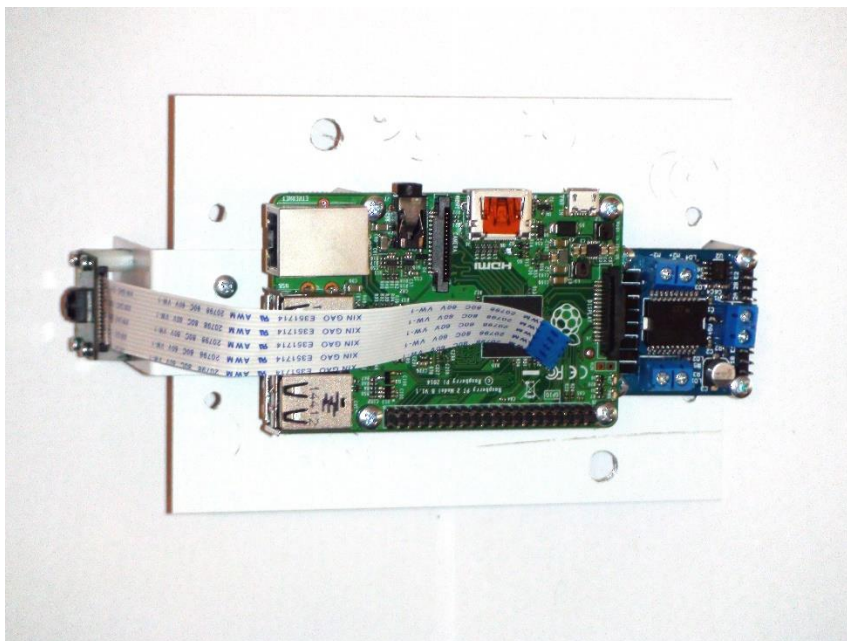
Σχήμα 5.3 – Η Raspberry Pi Camera Board με την βάση στήριξης.



Σχήμα 5.4 – Το Raspberry Pi 2 και το κύκλωμα του οδηγού
κινητήρων με το πλαίσιο βάσης στήριξης.

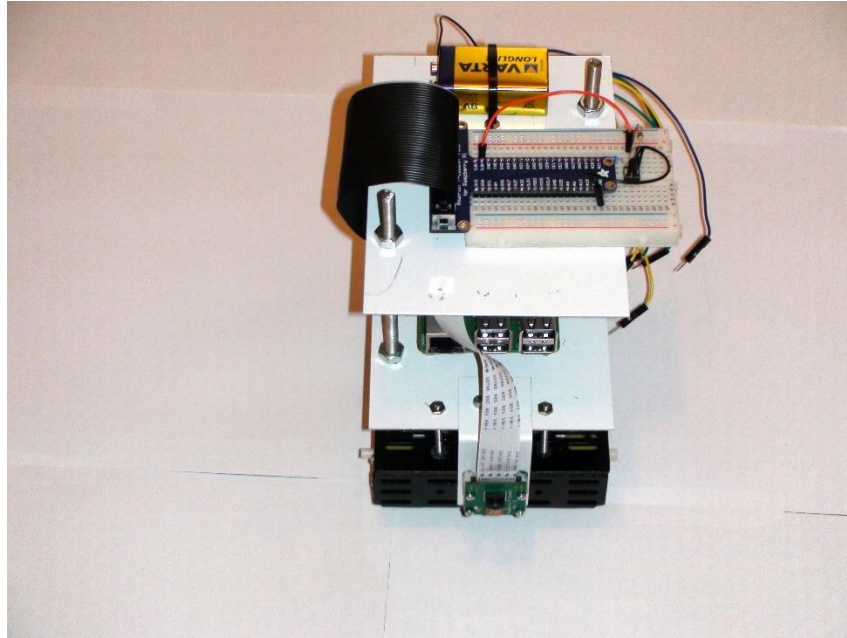


*Σχήμα 5.5 – Ένωση Raspberry Pi Camera με το πλαίσιο
βάσης στήριξης του Raspberry από μπροστά.*



*Σχήμα 5.6 – Ένωση Raspberry Pi Camera με το πλαίσιο
βάσης στήριξης του Raspberry από πάνω.*

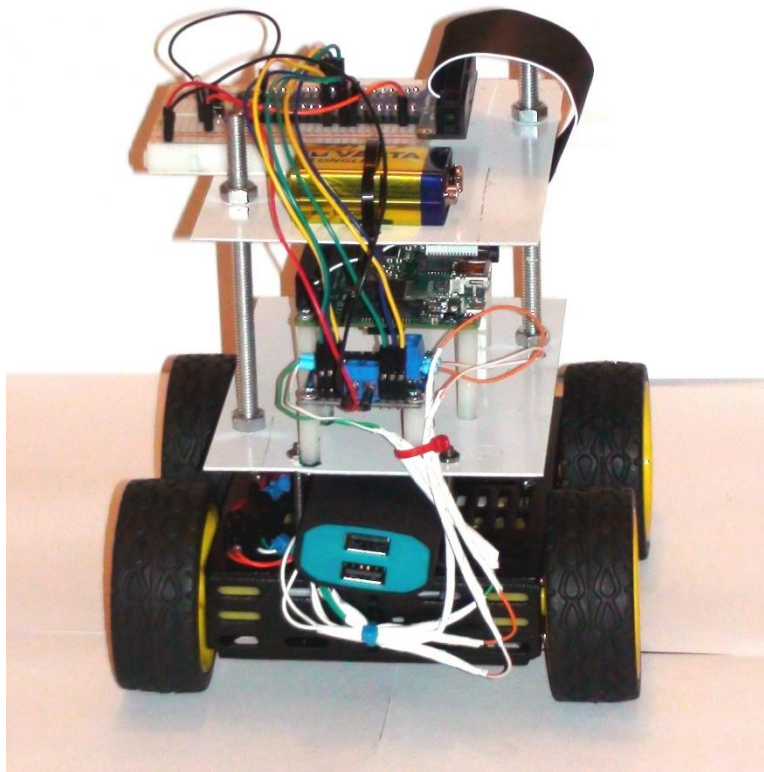
Σταθεροποιούμε τις ντίζες με τα μεγάλα παξιμάδια ρυθμίζοντας το κατάλληλο ύψος. Τοποθετούμε το πλαίσιο βάσης στήριξης του breadboard και το σταθεροποιούμε με τα μεγάλα παξιμάδια. Τοποθετούμε το breadboard και την μπαταρία των 9 Volts πάνω στην βάση(Σχήμα 5.7).



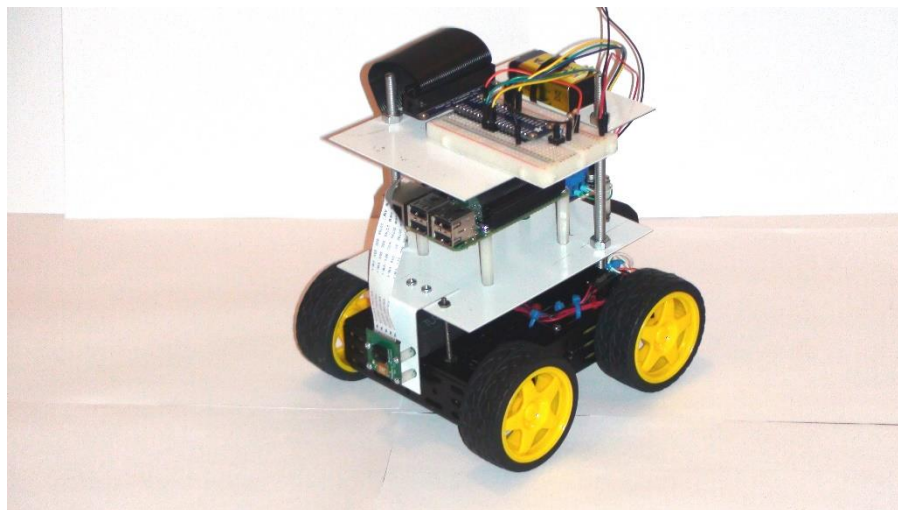
Σχήμα 5.7 – Ενσωμάτωση του Breadboard και της μπαταρίας 9V με το όχημα

Συνδέουμε τις εισόδους του οδηγού κινητήρων συνεχούς ρεύματος με τις διεπαφές GPIO του Raspberry Pi 2 πάνω στο breadboard. Συνδέουμε την τροφοδοσία του οδηγού κινητήρων συνεχούς ρεύματος με την μπαταρία 9Volts μέσω του breadboard. Τοποθετούμε τον διακόπτη με τις δύο αντιστάσεις πάνω στο breadboard και κατασκευάζουμε ένα κύκλωμα Pull Up Resistor για την διακοπή του προγράμματος(βλέπε Παράρτημα Σχήμα 11).

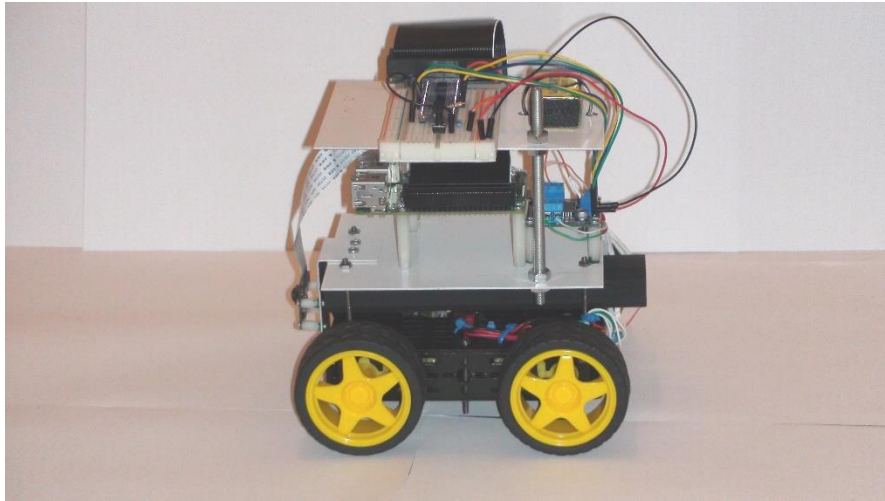
Τοποθετούμε το power bank στο καινό ανάμεσα στο καπάκι του σασί και στο πλαίσιο βάσης στήριξης του Raspberry(Σχήμα 5.9-5.11).



Σχήμα 5.9 – Τοποθέτηση Power Bank



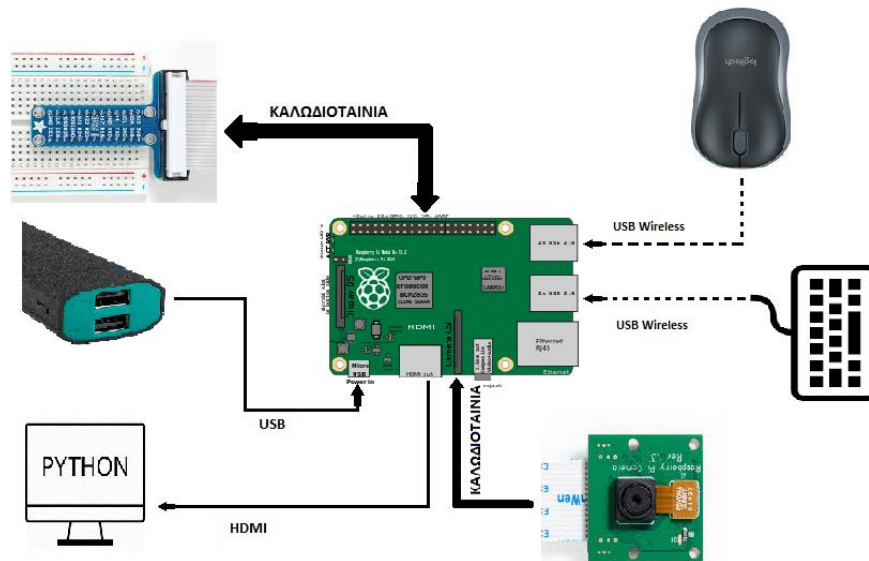
Σχήμα 5.10 – Αυτόνομο Μηχανοκίνητο Όχημα



Σχήμα 5.11 – Αυτόνομο Μηχανοκίνητο Όχημα πλάγια όψη

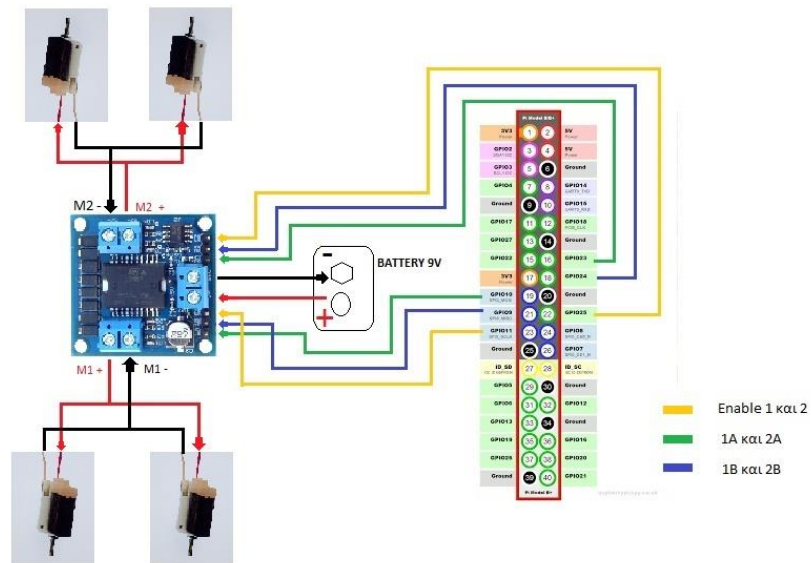
5.3 ΔΙΑΣΥΝΔΕΣΗ

Για την τροφοδοσία του Raspberry Pi 2 χρησιμοποιήσαμε ένα Power Bank TurboX και το συνδέουμε στην θύρα micro usb. Η Raspberry Pi Camera Board συνδέετε στην ειδική θύρα που έχει το Raspberry Pi 2 όπως βλέπετε στο σχήμα 5.22. Οι GPIO μεταφέρονται πάνω στο breadboard με την βοήθεια του GPIO T-cobbler (σχήμα 5.12).



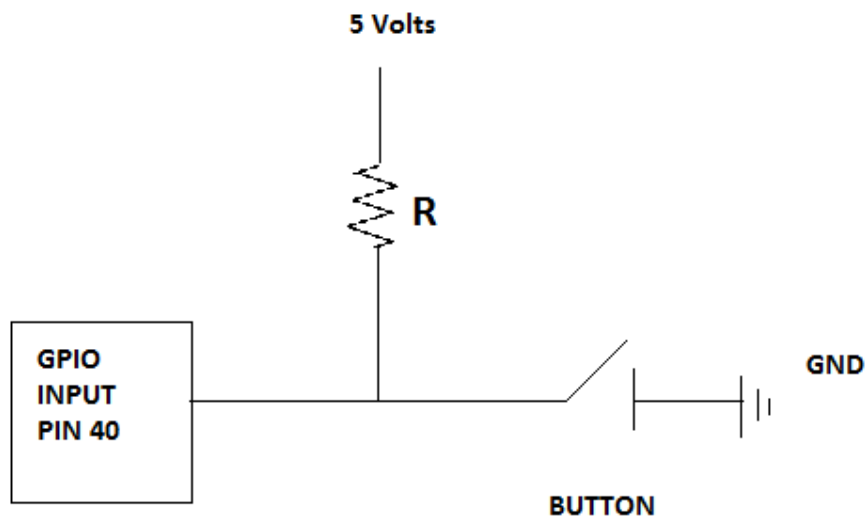
Σχήμα 5.12 – Συνδεσμολογία Raspberry Pi 2

Το κύκλωμα οδήγησης των κινητήρων αποτελείται από 2 εξόδους και 3 επαφές εισόδου για κάθε έξοδο και την τροφοδοσία του. Το κύκλωμα τροφοδοτείται από μπαταρία 9V(battery pack). Στην επαφή V+ συνδέεται ο θετικός πόλος της μπαταρίας και στην επαφή GND συνδέεται ο αρνητικός πόλος. Οι διεπαφές εισόδου, οι A1 B1 E1 είναι εισοδοί της M1 εξόδου και οι A2 B2 E2 είναι εισοδοί για την M2 έξοδο, συνδέονται με τις GPIO που βρίσκονται στο breadboard. Οι εξοδοί M1 και M2 τροφοδοτούν τους κινητήρες με ρεύμα. Η M1 τροφοδοτεί τους δεξιούς κινητήρες και η M2 τροφοδοτεί αριστερούς κινητήρες (σχήμα 5.13).



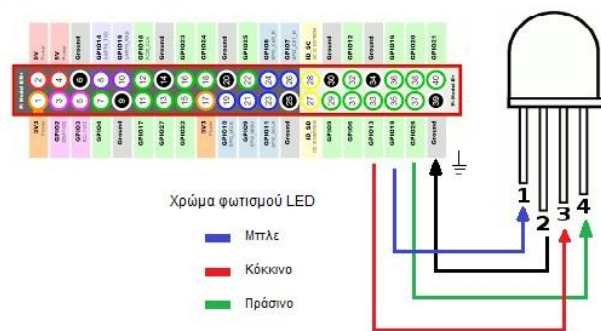
Σχήμα 5.13 – Συνδεσμολογία κυκλώματος οδήγησης κινητήρων.

Το κύκλωμα Pull Up Resistor αποτελείται από 2 αντιστάσεις και 1 Button (σχήμα 5.14). Με την ενεργοποίηση του button η διεπαφή 40 των GPIO αλλάζει κατάσταση.



Σχήμα 5.14 – Pull Up Resistor

Το RGB LED είναι τοποθετημένο πάνω στο breadboard. Χρησιμοποιεί τις διεπαφές 33,35,37 και αναλόγως πιες είναι ενεργοποιημένες θα ανάψει το αντίστοιχο χρώμα. Η κάθοδος του είναι συνδεδεμένη με την διεπαφή 39 των GPIO και χρησιμοποιείται σαν γείωση. Τα χρώματα των γραμμών στο σχήμα 5.15 δείχνουν τι χρώμα θα έχει το led αν ενεργοποιηθεί η αντίστοιχη διεπαφή.



Σχήμα 5.15 – Συνδεσμολογία RGB LED

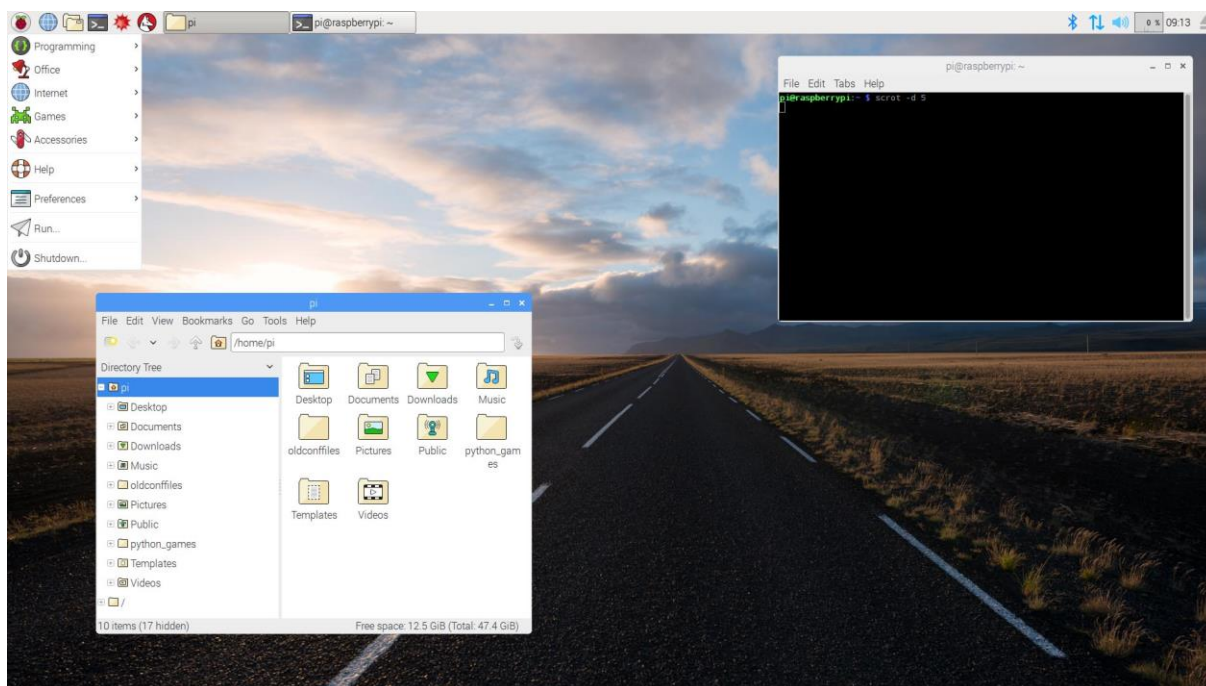
Κεφάλαιο 6 – Λογισμικό

Το Raspberry Pi2 έχει σχεδόν όλα τα χαρακτηριστικά ενός σύγχρονου προσωπικού υπολογιστή και το βασικότερο από αυτά είναι το λειτουργικό του σύστημα. Στο παρόν κεφάλαιο θα περιγράψουμε τη διαδικασία εγκατάστασης του λειτουργικού του συστήματος καθώς αυτό είναι η βάση για την εκτέλεση οποιουδήποτε προγράμματος και οποιασδήποτε λειτουργίας. Η εγκατάσταση γίνεται στην κάρτα μνήμης τύπου SD (16GB) που αποτελεί και το μοναδικό αποθηκευτικό χώρο που διαθέτει το μικροϋπολογιστικό μας σύστημα.

6.1 Λειτουργικό σύστημα Rasbian Jessie

Το λειτουργικό σύστημα που επιλέξαμε για εγκατάσταση στο Raspberry Pi2 είναι το Rasbian jessie. Το Rasbian [55] είναι ένα δωρεάν λειτουργικό σύστημα που βασίζεται στην διανομή Debian του linux[56] και είναι προσαρμοσμένο στο υλικό του μικροϋπολογιστή Raspberry. Το λειτουργικό αυτό σύστημα αποτελείται από μια μεγάλη γκάμα βασικών προγραμμάτων και εφαρμογών που έχουν σαν κύριο σκοπό τη λειτουργικότητα του Raspberry.

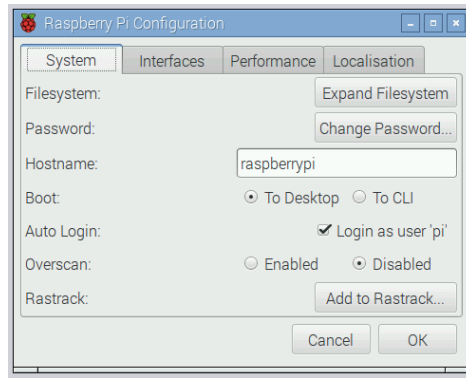
Η βασική έκδοση που ολοκληρώθηκε το 2015, περιέχει πάνω από σαράντα χιλιάδες πακέτα που αποσκοπούν στην βέλτιστη απόδοση του μικροϋπολογιστή. Φυσικά το λειτουργικό Rasbian ακόμα και τη στιγμή συγγραφής της παρούσας πτυχιακής εργασίας είναι σε διαρκή βελτίωση με σκοπό τη σταθερότητα και την απόδοση όσο πιο δυνατόν περισσότερων πακέτων Debian. Στο σχήμα 6.1 απεικονίζεται ένα στιγμιότυπο της βασικής επιφάνειας εργασίας.



Σχήμα 6.1- Στιγμιότυπο επιφάνειας εργασίας λειτουργικού συστήματος Raspbian

Πολλές από τις αλλαγές που πραγματοποιήθηκαν στο Jessie από τον προκάτοχό του Wheezy είναι μη ορατές στον τελικό χρήστη. Πραγματοποιήθηκαν τροποποιήσεις στο παρόν σύστημα, με σκοπό τη βελτίωση των επιδόσεων και της ευελιξίας του, υπάρχουν πολλές διορθώσεις στην λειτουργία του αλλά και διορθώσεις σφαλμάτων. Ταυτόχρονα με την αναβάθμιση του λογισμικού σε Jessie, πραγματοποιήθηκαν αλλαγές και βελτιώσεις στο περιβάλλον χρήστη επιφάνειας εργασίας. Μια σημαντική αλλαγή στην επιφάνεια εργασίας, η οποία δεν μπορεί να περάσει απαρατήρητη, είναι η προσθήκη εικονιδίου εξαγωγής στη γραμμή μενού. Το εικονίδιο μας επιτρέπει την ασφαλή εξαγωγή των δίσκων USB.

Ακόμα πρέπει να επισημάνουμε τον καινούριο διάλογο ρυθμίσεων, ο οποίος βρίσκεται στο Menu->Properties. Το Raspberry Pi Configuration (σχήμα) είναι η νέα έκδοση της παλιάς γραμμής εντολών raspi-config. Περιέχει τα ίδιες λειτουργίες σε μια ωραιότερη και πιο εύκολα προσβάσιμη διεπαφή.



Σχήμα 6.2 - Raspberry Pi Configuration

6.2 Γλώσσα Python

6.2.1 Τι είναι η Python

Η Python είναι μια διερμηνευόμενη (interpreted), αλληλεπιδραστική (interactive) και προσανατολισμένη σε αντικείμενα (object-oriented) γλώσσα προγραμματισμού, που αναπτύχθηκε από τον Guido van Rossum[57]. Το όνομά της προέρχεται από ένα από τα αγαπημένα τηλεοπτικά θεάματα του van Rossum, το ιπτάμενο τσίρκο των Monty Python's. Η Python είναι μια γλώσσα που συνδυάζει σημαντική ισχύ με πολύ σαφή σύνταξη. Χρησιμοποιεί ενότητες (modules), τάξεις (classes), εξαιρέσεις (exceptions) καθώς και πολύ υψηλού επιπέδου δυναμικούς τύπους δεδομένων.

Οι διερμηνευτές (interpreters) της Python είναι διαθέσιμοι για τα περισσότερα λειτουργικά συστήματα και ο πηγαίος κώδικας (source code) της Python διατίθεται δωρεάν. Η επίσημη ιστοσελίδα της γλώσσας είναι η <http://www.python.org>. Η ανάπτυξη της Python ξεκίνησε το 1990 στο CWI του Άμστερνταμ και συνεχίζεται στο CNRI του Reston. Η

γλώσσα διαθέτει μια κομψή αν και όχι πολύ απλοποιημένη σύνταξη και έναν μικρό αριθμό από ισχυρούς και υψηλού επιπέδου τύπους δεδομένων.

Η Python μπορεί να επεκταθεί προσθέτοντάς της καινούργιες ενότητες (modules) που να είναι γραμμένες σε μια γλώσσα που μεταγλωττίζεται, όπως είναι η C ή η C++. Αυτές οι ενότητες επέκτασης μπορούν να ορίσουν νέες συναρτήσεις (functions) και μεταβλητές (variables) καθώς επίσης και καινούργιους τύπους δεδομένων (object types). Η Python συγκρίνεται συχνά μ' άλλες διερμηνευόμενες γλώσσες προγραμματισμού, όπως είναι οι Java, JavaScript, Perl, Tcl και η Smalltalk, αλλά συγκρίσεις μπορούν να γίνουν και με τις C++, Common Lisp και Scheme.

6.2.2 Ο Διερμηνευτής της Python

Ο διερμηνευτής (interpreter) της Python είναι συνήθως εγκατεστημένος στον κατάλογο /usr/local/bin/, οπότε γράφοντας *python* από τη γραμμή εντολών, μπορούμε να την εκκινήσουμε. Ένας άλλος πιθανός κατάλογος είναι ο /usr/local/. Για να βγούμε από τον interpreter, μπορούμε να δώσουμε τον χαρακτήρα τέλους αρχείου (end-of-file character), που είναι ο *Control-D* στο Unix και ο *Control-Z* στο DOS και στα Windows. Αν αυτό δεν δουλέψει, μπορούμε να γράψουμε τις εξής εντολές :

```
"import sys; sys.exit()"
```

Ο πιο εύκολος τρόπος για να διαπιστώσουμε αν υποστηρίζεται η επεξεργασία εντολών ανά γραμμή (command line editing) είναι να πατήσουμε τα πλήκτρα *Control-P* μόλις εμφανισθεί η προτροπή (prompt) της Python. Αν ακουστεί ένας ήχος (beep), αυτό σημαίνει ότι είναι διαθέσιμο το command line editing, ενώ αν δεν συμβεί κάτι ή εμφανισθεί ο χαρακτήρας P, αυτό σημαίνει ότι δεν είναι διαθέσιμο το command line editing.

Ο διερμηνευτής μπορεί να διαβάζει και να εκτελεί εντολές αλληλεπιδραστικά (interactively), ενώ όταν καλείται με όρισμα ένα όνομα αρχείου, όπως για παράδειγμα C:\Python\Python p01, τότε διαβάζει και εκτελεί τις εντολές (script) που περιέχονται στο

αρχείο. Ένας τρίτος τρόπος για να ξεκινήσουμε τον interpreter είναι ο εξής : "python -c command [arg] ...", όπου εκτελούνται οι εντολές που περιέχονται στο command.

Η μεταβίβαση ορισμάτων στο διερμηνευτή γίνεται μέσω της μεταβλητής sys.argv η οποία είναι μία λίστα από αλφαριθμητικά (strings). Το μήκος είναι τουλάχιστον ίσο με ένα, στην περίπτωση που δεν υπάρχει κάποιο script και κάποια ορίσματα, οπότε το sys.argv[0] είναι ένα κενό αλφαριθμητικό.

6.2.3 Interactive Mode

Όταν οι εντολές δίνονται από το πληκτρολόγιο μία-μία, λέμε ότι ο interpreter βρίσκεται σε interactive mode και στην περίπτωση αυτή περιμένει την επόμενη εντολή με την προτροπή (prompt) >>> . Αν έχουμε ξεκινήσει μια εντολή και δεν την έχουμε τελειώσει, εμφανίζεται η προτροπή ... (τρεις τελείες). Ο interpreter εκτυπώνει ένα μήνυμα καλωσορίσματος που εμφανίζει τον αριθμό της έκδοσης της Python και κάποιες άλλες πληροφορίες πριν εμφανίσει το πρώτο prompt :

6.2.4 Βασικές Εισαγωγικές έννοιες στην Python

Στην Python υπάρχουν οι κυριολεκτικές σταθερές (Literal Constants). Ένα παράδειγμα κυριολεκτικής σταθεράς είναι ένας αριθμός όπως 5, 3.15, 5,15e-5 ή μία συμβολοσειρά. Αποκαλείται κυριολεκτική διότι κυριολεκτεί –η τιμή της ορίζεται κυριολεκτικά. Ο αριθμός 2 αναπαριστά τον εαυτό του και τίποτα άλλο – είναι σταθερά διότι η τιμή της δεν μπορεί να αλλάξει. Άρα, όλα αυτά αναφέρονται ως κυριολεκτικές σταθερές.

Οι αριθμοί στην Python είναι τριών τύπων: οι ακέραιοι(integers), οι αριθμοί κινητής υποδιαστολής (floating point) και οι μιγαδικοί αριθμοί(complex number)

Στην python όπως συμβαίνει με πολλές γλώσσες προγραμματισμού οι συμβολοσειρές (Strings) είναι μια ακολουθία από χαρακτήρες, η οποία μπορεί να αποτελείται από ένα πλήθος λέξεων. Οι λέξεις μπορεί να είναι στην αγγλική ή σε κάθε γλώσσα που υποστηρίζεται από το πρότυπο Unicode, ουσιαστικά σε σχεδόν κάθε γλώσσα του κόσμου. Από προεπιλογή, όλες οι συμβολοσειρές είναι σε Unicode. Οι συμβολοσειρές ορίζονται με χρήση είτε μονών είτε διπλών αλλά ακόμη και τριπλών εισαγωγικών. Η python υποστηρίζει τους γνωστούς χαρακτήρες διαφυγής [58] των περισσοτέρων γλωσσών προγραμματισμού όπως \n για την αλλαγή γραμμής, \t για τον χαρακτήρα Tab, \' για τη χρήση του χαρακτήρα εισαγωγικού κ.α. Οι συμβολοσειρές στην python είναι αμετάβλητες (immutable) αυτό σημαίνει ότι μετά τη δημιουργία τους, δεν μπορούμε να την αλλάξουμε..

Μέθοδος format

Μια συμβολοσειρά μπορεί να χρησιμοποιεί ορισμένες προδιαγραφές(specifications) και κατά συνέπεια, μπορεί να κληθεί η μέθοδος format για να τις αντικαταστήσει με αντίστοιχα ορίσματα (arguments) της μεθόδου format, όπως φαίνεται στο παράδειγμα που ακολουθεί.

```
age = 30

name = 'George'

print('The {0} is {1} years old'.format(name, age))

print('The {0} is programming with python;'.format(name))
```

Παρατηρούμε την πρώτη δήλωση {0} που αντιστοιχεί στη μεταβλητή name η οποία είναι το πρώτο όρισμα στη μέθοδο μορφοποίησης. Παρομοίως, η δεύτερη δήλωση {1} αντιστοιχεί στη μεταβλητή age ως το δεύτερο όρισμα στη μέθοδο μορφοποίησης.

χρησιμοποιήσαμε την εντολή format κατά την διάρκεια του πειραματισμού μου όταν θέλαμε να αποθηκεύω πολλές στιγμιότυπα, με σκοπό την σύγκριση μεταξύ τους.

```
i=0
```

```
while i<5:
```

```
    camera.resolution = (640, 320)
```

```
    camera.start_preview()
```

```
    # Camera warm-up time
```

```
    time.sleep(1)
```

```
    camera.stop_preview()
```

```
    camera.capture('fo{0}.jpg'. format(i))
```

```
    img = cv2.imread('fo{0}.jpg'. format(i),1)
```

```
    print(img.item(319,639,2))
```

```
    table=[0 for x in range(0,640)]
```

```
    for x in xrange(0,640):
```

Ο κώδικας που βλέπουμε παραπάνω έτρεχε 5 φορές. Σκοπός του κώδικα ήταν να απαθανατίσει πέντε στιγμιότυπα και να τις αποθηκεύσει με διαφορετικό όνομα.

Μεταβλητές (Variables)

Η χρήση μόνο κυριολεκτικών σταθερών καταντά σύντομα βαρετή – χρειαζόμαστε κάποιο τρόπο αποθήκευσης και επεξεργασίας κάθε πληροφορίας. Εδώ είναι που εφαρμόζονται οι μεταβλητές. Οι μεταβλητές είναι ακριβώς ότι το όνομα συνεπάγεται – οι τιμές τους ποικίλλουν, δηλαδή μπορείτε να αποθηκεύσετε οτιδήποτε με τη χρήση των μεταβλητών. Οι μεταβλητές είναι απλά τμήματα της μνήμης του Η/Υ σας όπου μπορείτε να αποθηκεύσετε πληροφορία. Αντίθετα με τις κυριολεκτικές σταθερές, χρειάζεστε κάποια μέθοδο (method) για να καλέσετε αυτές τις μεταβλητές αλλά και να τους δώσετε όνομα.

Ονοματοδοσία αναγνωριστικού

Οι μεταβλητές είναι παράδειγμα αναγνωριστικών. Τα αναγνωριστικά είναι δοθέντα ονόματα για να αναγνωρίσουν κάτι. Υπάρχουν ορισμένοι κανόνες που πρέπει να ακολουθήσετε για την ονοματοδοσία αναγνωριστικών:

- Ο πρώτος χαρακτήρας του αναγνωριστικού πρέπει να είναι ένα γράμμα της αλφαβήτου ή μια κάτω παύλα.
- Το υπόλοιπο του ονόματος του αναγνωριστικού μπορεί να αποτελείται από γράμματα, κάτω παύλες και αριθμούς.
- Στο όνομα ενός αναγνωριστικού διακρίνονται τα πεζά από τα κεφαλαία. Για παράδειγμα, το myname και το myName δεν είναι το ίδιο.
- Μπορείτε να χρησιμοποιείτε ελληνικές λέξεις ως ονόματα αναγνωριστικού

Τύποι δεδομένων

Οι μεταβλητές μπορούν να διατηρούν τιμές διαφορετικών τύπων που ονομάζονται τύποι δεδομένων. Οι βασικοί τύποι είναι αριθμοί και συμβολοσειρές, όπως ήδη αναφέραμε. Στη συνέχεια, θα δούμε πώς μπορούμε να δημιουργήσουμε δικούς μας τύπους χρησιμοποιώντας Κλάσεις.

Αντικείμενα

Πρέπει να θυμόμαστε ότι, η Python αντιλαμβάνεται οτιδήποτε χρησιμοποιείται σε ένα πρόγραμμα ως ένα αντικείμενο, με μια γενική έννοια της λέξης. Αντι να πούμε ‘το κατι’, λέμε ‘το αντικείμενο’.

Σημείωση για χρήστες αντικειμενοστρεφούς προγραμματισμού.

Η Python είναι έντονα αντικειμενοστρεφής με την έννοια ότι οτιδήποτε είναι ένα αντικείμενο συμπεριλαμβάνοντας αριθμούς συμβολοσειρές και συναρτήσεις.

Πώς να γράφετε προγράμματα python

Η τυπική διαδικασία εφεξής για την αποθήκευση και εκτέλεση ενός προγράμματος Python είναι η ακόλουθη:

1. Ανοίξτε τον επεξεργαστή κώδικα της προτίμισής σας.
2. Γράψτε τον κώδικα του προγράμματος που δίνεται στο παράδειγμα.
3. Αποθηκεύστε το ως αρχείο με το όνομα που αναφέρεται στο σχόλιο. Ακολουθώ την σύμβαση αποθήκευσης όλων των προγραμμάτων Python με την κατάληξη .py.
4. Εκτελέστε το διερμηνευτή με την εντολή `python var .py` ή απλούστερα χρησιμοποιήστε το IDLE για την εκτέλεση των προγραμμάτων. Μπορείτε να κάνετε επίσης τα αρχεία εκτελέσιμα όπως αναφέρθηκε παραπάνω.

Στη συνέχεια, θα δούμε τη χρήση μεταβλητών μαζί με κυριολεκτικές σταθερές μέσω ενός απλού παραδείγματος.

Εσοχή κώδικα (indentation)

Ο κενός χώρος είναι σημαντικός στην Python. Πραγματι, ο κενός χαρακτήρας στην αρχή της γραμμής είναι σημαντικός. Αυτό αποκαλείται εσοχή κώδικα. Οι αρχικοί κενοί χαρακτήρες (κενά και στηλοθέτες) στην αρχή της λογικής γραμμής καθορίζουν το επίπεδο εσοχής της λογικής γραμμής, και αυτό με τη σειρά του προσδιορίζει την ομαδοποίηση των εντολών.

Αυτό σημαίνει ότι οι εντολές που πάνε μαζί πρέπει και οφείλουν να έχουν το ίδιο επίπεδο εσοχής. Κάθε τέτοια ομάδα εντολών, καλείται πλοκάδα(block).

Προσοχή : Στην Python μετράνε οι χαρακτήρες διαστήματος!! Αν λοιπόν συναντήσετε ποτέ το ακόλουθο σφάλμα:

Indentation Error : expected an indented block

Σημαίνει ότι ξεχάσατε να τοποθετήσετε τους χαρακτήρες διαστήματος.

Χρήση της εσοχής

Να μην αναμγνύετε στηλοθέτες και κενά στις εσοχές διότι θα προκληθεί ασυμβατότητα μεταξύ διαφορετικών λειτουργικών. Σας συνιστώ τη χρήση είτε ενός μονού στηλοθέτη ή τεσσάρων κενών για κάθε επίπεδο εσοχής. Επιλέξτε ένα από τα δύο στυλ εσοχών. Και σημαντικότερο είναι να επιλέξετε ένα και να το χρησιμοποιείτε σταθερά, δηλαδή συνηθίστε αυτό το στυλ μόνο.

Σειρά υπολογισμού (προτεραιότητα τελεστών)

Ας υποθέσουμε ότι έχουμε μια έκφραση πχ $3+2*4$. Ποιο γίνεται πρώτα, η πρόσθεση ή ο πολλαπλασιασμός; Τα μαθηματικά μας λένε ότι πρώτα πρέπει να γίνει ο πολλαπλασιασμός. Αυτό σημαίνει ότι ο τελεστής του πολλαπλασιασμού έχει υψηλότερη προτεραιότητα από τον τελεστή της πρόσθεσης.

Ο πίνακας που ακολουθεί μας δίνει τον πίνακα προτεραιοτήτων για την Python, από τη χαμηλότερη προς την υψηλότερη προτεραιότητα. Αυτό σημαίνει ότι σε μία δεδομένη έκφραση, η Python θα υπολογίσει τους τελεστές και τις εκφράσεις που βρίσκονται χαμηλότερα στον πίνακα πριν από αυτούς που βρίσκονται ψηλότερα.

Ο παρακάτω πίνακας, αντιγραμμένος από το εγχειρίδιο αναφοράς της Python, παρέχεται χάριν πληρότητας. Είναι πολύ καλύτερο να χρησιμοποιείτε παρενθέσεις για να ομαδοποιείτε σωστά τελεστές και τελεστέους ώστε να ορίσετε ξεκάθαρα την προτεραιότητα. Έτσι το πρόγραμμα γίνεται πιο ευανάγνωστο.

6.2.5 Έλεγχος ροής

Στα προγράμματα υπάρχει πάντα μια σειρά εντολών, τις οποίες εκτελεί πιστά η Python με την ίδια σειρά. Αν θέλουμε όμως να αλλάξουμε την ροή εκτέλεσης; Αν, για παράδειγμα, θέλουμε το πρόγραμμα να πάρει μερικές αποφάσεις και να κάνει διαφορετικά πράγματα υπό διαφορετικές προϋποθέσεις. Αυτό επιτυγχάνεται χρησιμοποιώντας εντολές ελέγχου ροής. Υπάρχουν τρεις εντολές ελέγχου ροής στην Python - if, for και while.

Η εντολή if

Η εντολή if χρησιμοποιείται για να ελεγχθεί μια συνθήκη και εάν (if) η συνθήκη αυτή είναι αληθής, τότε εκτελείται ένα σύνολο ή πλοκάδα εντολών (που ονομάζεται if-block), διαφορετικά (else) γίνεται επεξεργασία ενός άλλου συνόλου εντολών (που ονομάζεται else-block). Η χρήση του όρου else είναι προαιρετική. Η εντολή if περιλαμβάνει άνω και κάτω τελεία στο τέλος – έτσι δηλώνουμε στην Python ότι ακολουθεί μια πλοκάδα εντολών.

Ο όρος elif ο οποίος συνδυάζει δύο συσχετιζόμενες εντολές if else-if else σε μία εντολή if-elif-else, χρησιμοποιείται για να κάνει το πρόγραμμα ευκολότερο και για να μειώσει τον αριθμό των εσοχών που απαιτούνται. Οι εντολές elif και else πρέπει επίσης να έχουν άνω και κάτω τελεία στο τέλος της λογικής γραμμής, ακολουθούμενες από τις αντίστοιχες πλοκάδες εντολών (με κατάλληλες εσοχές, βεβαίως). Μπορούμε να έχουμε άλλη μία εντολή if εντός της πλοκάδας if (if-block) μιας εντολής if κ.ο.κ. Αυτό αποκαλείται εμφωλευμένη εντολή if.

Να θυμηθούμε ότι τα τμήματα elif και else είναι προαιρετικά. Μια ελάχιστη έγκυρη εντολή if είναι:

```
if True:
```

```
    print('Ναι, είναι αληθές')
```

Αφού η Python τελειώσει την εκτέλεση ολόκληρης της εντολής if συμπεριλαμβανομένων και των συσχετιζόμενων όρων elif και else, προχωράει στην επόμενη εντολή στην πλοκάδα που περιέχει την εντολή if.

Η εντολή while

Η εντολή while μας επιτρέπει να εκτελούμε επανειλημμένα μια πλοκάδα εντολών, όσο μια προϋπόθεση παραμένει αληθής. Η εντολή while είναι ένα παράδειγμα αυτού που αποκαλείται εντολή βρόχου (looping statement). Μια εντολή while μπορεί να έχει έναν προαιρετικό όρο else.

Παράδειγμα:

```
number = 23
running = True
while running:
    guess = int(input('Εισάγετε έναν ακέραιο αριθμό : '))
    if guess == number:
        print('Συγχαρητήρια, τον μαντέψατε.')
        running = False # αυτό κάνει τον βρόχο while να σταματήσει εδώ
    elif guess < number:
        print('Όχι, είναι λίγο μεγαλύτερος.')
    else:
        print('Όχι, είναι λίγο μικρότερος.')
else:
```

```
print('Ο βρόχος while τερματίστηκε.')  
  
# Μπορείτε να προσθέσετε ότι άλλο θέλετε εδώ  
  
print('Τέλος')  
Έξοδος:
```

Πώς λειτουργεί:

Αυτό το πρόγραμμα είναι ένα παιχνίδι πρόγνωσης, αλλά το πλεονέκτημα είναι ότι επιτρέπει στον χρήστη να συνεχίσει να μαντεύει μέχρι να βρει τον σωστό αριθμό -δεν χρειάζεται δηλ. να εκτελεί επανειλημμένα το πρόγραμμα για κάθε πρόγνωση. Αυτό είναι ένα πολύ καλό παράδειγμα της χρήσης της εντολής `while`.

Μετακινούμε τις εντολές `input` και `if` εντός του βρόχου `while` και ορίζουμε την μεταβλητή `running` σε `True` πριν τον βρόχο `while`. Πρώτα ελέγχουμε αν η μεταβλητή `running` έχει την τιμή `True` και στην συνέχεια προχωράμε στην εκτέλεση της αντίστοιχης πλοκάδας `while` (`while-block`). Αφού εκτελεστεί αυτή η πλοκάδα, η προϋπόθεση ελέγχεται και πάλι, δηλ. στην περίπτωση μας η μεταβλητή `running`. Εάν η τιμή της συνεχίζει να είναι `true` (αληθές), εκτελείται και πάλι ολόκληρη η πλοκάδα `while` (`while-block`), διαφορετικά προχωράμε στην εκτέλεση της προαιρετικής πλοκάδας `else` (`else-block`) και μετά προχωράμε στην επόμενη εντολή.

Η πλοκάδα `else` εκτελείται όταν η προϋπόθεση του βρόχου `while` αποκτά την τιμή `False` (ψευδές) -αυτό μπορεί να συμβεί ακόμα και κατά την πρώτη φορά που ελέγχεται η προϋπόθεση. Εάν υπάρχει ένας όρος `else` για ένα βρόχο `while`, εκτελείται πάντοτε, εκτός κι αν διακόψετε τον βρόχο με την εντολή `break`.

Οι τιμές `True` (αληθές) και `False` (ψευδές) ονομάζονται Μπούλειαν τύποι (`Boolean types`) και μπορούμε να θεωρήσουμε ότι αντιστοιχούν στις τιμές 1 και 0 αντίστοιχα.

Συναρτήσεις

Οι συναρτήσεις είναι επαναχρησιμοποιήσιμα μέρη προγραμμάτων. Μας επιτρέπουν να δίνουμε ένα όνομα σε ένα σύνολο εντολών και να τρέχουμε εκείνο το σύνολο εντολών χρησιμοποιώντας το όνομά τους, οπουδήποτε στο προγράμμα μας και όσες φορές θέλουμε. Αυτό είναι γνωστό σαν κλήση (calling) της συνάρτησης.

Η έννοια των συναρτήσεων είναι πιθανόν το πιο σπουδαίο δομικό στοιχείο οποιουδήποτε μη στοιχειώδους προγράμματος (σε όλες τις γλώσσες προγραμματισμού), γι' αυτό θα διερευνήσουμε διάφορες πτυχές των συναρτήσεων.

Οι συναρτήσεις ορίζονται χρησιμοποιώντας τη λέξη κλειδί `def`, μετά την οποία ακολουθεί ένα όνομα που ταυτοποιεί την εκάστοτε συνάρτηση και κατόπιν ακολουθεί ένα ζευγάρι παρενθέσεων που μπορούν να περικλείουν μερικά ονόματα μεταβλητών, και η γραμμή τελειώνει με διπλή τελεία (`:`). Παρακάτω ακολουθεί ένα απλό παράδειγμα με ένα σύνολο εντολών που αποτελούν μέρος αυτής της συνάρτησης.

Παράδειγμα:

```
def sayHello():
```

```
    print('Hello World!') # σύνολο εντολών που ανήκουν στη συνάρτηση
```

```
# Τέλος της συνάρτησης
```

```
sayHello() # κλήση της συνάρτησης
```

Πώς δουλεύει:

Ορίζουμε μια συνάρτηση με το όνομα `sayHello` ακολουθώντας τη σύνταξη όπως εξηγήσαμε παραπάνω. Αυτή η συνάρτηση δεν έχει παραμέτρους γι' αυτό δε δηλώνονται καθόλου μεταβλητές ανάμεσα στις παρενθέσεις. Οι παράμετροι στις συναρτήσεις είναι απλά η είσοδος στη συνάρτηση ώστε να περνάμε διαφορετικές τιμές στη συνάρτηση και να παίρνουμε αντίστοιχα αποτελέσματα.

Να επισημάνουμε ότι μπορούμε να καλούμε την ίδια συνάρτηση δύο φορές, δηλαδή δε χρειάζεται να γράφουμε τον ίδιο κώδικα δύο φορές.

Παράμετροι συναρτήσεων

Μια συνάρτηση μπορεί να δεχθεί παραμέτρους, οι οποίες είναι τιμές που δίνετε στη συνάρτηση, έτσι ώστε αυτή να μπορεί να κάνει κάτι αξιοποιώντας αυτές τις τιμές. Αυτές οι παράμετροι μοιάζουν με τις μεταβλητές, διαφέροντας ως προς το ότι οι τιμές αυτών των μεταβλητών ορίζονται όταν καλούμε τη συνάρτηση και τους έχουν ήδη εκχωρηθεί τιμές όταν τρέχει η συνάρτηση.

Οι παράμετροι καθορίζονται μέσα στο ζευγάρι των παρενθέσεων στον ορισμό της συνάρτησης και διαχωρίζονται με κόμμα. Όταν καλούμε τη συνάρτηση δίνουμε και τις τιμές με τον ίδιο τρόπο. Σημείωση για την ορολογία που χρησιμοποιείται: οι ονομασίες που δίνετε στον ορισμό της συνάρτησης ονομάζονται παράμετροι ενώ οι τιμές που δίνετε όταν καλείτε τη συνάρτηση ονομάζονται ορίσματα.

Παράδειγμα:

```
def printMax(a, b):
```

```
    if a > b:
```

```
        print(a, 'είναι το μέγιστο')
```

```
    elif a == b:
```

```
        print(a, 'είναι ίσο με το', b)
```

```
    else:
```

```
        print(b, 'είναι το μέγιστο')
```

Στην πρώτη χρήση της printMax, απευθείας δίνουμε τους αριθμούς, δηλαδή τα ορίσματα. Στη δεύτερη χρήση της, καλούμε τη συνάρτηση χρησιμοποιώντας μεταβλητές.

Χρήση της εντολής global

Εάν θέλουμε να εκχωρήσουμε μια τιμή σε ένα όνομα που ορίζεται στο κορυφαίο επίπεδο του προγράμματος (δηλαδή όχι μέσα σε κάποιου είδους εμβέλεια όπως σε συναρτήσεις ή κλάσεις), τότε πρέπει να πούμε στην Python ότι το όνομα δεν είναι τοπικό αλλά καθολικό (global). Αυτό γίνεται με τη χρήση της εντολής global. Είναι αδύνατον να εκχωρήσουμε μια τιμή σε μια μεταβλητή που ορίζεται εκτός μιας συνάρτησης χωρίς την εντολή global. Μπορούμε να χρησιμοποιήσουμε τις τιμές τέτοιων μεταβλητών που ορίζονται έξω από τη συνάρτηση (υποθέτοντας ότι δεν υπάρχουν μεταβλητές με το ίδιο όνομα μέσα στη συνάρτηση). Χρησιμοποιώντας την εντολή global γίνεται ξεκάθαρο ότι η μεταβλητή βρίσκεται σε ένα εξωτερικό τμήμα εντολών.

Παράδειγμα:

```
x = 50
```

```
def func():
```

```
    global x
```

```
    print('Το x είναι', x)    x = 2    print('Άλλαξα το καθολικό x σε', x)
```

Πώς δουλεύει:

Η εντολή `global` χρησιμοποιείται για να δηλώσει ότι το `x` είναι μια καθολική μεταβλητή, γι' αυτό το λόγο όταν εκχωρούμε μια τιμή για το `x` μέσα στη συνάρτηση, αυτή η αλλαγή απεικονίζεται όταν χρησιμοποιούμε την τιμή του `x` στο κυρίως τμήμα.

Μπορείτε να καθορίσετε περισσότερες από μία καθολικές μεταβλητές χρησιμοποιώντας την ίδια εντολή `global`. Για παράδειγμα `global x, y, z`.

6.3 OpenCV

Προς ενίσχυση του κώδικα χρησιμοποιήσαμε την ανοιχτή βιβλιοθήκη λογισμικού OpenCV (*Open Source Computer Vision Library*) [7]. Η βιβλιοθήκη δημιουργήθηκε για να παρέχει εργαλεία για την όραση μέσω υπολογιστή σε εφαρμογές. Περιλαμβάνει πάνω από 2500 βέλτιστους αλγόριθμους. Αυτοί οι αλγόριθμοι μπορούν να χρησιμοποιηθούν για ανίχνευση και αναγνώριση αντικείμενων, προσώπων σε φωτογραφίες, να κατηγοριοποιήσουν ανθρώπινες πράξεις σε βίντεο, να ανιχνεύουν κίνηση σε κάμερες, να ανιχνεύουν τη κίνηση αντικειμένων, να εξάγουν τρισδιάστατα μοντέλα αντικειμένων και πολλές άλλες εφαρμογές. Η βιβλιοθήκη χρησιμοποιείται κατά κόρον από εταιρείες, ερευνητικές ομάδες και από κυβερνητικές οργανώσεις.

6.3.1 Hough Circle Transform

Ένας κύκλος αντιπροσωπεύεται μαθηματικά ως $(x - x_{\text{center}})^2 + (y - y_{\text{center}})^2 = r^2$ όπου $(x_{\text{center}}, y_{\text{center}})$ το κέντρο του κύκλου και r είναι η ακτίνα του κύκλου. Από την εξίσωση, μπορούμε να δούμε ότι έχουμε 3 παραμέτρους, οπότε χρειαζόμαστε έναν συσσωρευτή 3D για μετασχηματισμό hough, ο οποίος θα ήταν εξαιρετικά αναποτελεσματικός. Έτσι, η μέθοδος

που χρησιμοποιείται είναι ο κυκλικός μετασχηματισμός Hough επαυξημένο με την πληροφορία κλίσης των άκρων (Hough Gradient) όπως έχει επιγραφτεί από τους Kimme και Ballard [51]. Η εντολή συνάζεται όπως βλέπουμε στο παράδειγμα:

```
cv2.HoughCircles(img,cv2.HOUGH_GRADIENT, dp,  
minDist,param1=50,param2=30,minRadius=0,maxRadius=0)
```

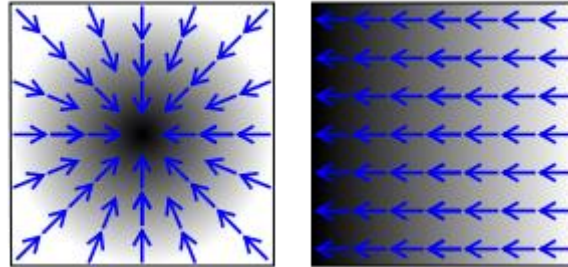
- **img:** είναι η εικόνα στην οποία ο αλγόριθμος θα εντοπίσει τα κυκλικά αντικείμενα(η εικόνα θα πρέπει να φορτώνεται σε κλίμακα του γκρι)
- **cv2.HOUGH_GRADIENT:** Είναι η μέθοδος με την οποία θα γίνει η ανίχνευση.
- **Dp:** Είναι η αντίστροφη αναλογία της ανάλυσης συσσωρευτή προς την ανάλυση της εικόνας. Μπορεί να πάρει δύο τιμές την 1 και την 2. Για την τιμή 1 έχει την ίδια ανάλυση με την εικόνα, ενώ όταν παίρνει την τιμή 2 έχει το μισό μήκος και πλάτος.
- **minDist:** Είναι η μικρότερη απόσταση που θα πρέπει να έχουν τα κέντρα των ανιχνεύσιμων κύκλων.
- **minRadius:** Είναι η μικρότερη ακτίνα που θα πρέπει να έχει ο κύκλος για να ανιχνευθεί.
- **maxRadius:** Είναι η μεγαλύτερη ακτίνα που θα πρέπει να έχει ο κύκλος για να ανιχνευθεί.

Πληροφορία κλίσης των άκρων

Η κλίση εικόνας είναι μια κατευθυντική αλλαγή στην ένταση ή στο χρώμα των εικόνων και χρησιμοποιείται για την εξαγωγή πληροφοριών από αυτές.

Στο λογισμικό γραφικών για ψηφιακή επεξεργασία εικόνας, ο όρος κλίση χρησιμοποιείται για τη σταδιακή μίξη των χρωμάτων που μπορεί να θεωρηθεί ως μια ακόμη διαβάθμιση από

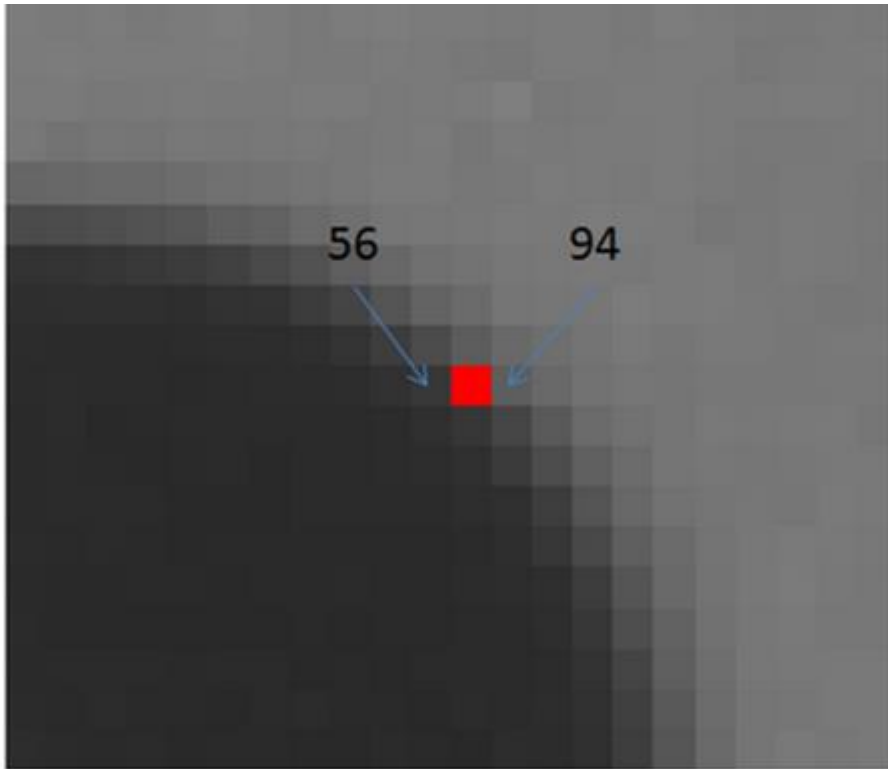
χαμηλές σε υψηλές τιμές, όπως χρησιμοποιείται από το άσπρο στο μαύρο στις εικόνες παρακάτω.



Σχήμα 6.3-(Δύο τύποι διαβαθμίσεων, με τα μπλε βέλη να δείχνουν την κατεύθυνση της κλίσης. Οι σκοτεινές περιοχές δείχνουν υψηλότερες τιμές)

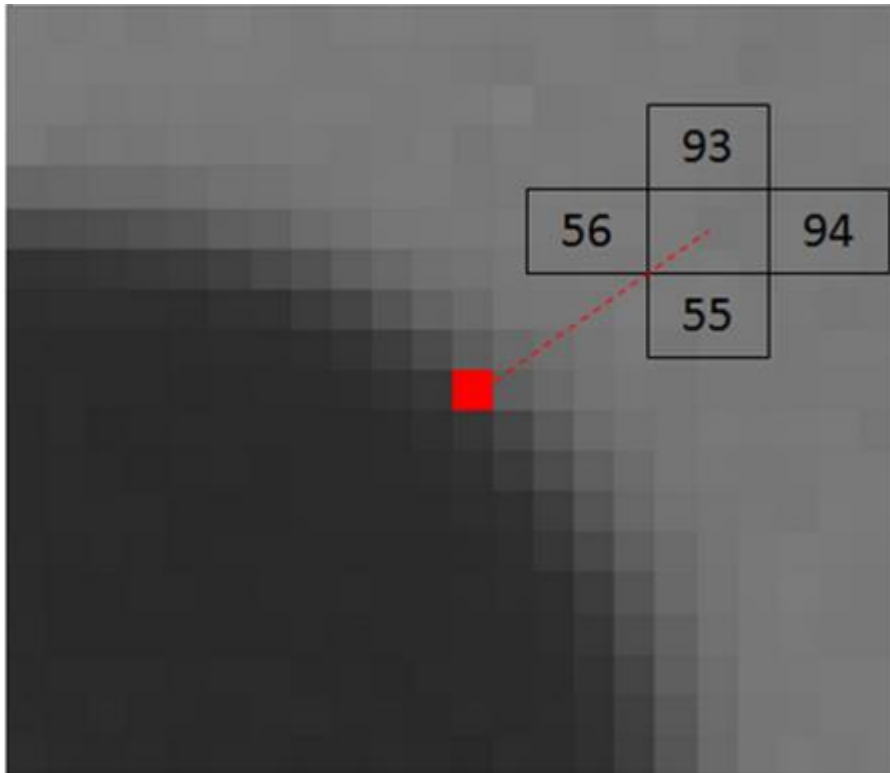
Στο λογισμό των μαθηματικών, η κλίση της συνάρτησης δυο μεταβλητών(σε αυτήν τη περίπτωση η συνάρτηση έντασης της εικόνας) σε κάθε σημείο της εικόνας είναι ένα δύο διαστάσεων διάνυσμα με τα στοιχεία να δίνονται από τις παραγώγους στις οριζόντιες και κάθετες διευθύνσεις. Σε κάθε σημείο της εικόνας, το διάνυσμα της κλίσης δείχνει την περιοχή με τη μεγαλύτερη δυνατή αύξηση της έντασης και το μήκος του διανύσματος κλίσης αντιπροσωπεύεται από τον ρυθμό της αλλαγής σε αυτή τη κατεύθυνση.

Η κλίση διανύσματος μπορεί να υπολογισθεί για κάθε ιχνοστοιχείο. Είναι ένας απλός υπολογισμός της αλλαγής των τιμών των ιχνοστοιχείων στη x-διεύθυνση και στη y-διεύθυνση γύρω από το ιχνοστοιχείο. Ας υποθέσουμε ότι θέλουμε να υπολογίσουμε τη κλίση στο χρωματισμένο ιχνοστοιχείο της παρακάτω εικόνας:



Σχήμα 6.4

Αυτή είναι μια γκρι εικόνα άρα οι τιμές θα είναι από 0-255(0 το μαύρο και 255 το άσπρο). Οι τιμές των ιχνοστοιχείων αριστερά και δεξιά έχουν τιμές 56 και 94. Αν αφαιρέσουμε το 56 από το 94 θεωρούμε ότι η αλλαγή στη x-διεύθυνση είναι 38. Σημείωση ότι μπορούμε να υπολογίσουμε τη διαφορά και να έχουμε ως αποτέλεσμα αρνητική τιμή. Αυτό σημαίνει ότι υπολογίσαμε από τις φωτεινές τιμές προς τις σκοτεινές. Το μόνο ζητούμενο είναι η διαδικασία αυτή να είναι ίδια για όλες τις τιμές. Στη συνέχεια κάνουμε το ίδιο για τα πάνω και κάτω ιχνοστοιχεία υπολογίζοντας την αλλαγή προς τη y-διεύθυνση:



Σχήμα 6.5

Το αποτέλεσμα είναι το ίδιο $38(93-55=38$ στη y- διεύθυνση)

Έχοντας και τις δυο τιμές μαζί μας δίνουν το διάνυσμα μας.

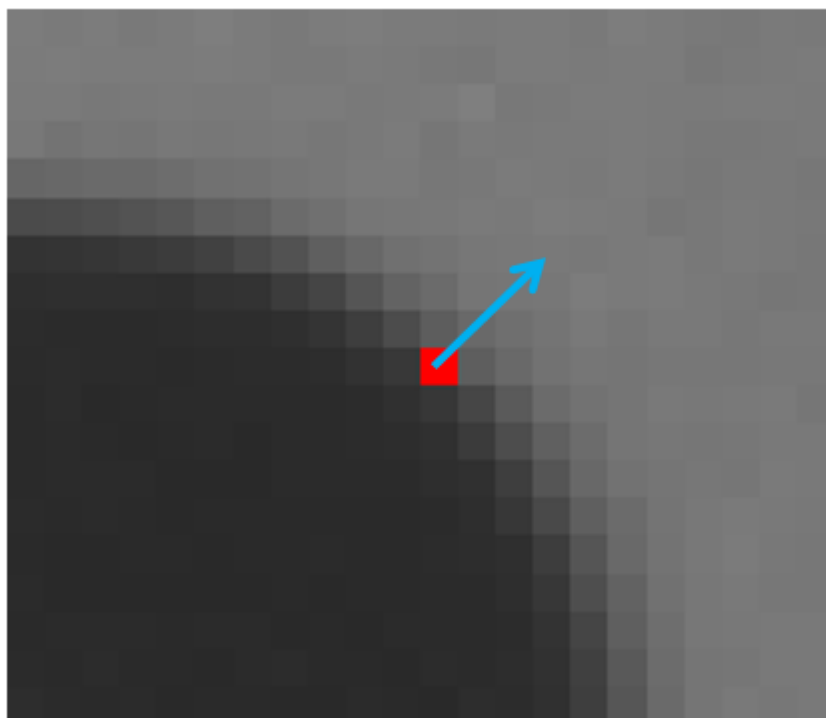
$$\begin{bmatrix} 38 \\ 38 \end{bmatrix}$$

Μπορούμε επιπλέον να χρησιμοποιήσουμε τις εξισώσεις για το πλάτος και τη γωνία του διανύσματος για τον υπολογισμό των τιμών:

$$\text{Πλάτος} = \sqrt{(38)^2 + (38)^2} = 53.74$$

$$\text{Γωνία} = \arctan\left(\frac{38}{38}\right) = 0.785\text{rads}(45^\circ)$$

Τώρα μπορούμε να σχεδιάσουμε το διάνυσμα στην εικόνα μας. Παρατηρούμε ότι η διεύθυνση του διανύσματος είναι κάθετο στη γωνία του κεφαλιού του πιγκούνου και αυτή είναι μια σημαντική ιδιότητα των διανυσμάτων.



Σχήμα 6.6

Ας δούμε τώρα το αποτέλεσμα από τον υπολογισμό όλων των αλλαγών στη x και y διεύθυνση κάθε ιχνοστοιχείου. Σημειώνουμε ότι η διαφορά από τις τιμές μπορεί να είναι από -255 έως 255 και αυτό έχει ως αποτέλεσμα να έχουμε να αποθηκεύσουμε πολλές τιμές σε ένα byte οπότε χρειάζεται να τις αλλάξουμε από 0 έως 255. Έχοντας κάνει αυτή την αλλαγή οι τιμές των ιχνοστοιχείων με τη μεγαλύτερη αρνητική τιμή θα είναι μαύρες και οι τιμές με τη μεγαλύτερη θετική αλλαγή άσπρες και οι τιμές με λίγη ή με καθόλου αλλαγή θα είναι γκρι.

Αλλαγή στη x -διεύθυνση



Αλλαγή στη y -διεύθυνση



Σχήμα 6.7- Διαφορές σύμφωνα με την διεύθυνση των ιχνοστοιχείων.

6.3.2 Σχεδιασμός κύκλων

Η μία δυνατότητα που μας δίνει η OpenCV είναι να ανιχνεύει αντικείμενα αλλά πως θα μπορέσουμε να υποδείξουμε το αντικείμενο που ανίχνευσε. Η OpenCV έχει αλγορίθμους οι οποίοι μας επισημάνουν τα αντικείμενα που ανιχνευτήκαν. Εμείς για να επισημάνουμε τους κύκλους που ανιχνεύει χρησιμοποιήσαμε δύο λειτουργίες. Η μια για να μας ζωγραφίζει έναν κύκλο ο οποίος θα περιβάλει το επιθυμητό αντικείμενο και η δεύτερη να μας προβάλει αυτή την εικόνα.

`cv2.circle(img, center, radius, color, thickness, lineType)`

- **img:** Είναι η εικόνα που θα επεξεργαστούμε.
- **Center:** Είναι το κέντρο του κύκλου ο οποίος θα μας υποδεικνύει το αντικείμενο που ανίχνευσε.
- **Radius:** Είναι η ακτίνα του κύκλου αυτού
- **Color:** Είναι το χρώμα του κύκλου.
- **lineType:** Είναι το είδος της γραμμής που θα χρησιμοποιήσει

`cv2.imshow('image',img)`

- **'image':** Είναι το όνομα του παραθύρου που θα εμφανιστεί η εικόνα.
- **Img:** Είναι το όνομα της εικόνας που θα εμφανιστεί.

6.3.3 Διαδικασία εγκατάστασης βιβλιοθήκης OpenCV

Απαραίτητη προϋπόθεση για την ανάπτυξη του λογισμικού μας είναι η εγκατάσταση της βιβλιοθήκης OpenCV στο Raspberry Pi2[59] επεκτείνοντας έτσι δυνατότητα του συστήματος για χρήση μηχανικής όρασης. Η εγκατάσταση περιλαμβάνει τη διαδικασία λήψης και ενεργοποίησης πακέτων όπως αυτή περιγράφεται στη συνέχεια.

Εγκαθιστούμε τα απαραίτητα πακέτα και αναπτυξιακά πακέτα εκτελώντας την εντολή

```
sudo apt-get install build-essential git cmake pkg-config
```

Εγκαθιστούμε τα απαραίτητα πακέτα για την επικοινωνία εισόδου/εξόδου τα οποία μας επιτρέπουν να φορτώνουμε διαφορετικά μορφότυπα εικόνων όπως jpeg, png, tiff, κ.α.

```
sudo apt-get install libjpeg-dev libtiff5-dev libjasper-dev libpng12-dev
```

Εγκαθιστούμε την ανεπτυγμένη βιβλιοθήκη GTK. Αυτή η βιβλιοθήκη χρησιμοποιείται για την ανάπτυξη γραφικής διεπαφής χρήστη (GUIs) και απαιτείται από τη βιβλιοθήκη highgui του OpenCV που μας επιτρέπει να προβάλουμε εικόνες στην οθόνη.

```
sudo apt-get install libgtk2.0-dev
```

Εγκαθιστούμε τα απαραίτητα πακέτα εισόδου/εξόδου τα οποία μας επιτρέπουν να φορτώνουμε αρχεία βίντεο με χρήση της OpenCV.

sudo apt-get install libavcodec-dev libavformat-dev libswscale-dev libv4l-dev

sudo apt-get install libxvidcore-dev libx264-dev

Εγκαθιστούμε βιβλιοθήκες, για τη βελτιστοποίηση ποικίλων λειτουργιών από την OpenCV.

sudo apt-get install libatlas-base-dev gfortran

Εγκαθιστούμε το virtual environment (virtualenv) και το virtual environment wrapper (virtualenvwrapper)

sudo pip install virtualenv virtualenvwrapper

sudo rm -rf ~/.cache/pip

Επεκτείνουμε το προφίλ χρήστη (~/.profile) με τις ακόλουθες εντολές

export WORKON_HOME=\$HOME/.virtualenvs

source /usr/local/bin/virtualenvwrapper.sh

Φορτώνουμε το ανανεωμένο προφίλ χρήστη

source ~/.profile

Δημιουργούμε το δικό μας εικονικό περιβάλλον χρήστη

mkvirtualenv cv

Στο επόμενο βήμα εγκαθιστούμε τα αναπτυξιακά εργαλεία της Python 2.7

sudo apt-get install python2.7-dev

Τέλος εγκαθιστούμε το NumPy καθώς η βιβλιοθήκη OpenCV αντιμετωπίζει τα αρχεία εικόνων ως πολυδιάστατους NumPy πίνακες.

pip install numpy

Κατεβάζουμε το OpenCV και το αποσυμπιέζουμε

cd ~

wget -O opencv.zip https://github.com/Itseez/opencv/archive/3.0.0.zip

unzip opencv.zip

Ολοκληρώνουμε με τη μεταγλώττιση του OpenCV

cd ~/opencv-3.0.0/

mkdir build

cd build

***cmake -D CMAKE_BUILD_TYPE=RELEASE ***

***-D CMAKE_INSTALL_PREFIX=/usr/local ***

***-D INSTALL_C_EXAMPLES=ON ***

***-D INSTALL_PYTHON_EXAMPLES=ON ***

***-D OPENCV_EXTRA_MODULES_PATH=~/*opencv_contrib-3.0.0/modules* ***

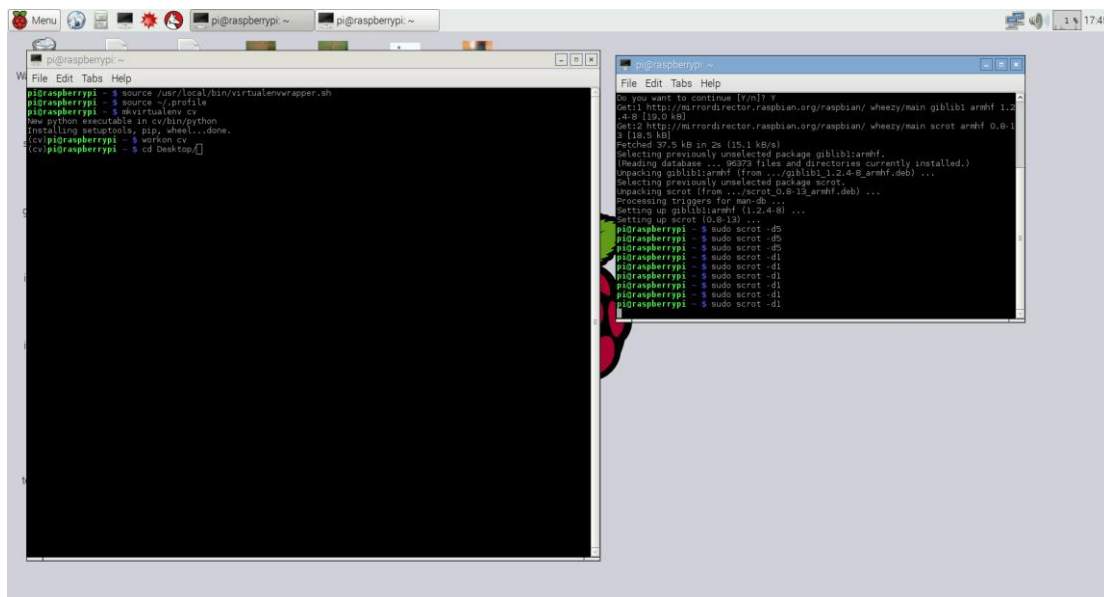
-D BUILD_EXAMPLES=ON ..

make

6.3.4 Διαδικασία εκτέλεσης προγράμματος

Στην παράγραφο αυτή περιγράφουμε τη διαδικασία που πρέπει να ακολουθήσουμε ώστε να τρέξει το πρόγραμμα μας. Σε περιβάλλον Raspbian ανοίγουμε το τερματικό παράθυρο εκτέλεσης εντολών(terminal), και εκτελούμε διαδοχικά τις παρακάτω εντολές όπως φαίνονται και στο σχήμα 6.8:

- *source ~/.profile*
- *mkvirtualenv cv*
- *workon cv*



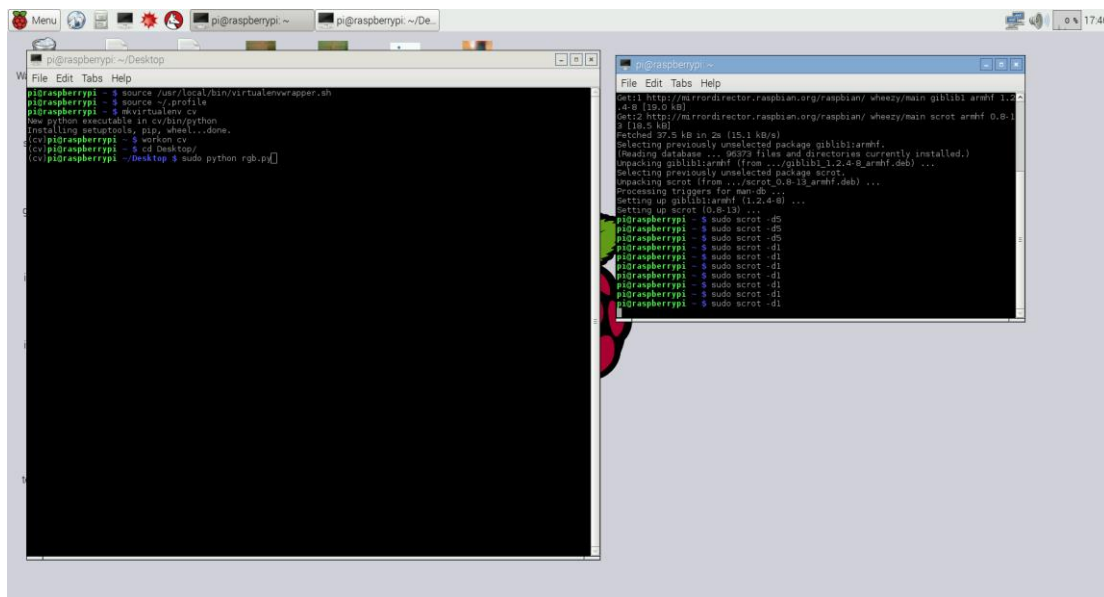
Σχήμα 6.8 - Οι εντολές για το εικονικό περιβάλλον

Για λόγους εξοικονόμησης χρόνου οι εντολές αυτές έχουν γραφτεί σε ένα σενάριο « script» με όνομα (startscript.sh)(βλέπε Παράρτημα Σχήμα 12). Αυτό το script εκτελεί τις παραπάνω εντολές με το που ανοίγουμε το τερματικό παράθυρο εκτέλεσης εντολών.

Έχοντας εκτελέσει τις απαραίτητες αλλαγές στις μεταβλητές περιβάλλοντος χρήστη και έχοντας φορτώσει τις απαραίτητες βιβλιοθήκες είμαστε έτοιμοι για την εκτέλεση του προγράμματος μας.

Εισάγεται στο τερματικό την (Σχήμα 6.9):

sudo python vehicle.py

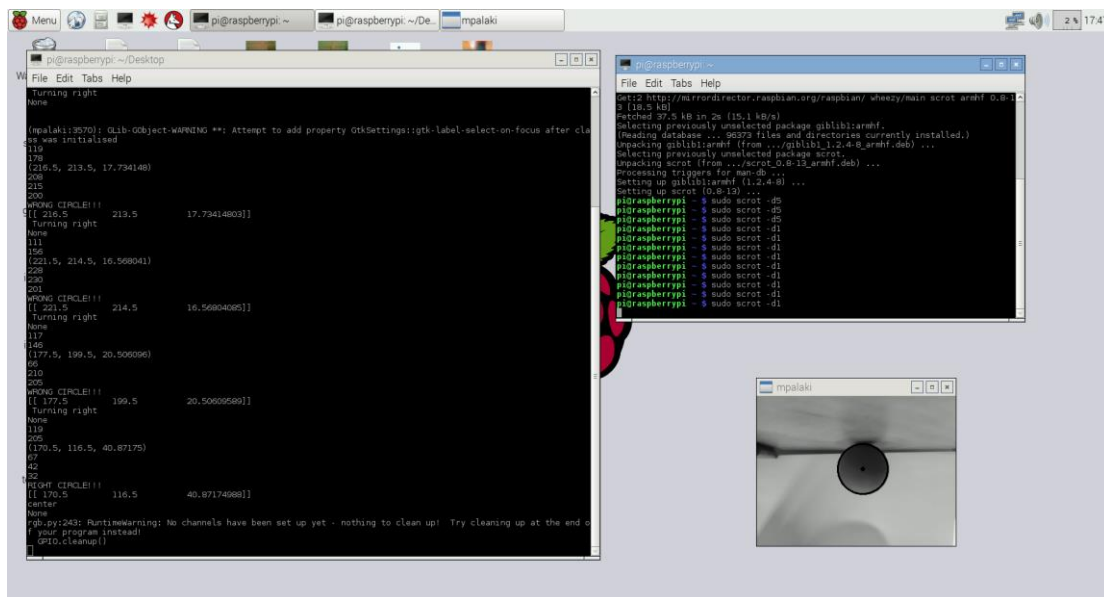


Σχήμα 6.9 - Εντολή εκτέλεσης του κώδικα

Ο αλγόριθμος που ακολουθεί το πρόγραμμα μας εκτελεί κάποια βήματα, τα οποία θα αναλυθούν στην ενότητα 7. Το πιο σημαντικό βήμα είναι ο έλεγχος του χρώματος στην περιοχή που ανιχνεύτηκε το μπαλάκι και η ταυτοποίησή του.

Για την σωστή εκτέλεση του αλγορίθμου έχουμε τις εξής παραδοχές:

Για να μειώσουμε την πιθανότητα λάθους, το περιβάλλον στο οποίο κινείται και κατ' επέκταση φωτογραφίζει το όχημα δεν έχει αντικείμενα ή φόντο ίδιου χρώματος με το μπαλάκι. Η μοναδική περιοχή με το επιθυμητό χρώμα είναι αυτή που ανήκει στο μπαλάκι το οποίο και πρέπει να ανιχνεύσει. Για την αποφυγή εντοπισμού άλλων σφαιρικών αντικειμένων θεωρούμε επίσης ότι το μοναδικό σφαιρικό αντικείμενο στο περιβάλλον είναι το μπαλάκι.



Σχήμα 6.10 – Λειτουργία προγράμματος

Κεφάλαιο 7^ο – Ανάπτυξη αλγορίθμου ακολούθησης αντικειμένου

Εισαγωγή

Στο παρόν κεφάλαιο ασχολούμαστε με την περιγραφή του αλγόριθμο που αναπτύξαμε για την ακολούθηση αντικειμένου από το μηχανοκίνητο όχημα μας. Επιλέξαμε ως αντικείμενο το οποίο ακολουθεί το όχημα μας ένα μπαλάκι διαμέτρου 10cm (Σχήμα 7.1). Η επιλογή του σφαιρικού αντικειμένου έγινε για να έχουμε τη δυνατότητα να μετακινούμε το αντικείμενο μας με ευκολία προς όλες τις κατευθύνσεις του επιπέδου, αλλά και να συνεχίζει να κινείται μετά έπειτα κάθε εφαρμογή δύναμης σ' αυτό.

Συνεπώς το πρόβλημα που καλούμαστε να επιλύσουμε είναι η παρακολούθηση της μικρής μπάλας από το όχημα μας έτσι ώστε να διατηρεί μία σταθερή σχετική θέση από αυτό έτσι ώστε να βρίσκεται διαρκώς μπροστά του. Για την επίλυση του προβλήματος αυτού χρησιμοποιήσαμε τεχνικές μηχανικής όρασης σε συνδυασμό με ψηφιακή επεξεργασία εικόνας.



Σχήμα 7.1- Αντικείμενα προς αναζήτηση

7.1 Περιγραφή- Οδηγίες Λειτουργιάς

Διακρίνουμε δύο βασικές λειτουργίες του αυτόνομου οχήματος μας. Στην πρώτη λειτουργία ο μικροϋπολογιστής Raspberry Pi2 και κατ' επέκτασή το όχημα είναι συνδεδεμένο με οθόνη πληκτρολόγιο και ποντίκι και η πλήρης εκτέλεση της εφαρμογής που έχουμε αναπτύξει στα πλαίσια της παρούσας πτυχιακής εργασίας μπορεί να παρακολουθηθεί από τον τελικό χρήστη καθ' όλη τη διάρκεια.

Στη δεύτερη λειτουργία απουσιάζει η διασύνδεση του μικροϋπολογιστή με την οθόνη και στην επικοινωνία οχήματος-χρήστη διακρίνουμε τις εξής καταστάσεις: εκκίνηση της εφαρμογής μέσω ασύρματου πληκτρολογίου, δυνατότητας τερματισμού της ή τροποποίησης παραμέτρων αυτής μέσω διακόπτη που βρίσκεται πάνω στο όχημα, ενώ κατά τη διάρκεια εκτέλεσης του προγράμματος μας υπάρχει η δυνατότητα ελέγχου της φάσης του κώδικα που εκτελείται μέσω ενδεικτικής λυχνίας LED που είναι τοποθετημένη στην κορυφή του οχήματος.

7.1.1 ΛΕΙΤΟΥΡΓΙΑ ΜΕ ΑΠΟΥΣΙΑ ΟΘΟΝΗΣ ΠΡΟΒΟΛΗΣ

Κατά τη λειτουργία χωρίς τη χρήση οθόνης προβολής η παροχή του αυτόνομου οχήματος γίνεται αποκλειστικά από δύο πηγές τροφοδοσίας: η πρώτη πηγή τροφοδοσίας (βλέπε Παράρτημα Σχήμα 4) είναι ένας συσσωρευτής τύπου Power Bank (10000mAh-5.0V) και παρέχει την απαραίτητη ενέργεια στο μικροϋπολογιστή για την αδιάλειπτη λειτουργία του, ενώ δεύτερη πηγή αποτελείται από μία διάταξη έξι συσσωρευτών 1,5V AA (βλέπε Παράρτημα Σχήμα 4) η οποία τροφοδοτεί τους τέσσερις κινητήρες συνεχούς τάσης του οχήματος.

Η λειτουργία του μικροϋπολογιστή αρχίζει αυτόματα με την εφαρμογή τάσης σε αυτόν. Για την εκκίνηση του λειτουργικού του συστήματος απαιτείται χρονικό διάστημα περίπου 1 λεπτού, ενώ μετά το πέρας αυτού μπορούμε να ξεκινήσουμε την εφαρμογή μας.

Για τη φόρτωση στη μνήμη και εκτέλεση του προγράμματος μας, έχουμε παραμετροποιήσει το λειτουργικό σύστημα (Raspbian) ώστε:

- το πάτημα του πλήκτρου <F5> να ανοίγει το παράθυρο του τερματικού Terminal
- με την έναρξη του τερματικού παραθύρου Terminal αρχίζει αυτόματα η εκτέλεση του προγράμματος χωρίς την πληκτρολόγηση της αντίστοιχης εντολής.

Με το πάτημα του πλήκτρου <F5> αρχίζει η φάση Α (βλέπε Παράρτημα Σχήμα 21) στην οποία παρατηρούμε ότι το λαμπάκι ενδεικτικής λυχνίας LED ανάβει στο μπλε χρώμα επιβεβαιώνοντας την έγκυρη εκκίνηση του προγράμματος μας. Στη κατάσταση αυτή μπορούμε να επιλέξουμε το χρώμα του αντικειμένου-μπάλας το οποίο θέλουμε να ανιχνεύει το όχημα μας στην φάση Β. Το χρονικό διάστημα πριν την ολοκλήρωση της Α είναι 4sec αλλά ανανεώνεται σε κάθε νέα επιλογή χρώματος. Η επιλογή γίνεται με το πάτημα του μικροδιακόπτη όπως φαίνεται στο σχήμα(βλέπε Παράρτημα Σχήμα 21). Συγκεκριμένα σε κάθε πάτημα του αλλάζουμε κυκλικά το επιλεγόμενο χρώμα με σειρά ΜΠΛΕ-ΠΡΑΣΙΝΟ-ΚΟΚΚΙΝΟ.

Η ολοκλήρωση της Α οδηγεί το πρόγραμμα στη φάση Β όπου πλέον η ενδεικτική λυχνία αναβοσβήνει στο χρώμα που έχουμε επιλέξει σε κάθε λήψη εικόνας. Το όχημα στην κατάσταση αυτή λαμβάνει στιγμιότυπα τα επεξεργάζεται με τεχνικές μηχανικής όρασης και εκτελεί τις κατάλληλες κινήσεις, διαδικασία ή οποία επαναλαμβάνεται μέχρις ότου πατήσουμε το μικροδιακόπτη. Οι επιλογές με το μικροδιακόπτη είναι δύο: μονό πάτημα μικρής διάρκειας επαναφέρει το πρόγραμμα στην εκ' νέου επιλογή χρώματος (κατάσταση Α, LED διαρκώς αναμμένο) ενώ παρατεταμένο πάτημα διάρκειας 5sec τερματίζει το πρόγραμμα. Σε κάθε περίπτωση πάτημα του διακόπτη συνοδεύεται από άναμμά και σβήσιμο της ενδεικτικής λυχνίας με φθίνουσα συχνότητα.

7.1.2 ΛΕΙΤΟΥΡΓΙΑ ΜΕ ΟΘΟΝΗΣ ΠΡΟΒΟΛΗΣ

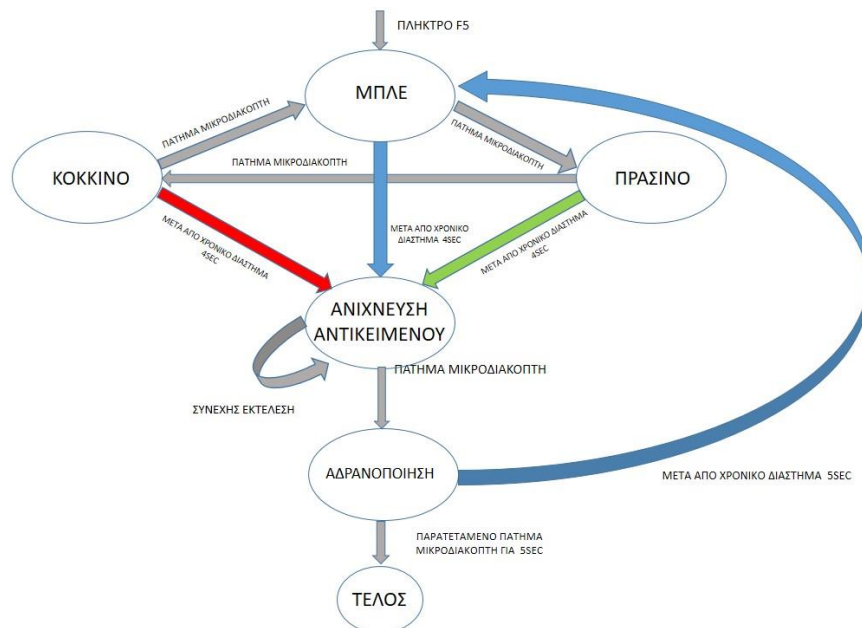
Στη λειτουργία της εφαρμογής μας με συνδεδεμένη την οθόνη προβολής μας παρέχεται ένα ολοκληρωμένο περιβάλλον γραφικής διασύνδεσης με το πρόγραμμα μας και ελέγχου του από πληκτρολόγιο και ποντίκι.

Η εκκίνηση του προγράμματος μπορεί να γίνει είτε με το πλήκτρο <F5> είτε μέσα από το τερματικό παράθυρο (Terminal) με την εντολή `'sudo python vehicle.py'`, όπου `vehicle.py` το όνομα του αρχείου. Οι φάσεις που εκτελούνται είναι ίδιες με αυτές της λειτουργίας με απουσία οθόνης προβολής, ο χειρισμός αλλαγής χρώματος γίνεται πάλι από το μικροδιακόπτη αλλά ο τερματισμός του μπορεί να γίνει και με το κλείσιμο του προγράμματος από το Terminal (συνδυασμός πλήκτρων `Ctrl+c`).

Κατά τη διάρκεια που η κάμερα του οχήματος μας λαμβάνει στιγμιότυπα αυτά προβάλλονται στην επιφάνεια εργασίας επανυξημένα κατά τους κύκλους που έχει ανιχνεύσει η βιβλιοθήκη μηχανικής όρασης OpenCV σημειώνοντας του με μαύρο περίγραμμα. Ο χρόνος παραμονής του εικονιδίου στην επιφάνεια εργασίας καθορίζεται από τον χρόνο που του έχουμε βάλει την εντολή `WaitKey()`.

7.2 Μηχανή πεπερασμένων καταστάσεων

Ο αλγόριθμος που αναπτύξαμε περιγράφεται από μία μηχανή πεπερασμένων καταστάσεων(FSM). Το πλήρες διάγραμμα της μηχανής αυτής παρουσιάζεται στο σχήμα 7.2.



Σχήμα 7.2 – Σχήμα εναλλαγής καταστάσεων(FSM)

Με το πάτημα του πλήκτρου εκκίνησης F4 ο αλγόριθμος μας βρίσκεται στην αρχική κατάσταση SB ο οποία πληροφορεί το όχημα μας ότι πρόκειται να ανιχνευτεί αντικείμενο-μπαλάκι χρώματος μπλε. Το πάτημα του μικροδιακόπτη εντός χρονικού πλαισίου 4sec οδηγεί κυκλικά τη μηχανή μας στις καταστάσεις $SG \rightarrow SR \rightarrow SB \rightarrow SG \rightarrow \dots$

Μετά το πέρας των 4sec μεταβαίνουμε αυτόματα στην κύρια κατάσταση του αλγορίθμου μας την SD. Σε αυτήν την κατάσταση λαμβάνουμε διαρκώς στιγμιότυπα του χώρου μπροστά από το όχημα και εκτελούμε κατάλληλες κινήσεις ώστε να πετύχουμε παρακολούθηση της πορείας του αντικειμένου.

Με το πάτημα του μικροδιακόπτη ο αλγόριθμος μεταβαίνει στην κατάσταση αδρανοποίησης SP. Σε αυτήν την κατάσταση έχει διακοπεί η διαδικασία ανίχνευσης και ένα παρατεταμένο πάτημα του μικροδιακόπτη οδηγεί το σύστημα στη διακοπή του αλγορίθμου σε διαφορετική περίπτωση δηλαδή αν δεν πατηθεί ο μικροδιακόπτης το σύστημα μετά από χρονικό διάστημα 5sec η μηχανή πεπερασμένων καταστάσεων μεταβαίνει στην αρχική κατάσταση SB.

7.3 Αναλυτική περιγραφή της κατάστασης SD

λήψη στιγμιότυπου

- εντοπισμός με τεχνικές μηχανικής όρασης της περιοχής της εικόνας που βρίσκεται το μπαλάκι, ανιχνεύοντας το κυκλικό περίγραμμα του στο δισδιάστατο χώρο
- ψηφιακή επεξεργασία της εικόνας ώστε να επιβεβαιωθεί ότι στην περιοχή που ανιχνεύτηκε από το προηγούμενο βήμα υπάρχει το μπλε μπαλάκι.
- λήψη απόφασης για την επόμενη κίνηση του μηχανοκίνητου οχήματος.
- επανάληψη όλων των προηγούμενων βημάτων για τη συνέχιση της διαδικασίας μέχρι ότου επιλεγεί τερματισμός από το χρήστη.

7.3.1 Λήψη στιγμιότυπου

Το πρώτο βήμα του αλγορίθμου μας είναι η λήψη μίας εικόνας-στιγμιότυπο της περιοχής του χώρου που βρίσκεται μπροστά από το μηχανοκίνητο όχημα μας. Η εικόνα αυτή είναι έγχρωμη σε μορφότυπο jpg. Για την εκτέλεση των επόμενων βημάτων είναι απαραίτητη η αποθήκευση της σε ένα επιπλέον αντίγραφο ίδιου μορφότυπου αλλά με χρώματα που περιέχουν τις διαβαθμίσεις του γκρι. Το στιγμιότυπο έχει ανάλυση 320x240 , ιδανική στο να μην παραμορφώνεται η εικόνα και να μειώνει τον χρόνο εκτέλεσης του αλγορίθμου.

7.3.2 Ανίχνευση του κυκλικού περιγράμματος

Φορτώνοντας την εικόνα που περιέχει τις διαβαθμίσεις του γκρι, ο αλγόριθμος εντοπίζει πάνω στο δισδιάστατο χώρο της εικόνας όλα τα κυκλικά περιγράμματα. Η εντολή

που χρησιμοποιούμε είναι cv2.HoughCircles και τα αποτελέσματά της αποθηκεύονται σε ένα πίνακα. Ο πίνακας περιέχει τις συντεταγμένες των κέντρων των κύκλων καθώς και την ακτίνα τους.

7.3.3 Επιβεβαίωση της ανίχνευσης με ψηφιακή επεξεργασία εικόνας

Για να γίνει πιο ακριβής ο αλγόριθμος προστέθηκε η ψηφιακή επεξεργασία της εικόνας. Στην ψηφιακή επεξεργασία εντοπίζονται περιοχές της εικόνας που περιέχουν έντονο το μπλε χρώμα. Αυτές οι περιοχές συγκρίνονται με τα κέντρα των κύκλων που βρήκαμε και αν ταυτίζονται με μια τυπική απόκλιση τότε το μπαλάκι έχει εντοπιστεί με πολύ μεγάλη ακρίβεια.

7.3.4 Απόφαση κίνησης του οχήματος

Αφού επιβεβαιωθεί ότι εντοπίστηκε το μπαλάκι και έχουμε τις συντεταγμένες του κέντρου του και την ακτίνα του, ο αλγόριθμος πρέπει να πάρει κάποιες αποφάσεις πως θα κινηθεί. Μετά από κάποιες πειραματικές δοκιμές που πραγματοποιήθηκαν υπό συγκεκριμένες συνθήκες. Προέκυψαν κάποιες εξισώσεις οι οποίες χρησιμοποιήθηκαν για την κίνηση του οχήματος. Το όχημα δεν έχει κάποιο αισθητήριο το οποίο να μετράει την απόσταση, Έτσι με την βοήθεια τη εικόνας και της εξίσωσης ο αλγόριθμος μεταφράζει την ακτίνα σε απόσταση($l=880.18/x$) (βλέπε Παράρτημα Σχήματα 19-20). Το επόμενο βήμα είναι η μετατροπή της απόστασης σε χρόνο. Αυτό επιτεύχθηκε με την βοήθεια της εξίσωσης ($time_to_move=(33-l)/33$) αν το αντικείμενο είναι κοντά στο όχημα και με την εξίσωση ($time_to_move=(l-33)/l$) αν είναι μακριά. Μετά τον υπολογισμό του χρόνου είναι έτοιμο να πραγματοποιήσει τις κινήσεις του. Πρώτα στρέφεται ώστε το όχημα να είναι σε μία ευθεία με το μπαλάκι. Το επόμενο βήμα είναι να κινηθεί προς το μπαλάκι για χρόνο t βρήκαμε από την εξίσωση.

7.4 Κινήσεις οχήματος για εντοπισμό μπάλας

Το όχημα πραγματοποιεί μια σειρά από κινήσεις για να εντοπίσει το αντικείμενο που αναζητά. Οι κινήσεις του αυτές μπορούν να παρομοιαστούν όμοιες με αυτές του ανθρώπου, όταν αναζητά κάποιο χαμένο αντικείμενο. Σε κάθε κίνηση το όχημα λαμβάνει στιγμιότυπα και ανιχνεύει αν υπάρχει το επιθυμητό αντικείμενο. Αν εντοπιστεί τότε σταματάει να εκτελεί τις κινήσεις εντοπισμού της μπάλας και κινείται προς το αντικείμενο. Οι 3 πρώτες κινήσεις ελέγχουν αν το αντικείμενο βρίσκεται στην δεξιά μεριά του οχήματος. Όπως βλέπουμε στο σχήμα... το όχημα στρέφεται 2 φορές δεξιά και λαμβάνει 2 στιγμιότυπα για κάθε στροφή. Αν δεν εντοπισθεί το αντικείμενο τότε το όχημα πηγαίνει στην αρχική του θέση και λαμβάνει πάλι στιγμιότυπο. Στις επόμενες 3 κινήσεις το όχημα ελέγχει την αριστερή του μεριά. Όπως βλέπουμε στο σχήμα... το όχημα στρέφεται 2 φορές πάλι αλλά αυτή την φορά με κατεύθυνση αριστερά και λαμβάνει 1 στιγμιότυπα για κάθε στροφή. Αν δεν ανιχνευθεί το αντικείμενο τότε επιστρέφει πάλι στην αρχική του θέση λαμβάνοντας πάλι στιγμιότυπο. Πραγματοποιώντας τις 6 βασικές του κινήσεις και μην έχοντας εντοπίσει το αντικείμενο, το όχημα θα στραφεί κατά 180° μοίρες και θα πραγματοποιήσει τις ίδιες κινήσεις για την αντίθετη κατεύθυνση. (βλέπε Παράρτημα Σχήματα 13-18)

7.5 Ανάλυση και επεξήγηση κώδικα

Στις προηγούμενες υποενότητες αναλύσαμε τις λειτουργίες του οχήματος και πήραμε μια γενική ιδέα πως λειτουργεί. Στην παρούσα υποενότητα θα αναλύσουμε τον κώδικα, που εκτελεί η παρούσα εργασία. Ο κώδικας επιτρέπει στο λογισμικό να συνεργάζεται αρμονικά με το υλικό κομμάτι της εργασίας. Θα αναφερθούμε στον κώδικα τμηματικά γιατί μας

επιτρέπουν καλύτερη ανάλυσή του και την ευκολότερη κατανόησή του από τον αναγνώστη. Όλος ο κώδικας που χρησιμοποιήσαμε βρίσκεται αυτούσιος στο κεφάλαιο Κώδικα.

Βιβλιοθήκες

Η εισαγωγή των βιβλιοθηκών γίνεται στην αρχή του κώδικα. Μεν την εντολή `<<import>>` στην αρχή της εντολής. Δηλώνουμε ότι θέλουμε να εισάγουμε μια βιβλιοθήκη η οποία να υποστηρίζεται από την γλώσσα προγραμματισμού, στην συγκεκριμένη περίπτωση την Python. Οι βιβλιοθήκες που εισήχθησαν είναι:

- **import sys** : εισάγουμε την βιβλιοθήκη sys
- **sys.path.append('/usr/local/lib/python2.7/site-packages')** : δηλώνουμε την διαδρομή που θα πρέπει να ακολουθήσει η γλώσσα Python για την εντόπιση της βιβλιοθήκης OpenCV.
- **import time**: εισάγουμε την βιβλιοθήκη time
- **import picamera**: για την λειτουργία της κάμερας
- **import RPi.GPIO as GPIO**: για να λειτουργούμε τις διεπαφές
- **from time import sleep**: εισάγουμε την λειτουργία sleep
- **import numpy as np**: για να μπορούμε να λειτουργούμε πολυδιάστατους πίνακες
- **import cv2**: εισαγωγή της OpenCV

Μεταβλητές

Οι μεταβλητές, που δηλώνονται στην αρχή του κώδικα, κάτω από τις βιβλιοθήκες, αποκτούν την ιδιότητα GLOBAL. Δηλαδή μπορούν να χρησιμοποιηθούν οποιαδήποτε στιγμή

μέσα στον κώδικα χωρίς να τις δηλώσεις ξανά. Οι κατηγορίες που χώρισα τις μεταβλητές μου είναι :

- Η DEFINE COLORS είναι οι μεταβλητές των χρωμάτων. Εκεί ορίζουμε τα χρώματα ως αριθμούς για να μας διευκολύνουν στην εκτέλεση του κώδικα. Μόνο ο χρήστης μπορεί να μεταβάλει την τιμή.
- Η DEFINE MOVES περιέχει τις μεταβλητές των κινήσεων. Η κάθε μεταβλητή ισοδυναμεί με έναν αριθμό, ο οποίος είναι σταθερός και δεν μεταβάλετε κατά την εκτέλεση του προγράμματος. Μόνο ο χρήστης έχει την δυνατότητα να αλλάξει τις τιμές.
- Η STATES DECLARATION περιέχει τις μεταβλητές που αφορούν τις καταστάσεις του προγράμματος. Όλες οι μεταβλητές της δεν αλλάζουν την τιμή τους κατά την διάρκεια του προγράμματος, εκτός των STATE και nextState, οι οποίες μας υποδεικνύουν σε πια κατάσταση του προγράμματος είμαστε και ποια θα είναι η επόμενη αντίστοιχα.
- Η variables περιέχει μεταβλητές, οι οποίες μας βοηθούν κατά την διάρκεια του προγράμματος. Οι μεταβλητές αλλάζουν τις τιμές τους με σκοπό την ορθή λειτουργία του προγράμματος.
- Η GENERAL COUNTERS περιέχει μετρητές, οι οποίοι αυξάνονται και μηδενίζονται κατά την διάρκεια του προγράμματος.
- Η STATIONARY περιέχει μεταβλητές οι οποίες μεταβάλλονται μόνο από τον χρήστη και όχι από το πρόγραμμα.

Συναρτήσεις

Μετά τις μεταβλητές και πριν το κύριο πρόγραμμα υπάρχουν οι συναρτήσεις. Οι συναρτήσεις ξεχωρίζουν λογού ότι ξεκινάνε με τον χαρακτήρα `def` (από την λέξη *definition*) και το όνομα της συνάρτησης. Συνήθως τις τοποθετούνε πριν το κύριο πρόγραμμα και καλούνται με το όνομά τους μέσα σε αυτό. Οι συναρτήσεις που χρησιμοποιώ είναι:

- Η συνάρτηση *btnPressed* ορίζει το ακροδέκτη 40 της GPIO ως είσοδο. Επιστρέφει την κατάσταση του διακόπτη ως `True` ή `False`.
- Η συνάρτηση *main_state_led_blinking* καλείται καθ' όλη τη διάρκεια λήψης στιγμιότυπων από την κάμερα(κατάσταση SD) και έχει ως σκοπό την περιοδική εναλλαγή λειτουργίας του led από ανοιχτό σε κλειστό. Η συνάρτηση επιστρέφει την τιμή της μεταβλητής *Counter_led_on_detect* την τιμή της οποίας και αντιστρέφει σε κάθε κλήση της με σκοπό την εναλλαγή στην κατάσταση του led.
- Η συνάρτηση *GPIO_setup_color* δέχεται την μεταβλητή *Color_Detect* και τις global μεταβλητές *high_color*, *low_color1*, *low_color2*. Η τιμή της *Color_Detect* καθορίζει το χρώμα του υπό ανίχνευση αντικειμένου καθώς και το χρώμα του αντικειμένου που αναζητά το όχημα μας. οι global μεταβλητές *high_color*, *low_color1*, *low_color2* μεταβάλλονται μεταξύ των τιμών 33,35 και 37. Οι τιμές αυτές αποτελούν του ακροδέκτες που έχουμε συνδέσει τους ακροδέκτες του led.
- Η συνάρτηση *activate_current_led_color* δέχεται τις μεταβλητές *high_color*, *low_color1*, *low_color2*. Καθιστά τις τιμές αυτών ως εξόδους της GPIO και θέτει σε κατάσταση `HIGH` την GPIO με αριθμό *high_color*, ενώ θέτει σε `LOW` κατάσταση τις GPIO με αριθμούς *low_color1* και *low_color2*. Η εντολή μας ενεργοποιεί το RGB LED στο χρώμα που έχουμε επιλέξει.

- Η συνάρτηση `draw_circle` καλείται για να απεικονίσει τους κύκλους-αντικείμενα πάνω στο στιγμιότυπο. Δέχεται σαν είσοδο το στιγμιότυπο `c`, τις συντεταγμένες του κέντρου `a1,a2` και την ακτίνα `a3` του κύκλου-αντικειμένου. Η απεικόνιση γίνεται με δύο κύκλους ο ένας απεικονίζει το κέντρο και ο άλλος την περιφέρεια.
- Η συνάρτηση `Flashing` αναβοσβήνει για ένα συγκεκριμένο πλήθος επαναλήψεων το RGB LED. Ενώ όταν φτάνει προς το τέλος ο ρυθμός γίνεται πιο γρήγορος. Δέχεται σαν είσοδο το πλήθος των επαναλήψεων και την μεταβλητή `high_color`.
- Η συνάρτηση `detect_circle_of_object` ελέγχει αν ο κύκλος που έχουμε εντοπίσει είναι το επιθυμητό αντικείμενο. Εισάγεται η εικόνα, οι συντεταγμένες του κέντρου και η ακτίνα του κύκλου. Ελέγχει αν σε μία περιοχή γύρω από το κέντρο του κύκλου επικρατεί ένα χρώμα και αν το χρώμα αυτό είναι αυτό που επιθυμούμε να έχει το αντικείμενο. Όταν πιστοποιηθούν όλα αυτά η συνάρτηση επιστρέφει τις συντεταγμένες του κύκλου, την ακτίνα του και την μεταβλητή `object_detected` σε κατάσταση `True`, αλλιώς μόνο την `object_detected` σε κατάσταση `False`.
- Η συνάρτηση `move` περιέχει όλες τις κινήσεις του αυτοκινήτου, Μπροστά, Πίσω, Δεξιά και Αριστερά. Δέχεται 2 μεταβλητές τον χρόνο που θα κινηθεί και την κίνηση που θα κάνει.
- Η συνάρτηση `moveRL` αποφασίζει αν το όχημα θα κινηθεί Αριστερά ή Δεξιά και υπολογίζει τον χρόνο περιστροφής.
- Η συνάρτηση `dest` υπολογίζει την απόσταση οχήματος-αντικειμένου, τον χρόνο κίνησης του οχήματος και αποφασίζει την κατεύθυνση που θα ακολουθήσει το όχημα

(Μπροστά ή Πίσω). Δέχεται την ακτίνα του κύκλου υπολογίζει την απόσταση, τον χρόνο και την κατεύθυνση.

- Η συνάρτηση `object_not_found` πραγματοποιεί κινήσεις όταν το αντικείμενο δεν έχει εντοπιστεί. Δεν δέχεται σαν είσοδο κάποια μεταβλητή, αλλά χρησιμοποιεί τον μετρητή `next_move_on_fault_detection` για να ελέγχει σε ποια κίνηση βρίσκεται.

Κύριος κώδικας

Το κύριο πρόγραμμα ξεκινάει με την εντολή `with picamera.PiCamera() as camera` η οποία δηλώνει την PiCamera ως την `camera` του προγράμματος. Συνεχίζει βάζοντας το πρόγραμμα σε ένα βρόχο καλώντας την εντολή `while`, ο οποίος επαναλαμβάνεται μέχρι η εντολή `while` γίνει `False`. Ένα σημαντικό βήμα είναι η ενεργοποίηση των GPIO, το οποίο πραγματοποιείται με την εντολή `GPIO.setmode(GPIO.BOARD)`.

Από το `if(STATE==SB):` έως το `elif(STATE==SF):` καταγράφονται όλες οι καταστάσεις του προγράμματος και γίνεται η μεταφορά από την μία κατάσταση στην άλλη. Για τις καταστάσεις `if(STATE==SB):`, `elif(STATE==SG):` και `elif(STATE==SR):` η λειτουργία του κώδικα είναι η ίδια. Ορίζουν το χρώμα του αντικειμένου που επιθυμούμε να εντοπίσουμε στην παρούσας κατάσταση και ενεργοποιούν τις GPIO του RGB LED έτσι ώστε να υποδεικνύεται το χρώμα αυτό. Μετά για ένα χρονικό διάστημα ελέγχουν αν ενεργοποιήθηκε ο μικροδιακόπτης. Αν έχει ενεργοποιηθεί προχωράει στην επόμενη κατάσταση-χρώμα. Αν περάσει το χρονικό διάστημα και δεν έχει ενεργοποιηθεί ο μικροδιακόπτης τότε περνάμε στην κύρια κατάσταση `SD`, με χρώμα επιλογής αυτό που μας υποδεικνύει το RGB LED.

Στην `elif(STATE==SD)` κατάσταση ελέγχεται πάλι ο μικροδιακόπτης. Με το πάτημά του μεταφερόμαστε στην `elif(STATE==SP)` κατάσταση όπου ο αναβοσβήνει το RGB LED και μετά ελέγχει την κατάσταση του μικροδιακόπτη. Αν δεν είναι πατημένος μεταφέρεται στις καταστάσεις `if(STATE==SB):`, `elif(STATE==SG):` και `elif(STATE==SR)`, αν είναι πατημένος στην κατάσταση `elif(STATE==SF)`, όπου καθαρίζει τις GPIO (`GPIO.cleanup()`) και τερματίζει το πρόγραμμα (`quit()`)

Στην κατάσταση `if(STATE==SD)` τρέχει όλο το πρόγραμμα και αποτελεί την κύρια κατάσταση του προγράμματος. Στην αρχή δηλώνεται η ανάκληση της κάμερας (`camera.resolution = (maxX, maxY)`) και μετά από μία γρήγορη ένδειξη της εικόνας (`camera.start_preview()`) γίνεται η λήψη της (`camera.capture('capture.jpg')`). Την εισάγουμε μια έγχρωμη (`img_COLOR= cv2.imread(filename)`) και μια σε αποχρώσεις του γκρι (`img_GRAY = cv2.imread(filename,0)`). Μετά εντοπίζουμε τα κυκλικά αντικείμενα (`detect_circle_of_object(img_GRAY,k[0],k[1],k[2])`) και αν υπάρχουν σχεδιάζουμε τους κύκλους (`draw_circle(cimg_GRAY,k[0],k[1],k[2])`). Γίνεται ο έλεγχος (`detect_circle_of_object(img_COLOR,k[0],k[1],k[2])`) αν εντοπιστεί το επιθυμητό αντικείμενο κινείται προς το μπαλάκι αν είναι στο κέντρο (`dest(k[2])`) αλλιώς στρέφει προς αυτό. Όταν δεν εντοπιστεί το αντικείμενο εκτελεί τις κινήσεις εντοπισμού του αντικειμένου (`object_not_found()`). Στην περίπτωση που δεν εντοπισθούν κύκλοι εκτελείται η εντολή (`fault_image()`). Το RGB LED κάθε φορά γίνεται λήψη στιγμιότυπου μία ανάβει και μια σβήνει (`main_state_led_blinking()`).

Κεφάλαιο 8^ο - Στόχος και βελτίωση της εργασίας

Εισαγωγή

Στο παρόν κεφάλαιο αναφέρομαι αναλυτικά για την επίτευξη του στόχου μου. Κατά πόσο ήταν υλοποιήσιμος και πια τμήματα της παρούσας εργασίας χρίζουν περαιτέρω βελτίωσης.

8.1 Επίτευξη στόχου

Η εργασία είχε ως σκοπό την χρήση μηχανικής όρασης για τις λειτουργίες ενός οχήματος. Το αυτοποιημένο αυτό όχημα κατάφερε με την χρήση της μηχανικής όρασης και συγκεκριμένα της βιβλιοθήκης OpenCV να μειώσει την παρέμβαση του χρήστη στο ελάχιστο. Ο χρήστης παρεμβαίνει στο πρόγραμμα κατά την έναρξή του και στον τερματισμό του προγράμματος. Το αυτοποιημένο όχημα έχει την δυνατότητα μέσω μιας κάμερας και της βιβλιοθήκης OpenCV (σχήμα 8.1) να εντοπίζει το επιθυμητό αντικείμενο με μεγάλη ακρίβεια και να κατευθύνεται προς αυτό χωρίς την παρέμβαση του χρήστη. Μέσο της υλοποίησης ενός πειράματος και με την βοήθεια των αλγορίθμων της OpenCV επιτεύχθηκε η δυνατότητα υπολογισμού της απόστασης χωρίς την χρήση άλλων οργάνων και της δημιουργίας μιας συγκεκριμένης απόστασης μεταξύ οχήματος και αντικειμένου.



Σχήμα 8.1 – Εντοπισμός αντικειμένου

Ο μικροϋπολογιστής Raspberry PI μου έδωσε την δυνατότητα να μην έχω περιορισμούς στις κινήσεις του οχήματος λόγω καλωδίων επικοινωνίας υπολογιστή-οχήματος ή εμβέλειας του ασύρματου δικτύου. Ακόμα δε η τροφοδοσία του με την με Power Bank απελευθέρωσαν πλήρως τις κινήσεις του. Δεν μπορούν να παραληφθούν οι δυνατότητες του Raspberry PI. Ο γρήγορος επεξεργαστής του και η μεγάλη RAM που διαθέτει έπαιξαν καταλυτικό ρόλο για την υλοποίηση της παρούσας εργασίας. Η δυνατότητα του να επεξεργάζεται μεγάλες ποσότητες δεδομένων σε ελάχιστο χρονικό διάστημα μείωσαν τον χρόνο αντίδρασης του αυτοποιημένου οχήματος στο ελάχιστο δυνατό χρονικό διάστημα, ώστε το όχημα να αποκτήσει μια ικανοποιητική λειτουργία.

Όλα τα παραπάνω, καθώς και η αρμονική συνύπαρξη λειτουργικού και μηχανικού μέρους βοήθησαν στην δημιουργία και στην καλή λειτουργία της παρούσας εργασίας και κατά συνέπια στην επίτευξη του στόχου μου κατά ένα μεγάλο βαθμό.

8.2 Βελτιώσεις

Όπως ανάφερα παραπάνω η εργασία έχει υλοποιηθεί κατά ένα μεγάλο βαθμό . Αυτό σημαίνει ότι πρέπει να πραγματοποιηθούν βελτιώσεις τόσο στο λειτουργικό όσο και στο μηχανικό κομμάτι του αυτοποιημένου οχήματος.

Η σημαντικότερη βελτίωση και η πιο εφικτή είναι η πλήρως αυτοματοποίηση του οχήματος. Ο χρήστης μπορεί και παρεμβαίνει στο λειτουργία του αυτοματοποιημένου οχήματος για να υποδείξει τι χρώμα έχει το επιθυμητό αντικείμενο με την χρήση ενός μικροδιακόπτη. Αυτό μπορεί να μετατραπεί με τον κατάλληλο κώδικα και να μην χρειάζεται η επιλογή χρώματος από τον χρήστη, αλλά απλούστατα να του υποδεικνύει το χρώμα του αντικειμένου μέσω της κάμερας. Δηλαδή να του εμφανίζει το αντικείμενο μια φορά στην κάμερα και το όχημα να το αποθήκευε ως το αντικείμενο που πρέπει να εντοπίσει και να ακολουθήσει.

Η αύξηση των δυνατοτήτων του Raspberry PI μπορούν να κάνουν το όχημα ακόμα πιο γρήγορο η και να το μεταφέρουν σε ένα άλλο επίπεδο. Αυτό που θέλω να πω είναι ότι για την λήψη στιγμιότυπων, που πραγματοποιούνται σε αυτήν την εργασία, θα μπορούσαμε

να έχουμε λήψη βίντεο. Αυτό επιβάλλει όμως και την χρήση μίας καλύτερης σε ανάλυση κάμερας.

Ακόμα μία ελαφρά βελτίωση μπορεί να πραγματοποιηθεί στην βιβλιοθήκη OpenCV(σχήμα 8.2-8.3). Οι αλγόριθμοι της μπορούν να βελτιωθούν ως προς την ακρίβεια αλλά και ως προς το σφάλμα. Έτσι θα αποκτήσουν μεγαλύτερη αξιοπιστία και τα περιθώρια σφάλματος θα είναι μηδαμικά.

Περιφερειακές μονάδες του οχήματος όπως η παροχή των κινητήρων αλλά και οι συνεχούς τάσης κινητήρες μπορούν να βελτιωθούν. Μεγαλύτερης ισχύς κινητήρες και καλύτερης ποιότητας παροχή θα βοηθούσαν το όχημα να κινείται γρηγορότερα και για μεγαλύτερο χρονικό διάστημα.



Σχήμα 8.2-Εντοπισμος παραπάνω κύκλων



Σχήμα 8.3- Εντοπισμός μεγαλύτερου κύκλου

BIBΛΙΟΓΡΑΦΙΑ

- [1] Reinhard Klette (2014). *Concise Computer Vision*. Springer. ISBN 978-1-4471-6320-6.
- [2] Linda G. Shapiro and George C. Stockman (2001). *Computer Vision*. Prentice Hall. ISBN 0-13-030796-3.
- [3] Tim Morris (2004). *Computer Vision and Image Processing*. Palgrave Macmillan. ISBN 0-333-99451-5.
- [4] Bernd Jähne and Horst Haußecker (2000). *Computer Vision and Applications, A Guide for Students and Practitioners*. Academic Press. ISBN 0-13-085198-1.
- [5] Milan Sonka, Vaclav Hlavac and Roger Boyle (2008). *Image Processing, Analysis, and Machine Vision*. Thomson. ISBN 0-495-08252-X.
- [6] David A. Forsyth and Jean Ponce (2003). *Computer Vision, A Modern Approach*. Prentice Hall. ISBN 0-13-085198-1.
- [7] C. Papageorgiou and T. Poggio, “A Trainable Pedestrian Detection system”, *International Journal of Computer Vision (IJCV)*, page 1:15-33,2000
- [8] <http://vislab.it/proud-en/>
- [9] <https://cyberpython.github.io/byte-of-python/index.html>
- [10] <http://dide.flo.sch.gr/Plinet/Tutorials/Tutorials-Python-Introduction.html>
- [11] <https://www.tutorialspoint.com/python/>
- [12] <https://www.raspberrypi.org/>
- [13] <https://www.sparkfun.com/products/12089>
- [14] Navneet Dalal and Bill Triggs. Histogram of Oriented Gradients for Human Detection.

- [15]] Constantine Papageorgiou and Tomaso Poggio. A Trainable System for Object Detection.
- [16] Piotr Porwik, Agnieszka Lisowska The Haar–Wavelet Transform in Digital Image Processing: Its Status and Achievements
- [17] Henry Schneiderman and Takeo Kanade. Object Detection Using the Statistics of Parts.
- [18] Paul Viola, Michael J. Jones and Danie Snow. Detecting Pedestrians Using Patterns of Motion and Appearance.
- [19] Shengcai Liao, Xiangxin Zhu, Zhen Lei, Lun Zhang and Stan Z. Li. *Learning Multi-scale Block Local Binary Patterns for Face Recognition*. International Conference on Biometrics (ICB), 2007, pp. 828-837.
- [20] Charles A. Poynton (2003). *Digital Video and HDTV: Algorithms and Interfaces*. Morgan Kaufmann. pp. 260, 630.
- [21] Ojala, T., Pietikäinen, M. and Harwood, D. (1996), A Comparative Study of Texture Measures with Classification Based on Feature Distributions. Pattern Recognition 19(3):51-59.
- [22] Ojala, T. and Pietikäinen, M. (1999), Unsupervised Texture Segmentation Using Feature Distributions. Pattern Recognition 32:477-486.
- [23] Ojala, T., Pietikäinen, M. and Mäenpää, T. (2002), Multiresolution Gray-scale and Rotation Invariant Texture Classification with Local Binary Patterns. IEEE Trans. Pattern Analysis and Machine Intelligence 24(7): 971-987.
- [24] T. Ojala, M. Pietikäinen, and D. Harwood (1996), "A Comparative Study of Texture Measures with Classification Based on Feature Distributions", Pattern Recognition, vol. 29, pp. 51-59.

- [25] Ahonen, T., Hadid, A. and Pietikäinen, M. (2006), Face Description with Local Binary Patterns: Application to Face Recognition. *IEEE Trans. Pattern Analysis and Machine Intelligence* 28(12):2037-2041.
- [26] Zhao, G. and Pietikäinen, M. (2007), Dynamic Texture Recognition Using Local Binary Patterns with an Application to Facial Expressions. *IEEE Trans. Pattern Analysis and Machine Intelligence* 29(6):915-928.
- [27] Li, S.Z., Chu, R.F., Liao, S.C. and Zhang, L.(2007), Illumination Invariant Face Recognition Using Near-Infrared Images. *IEEE Trans. Pattern Analysis and Machine Intelligence* 29(4):627-639.
- [28] Zhang, W., Shan, S., Gao, W., Chen, X. and Zhang H. (2005), Local Gabor Binary Pattern Histogram Sequence (LGBPHS): A Novel Non-statistical Model for Face Representation and Recognition. In *Proc. Tenth IEEE International Conference on Computer Vision*, 1:786-791.
- [29] Hadid, A. and Pietikäinen, M. (2009), Combining Appearance and Motion for Face and Gender Recognition from Videos. *Pattern Recognition* 42(11):2818-2827.
- [30] Zhao, G., Barnard, M. and Pietikäinen, M. (2009), Lipreading with Local Spatiotemporal Descriptors. *IEEE Trans. Multimedia* 11(7):1254-1265.
- [31] Pietikäinen, M., Hadid, A., Zhao, G. and Ahonen, T. (2011), *Computer Vision Using Local Binary Patterns*, Springer, 207 p
- [32] Li, S.Z., Chu, R.F., Liao, S.C. and Zhang, L.(2007), Illumination Invariant Face Recognition Using Near-Infrared Images. *IEEE Trans. Pattern Analysis and Machine Intelligence* 29(4):627-639.
- [33] Liao, S., Law M.W.K. and Chung A.C.S. (2009), Dominant Local Binary Patterns for Texture Classification. *IEEE Trans. Image Processing* 18(5):1107-1118.

- [34] Lowe, David G. (1999). Object recognition from local scale-invariant features. *Proc. 7th International Conference on Computer Vision (ICCV'99)* (Corfu, Greece): 1150-1157. doi:[10.1109/ICCV.1999.790410](https://doi.org/10.1109/ICCV.1999.790410).
- [35] Lowe, David G. (2004). Distinctive image features from scale-invariant key points. *International Journal of Computer Vision* 60(2): 91-110.
- [36] Heikkilä, M., Pietikäinen, M. and Schmid, C. (2009), Description of Interest Regions with Local Binary Patterns. *Pattern Recognition* 42(3):425-436.
- [37] Mäenpää, T. and Pietikäinen, M. (2004), Classification with Color and Texture: Jointly or Separately? *Pattern Recognition* 37(8):1629-1640.
- [38] Tan X. and Triggs B. (2007), Fusing Gabor and LBP feature Sets for Kernel-based Face Recognition, *Proceedings of the IEEE International Workshop on Analysis and Modeling of Face and Gesture*, pp.235-249.
- [39] Wang J.-G., Yau W.-Y. and Wang H. L. (2009), Age Categorization via ECOC with Fused Gabor and LBP Features. *Proceedings of the IEEE Workshop on Applications of Computer Vision (WACV)*, pp.313-318.
- [40] Heikkilä, M. and Pietikäinen, M. (2006), A Texture-Based Method for Modeling the Background and Detecting Moving Objects. *IEEE Trans. Pattern Analysis and Machine Intelligence* 28(4):657-662.
- [41] Heusch G., Rodriguez Y. and Marcel S. (2006), Local Binary Patterns as an Image Processing for Face Authentication, *Proceedings of the IEEE International Conference on Automatic Face and Gesture Recognition*.
- [42] Lowe, David G. (2004). Distinctive image features from scale-invariant key points. *International Journal of Computer Vision* 60(2): 91-110.
- [43] Lienhart, R. and Maydt, J. An extended set of Haar-like features for rapid object

detection. IEEE ICIP 2002, Vol. 1, pp. 900-903, Sep. 2002.

[44] Open Computer Vision Library Reference Manual. Intel Corporation, USA, 2001.

[45] Viola, P. and Jones, M. Rapid object detection using boosted cascade of simple features. IEEE Conference on Computer Vision and Pattern Recognition, 2001.

[46] Viola, P. and Jones, M. Rapid object detection using boosted cascade of simple features. IEEE Conference on Computer Vision and Pattern Recognition, 2001.

[47] Wilson, P and Fernandez, J. Establishing a face recognition research environment using open source software. ASEE Gulf-Southwest Annual Conference, March, 2005.

[48] Menezes, P., Barreto, J.C. and Dias, J. Face tracking based on Haar-like features and eigenfaces. 5th IFAC Symposium on Intelligent Autonomous Vehicles, Lisbon, Portugal, July 5-7, 2004.

[49] Howell, Kenneth B. (2001). *Principles of Fourier Analysis*, CRC Press. ISBN 978-0-8493-8275-8

[50] Haar, Alfréd (1910), "Zur Theorie der orthogonalen Funktionensysteme", *Mathematische Annalen* page 361

[51] Lee, B.; Tarn, Y. S. (1999). "Application of the discrete wavelet transform to the monitoring of tool failure in end milling using the spindle motor current". *International Journal of Advanced Manufacturing Technology* 15 (4): 238–243. doi:10.1007/s001700050062.

[52] Tom M. Mitchell (1997). *Machine Learning*

[53] <https://hackaday.io/projects>

[54] https://en.wikipedia.org/wiki/Raspberry_Pi_Foundation

- [55] <https://www.raspberrypi.org/blog/raspbian-jessie-is-here/>
- [56] <https://el.wikipedia.org/wiki/Debian>
- [57] https://en.wikipedia.org/wiki/Guido_van_Rossum
- [58] <https://cyberpython.github.io/byte-of-python/basics.html>
- [59] <http://www.pyimagesearch.com/2015/10/26/how-to-install-opencv-3-on-raspbian-jessie/>

ΚΩΔΙΚΑΣ

```
import sys

sys.path.append('/usr/local/lib/python2.7/site-packages')

import time

Import picamera

import RPi.GPIO as GPIO

from time import sleep

import numpy as np

import cv2

from sys import exit


===DEFINE COLORS=====

Timer=0

BLUE=0

GREEN=1

RED=2


===DEFINE MOVES=====

STOP=0

RIGHT=1

LEFT=2

FORWARD=3

REVERSE=4


===STATES DECLARATION=====

SB=0

SG=1

SR=2

SD=3

SF=4

SP=5

nextState=SB

STATE=SB #INITIAL STATE
```

```

#=====variables=====

Pressed=0

waitkey=False

#testRotation=False

high_color=0

low_color1=0

low_color2=0

x_center=0

y_center=0

rR=0

object_detected=False

Color_Detect=BLUE

#=====GENERAL COUNTERS=====

next_move_on_fault_detection=0

Counter_led_on_detect=0#metritis

no_detection_counter=0

#=====STATIONARY=====

maxX=320

maxY=240

time_to_turn=0.25

StartIn=20 #start_in sec

#####

def btnPressed():

    inp=40

    GPIO.setup(inp,GPIO.IN)

    return GPIO.input(inp)

def main_state_led_blinking():

    global Counter_led_on_detect, high_color

    Counter_led_on_detect=Counter_led_on_detect+1

    GPIO.setup(high_color,GPIO.OUT)

    if(Counter_led_on_detect==1):

        GPIO.output(high_color,GPIO.HIGH)

    else:

        GPIO.output(high_color,GPIO.LOW)

        Counter_led_on_detect=0

    return Counter_led_on_detect

```

```

def GPIO_setup_color(Color_Detect):

    global high_color, low_color1, low_color2

    high_color=33+2*((Color_Detect+1)%3)

    low_color1=33+2*((Color_Detect+2)%3)

    low_color2=33+2*((Color_Detect+0)%3)

    return (high_color, low_color1, low_color2)


def activate_current_led_color(high_color, low_color1, low_color2):

    GPIO.setup(high_color,GPIO.OUT)

    GPIO.setup(low_color1,GPIO.OUT)

    GPIO.setup(low_color2,GPIO.OUT)


    GPIO.output(high_color,GPIO.HIGH)

    GPIO.output(low_color1,GPIO.LOW)

    GPIO.output(low_color2,GPIO.LOW)

    return;


def draw_circle(c,a1,a2,a3): #k[0],k[1],k[2]:

    cv2.circle(c,(a1,a2),a3,(0,255,0),2)

    cv2.circle(c,(a1,a2),2,(0,0,255),3)

    return;


def Flashing(m, high_color):

    GPIO.setup(high_color,GPIO.OUT)

    for i in range (0,m):

        dc=100-(80/m)*i

        ON=0.01*dc

        Counter_bad_imgF=0.01*(100-dc)

        GPIO.output(high_color,GPIO.LOW)

        sleep(Counter_bad_imgF)          #Counter_bad_imgF

        GPIO.output(high_color,GPIO.HIGH)

        sleep(ON)    #ON

    return;

```

```

def detect_circle_of_object(t,x,y,radius):

    i=0

    j=0

    Tr=0

    Fl=0

    global object_detected,x_center,y_center,rR,no_detection_counter,Color_Detect

    C0= t.item(y,x,(0+Color_Detect)%3)

    C1=t.item(y,x,(1+Color_Detect)%3)

    C2=t.item(y,x,(2+Color_Detect)%3)

    print('C0=',C0,' C1=',C1,' C2=',C2);

    if Color_Detect==BLUE:

if radius > 20:

    for i in range(int(x-(radius/3)),int(x+(radius/3))):

        for j in range(int(y-(radius/3)),int(y+(radius/3))):

            if (C0-15)>C1 and (C0-15)>C2:

                Tr=Tr+1

            else:

                Fl=Fl+1

else:

    for i in range(int(x-4),int(x+4)):

        for j in range(int(y-4),int(y+4)):

            if (C0-15)>C1 and (C0-15)>C2:

                Tr=Tr+1

            else:

                Fl=Fl+1

elif Color_Detect==GREEN:

if radius > 20:

    for i in range(int(x-(radius/3)),int(x+(radius/3))):

        for j in range(int(y-(radius/3)),int(y+(radius/3))):

            if (C0-15)>C1 and (C0+8)>C2:

                Tr=Tr+1

            else:

                Fl=Fl+1

else:

    for i in range(int(x-4),int(x+4)):

        for j in range(int(y-4),int(y+4)):

            if (C0-15)>C1 and (C0+8)>C2:

                Tr=Tr+1

```

```

        else:

            Fl=Fl+1

elif Color_Detect==RED:

    if radius > 20:

        for i in range(int(x-(radius/3)),int(x+(radius/3))):

            for j in range(int(y-(radius/3)),int(y+(radius/3))):

                if (C0-60)>C1 and (C0-60)>C2:

                    Tr=Tr+1

                else:

                    Fl=Fl+1

    else:

        for i in range(int(x-4),int(x+4)):

            for j in range(int(y-4),int(y+4)):

                if (C0-60)>C1 and (C0-60)>C2:

                    Tr=Tr+1

                else:

                    Fl=Fl+1

    if Tr>Fl and y<200:

        print('*****BALL DETECTED*****')

        object_detected=True

        x_center=x

        y_center=y

        rR=radius

        no_detection_counter=0

        return object_detected,x_center,y_center,rR,no_detection_counter;

    else:

        no_detection_counter=no_detection_counter+1

        if no_detection_counter>2:

            object_detected=False

            no_detection_counter=0

        return object_detected,no_detection_counter;

def move(t,d):

    Motor1A = 16

    Motor1B = 18

    Motor1E = 22

    Motor2A=19

    Motor2B=21

    Motor2E=23

```

```

GPIO.setup(Motor1A,GPIO.OUT)

GPIO.setup(Motor1B,GPIO.OUT)

GPIO.setup(Motor1E,GPIO.OUT)

GPIO.setup(Motor2A,GPIO.OUT)

GPIO.setup(Motor2B,GPIO.OUT)

GPIO.setup(Motor2E,GPIO.OUT)


GPIO.output(Motor1A,GPIO.LOW)

GPIO.output(Motor1B,GPIO.LOW)

GPIO.output(Motor2A,GPIO.LOW)

GPIO.output(Motor2B,GPIO.LOW)


GPIO.output(Motor1E,GPIO.HIGH)

GPIO.output(Motor2E,GPIO.HIGH)


if d==RIGHT:

    GPIO.output(Motor1A,GPIO.HIGH)

    GPIO.output(Motor2A,GPIO.HIGH)


elif d==LEFT:

    GPIO.output(Motor1B,GPIO.HIGH)

    GPIO.output(Motor2B,GPIO.HIGH)


elif d==FORWARD:

    GPIO.output(Motor1B,GPIO.HIGH)

    GPIO.output(Motor2A,GPIO.HIGH)


elif d==REVERSE:

    GPIO.output(Motor1A,GPIO.HIGH)

    GPIO.output(Motor2B,GPIO.HIGH)

sleep(t)


GPIO.output(Motor1E,GPIO.LOW)

GPIO.output(Motor2E,GPIO.LOW)

#-----

def moveRL( x ):

    global maxX,object_detected

    if x<maxX/2-10:

```

```

        move(((maxX/2-x)/maxX)/2,1)

    elif x>maxX/2+10:

        move(((x-maxX/2)/maxX)/2,2)

    return;

#-----

def dest(x): #mpros piso

    global object_detected

    l=880.18/x

    if l < 30 :

        time_to_move=(33-l)/33

    if time_to_move < 1.5:

        move(time_to_move,REVERSE)

    elif l > 36:

        time_to_move=(l-33)/l

        if time_to_move<1.5:

            move(time_to_move,FORWARD)

    return;

#_____

def object_not_found():

    Counter_bad_img=0

    global next_move_on_fault_detection

    if (next_move_on_fault_detection==0 or next_move_on_fault_detection==1):

        move(time_to_turn,RIGHT)

        print('->')

    elif next_move_on_fault_detection==2:

        move(time_to_turn*2,LEFT)

        print('<- <-')

    elif next_move_on_fault_detection==3 or next_move_on_fault_detection==4:

        move(time_to_turn,LEFT)

        print('<-')

    elif next_move_on_fault_detection==5:

        move(time_to_turn*2,RIGHT)

        print('-> ->')

    elif next_move_on_fault_detection==6:

        move(time_to_turn*4,RIGHT)

```

```

        print('-> -> -> ->')

    else:

        next_move_on_fault_detection=0

    next_move_on_fault_detection=next_move_on_fault_detection+1

    return next_move_on_fault_detection;

```

```

with picamera.PiCamera() as camera:

    newC=True

    while True:

        GPIO.setmode(GPIO.BOARD)

        if(STATE==SB):

            Color_Detect=BLUE # 0,1,2

            GPIO_setup_color(Color_Detect)

            activate_current_led_color(high_color, low_color1, low_color2)

            if(btnPressed()==Pressed): #if button pressed #and SelectColor) :

                if(new_Cmd==True):

                    new_Cmd=False

                    Timer=0

                    nextState=SG

                else:

                    Timer=Timer+1

                    sleep(0.25)

                    new_Cmd=True

                    Timer=Timer+1

                    if(Timer>StartIn):

                        Flashing(1, high_color)

                        nextState=SD

        elif(STATE==SG):

            Color_Detect=GREEN # 0,1,2

            GPIO_setup_color(Color_Detect)

            activate_current_led_color(high_color, low_color1, low_color2)

            if(btnPressed()==Pressed): #if button pressed #and SelectColor) :

                if(new_Cmd==True):

                    new_Cmd=False

                    Timer=0

```

```

        nextState=SR

    else:

        Timer=Timer+1

        sleep(0.25)

        GPIO_setup_color(Color_Detect)

        activate_current_led_color(high_color, low_color1, low_color2)

        new_Cmd=True

        Timer=Timer+1

        if(Timer>StartIn):

            Flashing(1, high_color)

            nextState=SD

elif(STATE==SR):

    Color_Detect=RED # 0,1,2

    GPIO_setup_color(Color_Detect)

    activate_current_led_color(high_color, low_color1, low_color2)

    if(btnPressed()==Pressed): #if button pressed #and SelectColor) :

        if(new_Cmd==True):

            new_Cmd=False

            Timer=0

            nextState=SB

    else:

        Timer=Timer+1

        sleep(0.25)

        GPIO_setup_color(Color_Detect)

        activate_current_led_color(high_color, low_color1, low_color2)

        new_Cmd=True

        Timer=Timer+1

        if(Timer>StartIn):

            Flashing(1, high_color)

            Timer=0

            nextState=SD

elif(STATE==SD):

    if(btnPressed()==Pressed): #if button pressed #and SelectColor) :

        if(new_Cmd==True):

            new_Cmd=False

            Timer=0

            nextState=SP

    else:

        new_Cmd=True

elif(STATE==SP):

```

```

        Flashing(5, high_color)

    if(btnPressed()==Pressed):

        Timer=0

        nextState=SF

    else:

        Timer=0

        nextState=SB

elif(STATE==SF):

    GPIO.cleanup()

quit()      #end Counter_bad_img program

```

#=====

```

if(STATE==SD):

    main_state_led_blinking()

    camera.resolution = (maxX, maxY)

    camera.start_preview()

    time.sleep(0.002) # Camera warm-up time

    camera.stop_preview()

    camera.capture('capture.jpg')

    filename='capture.jpg'

    img_COLOR= cv2.imread(filename)

    img_GRAY = cv2.imread(filename,0)

    cimg_GRAY = cv2.medianBlur(img_GRAY,5)

    circles = cv2.HoughCircles(cimg_GRAY,
cv2.HOUGH_GRADIENT,1,100,param1=175,param2=15,minRadius=3,maxRadius=95)

    if circles is not None:

        for k in circles[0, :]:

            draw_circle(cimg_GRAY,k[0],k[1],k[2])

            detect_circle_of_object(img_COLOR,k[0],k[1],k[2]);

            if object_detected:

                break

        if object_detected:

            next_move_on_fault_detection=0

```

```

        if x_center>maxX/2-30 and x_center<maxX/2+30:

            dest(rR)

        else : #anazitoyme to mpalaki

            moveRL(x_center)

    else:

        print('next_move_on_fault_detection=',next_move_on_fault_detection)

        object_not_found()

else:

    if object_detected==False:

        object_not_found()

    else:

        no_detection_counter=no_detection_counter+1

        if no_detection_counter>2:

            no_detection_counter=0

            object_detected=False

cv2.imshow('mpalaki',cimg_GRAY)

if waitkey==True:

    cv2.waitKey()

else:

    cv2.waitKey(100)

    cv2.destroyAllWindows()

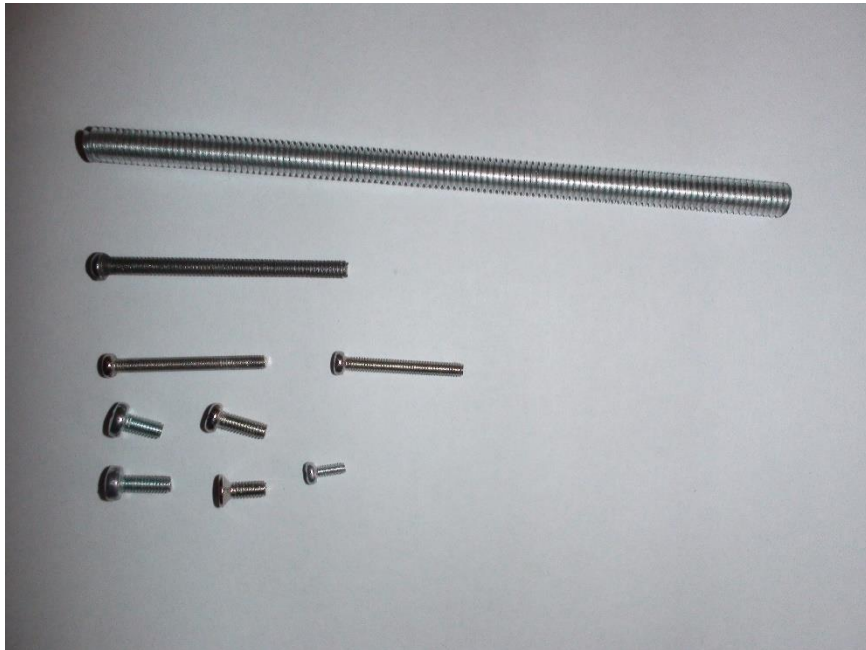
GPIO.cleanup()

STATE=nextState#change the state at next iteration

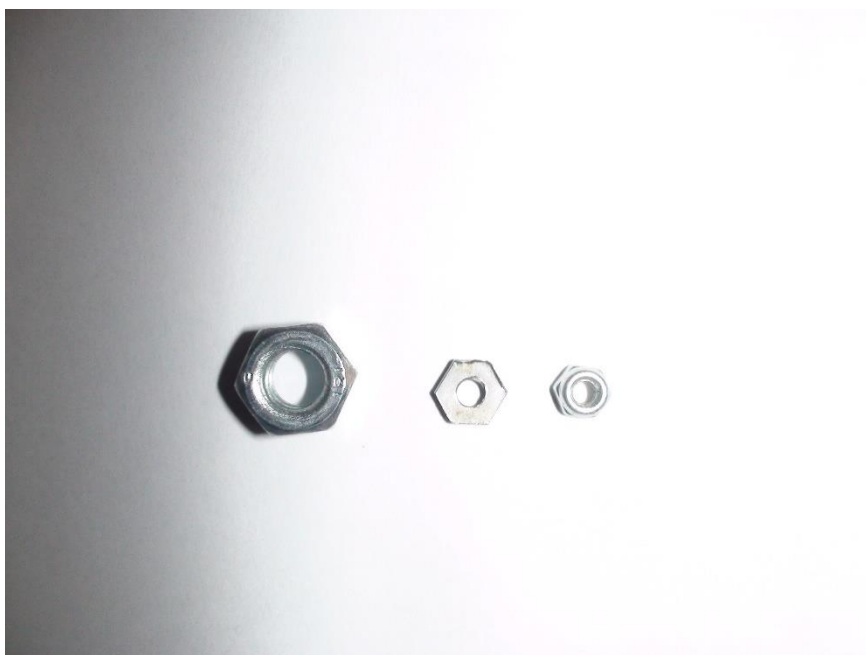
#####

```

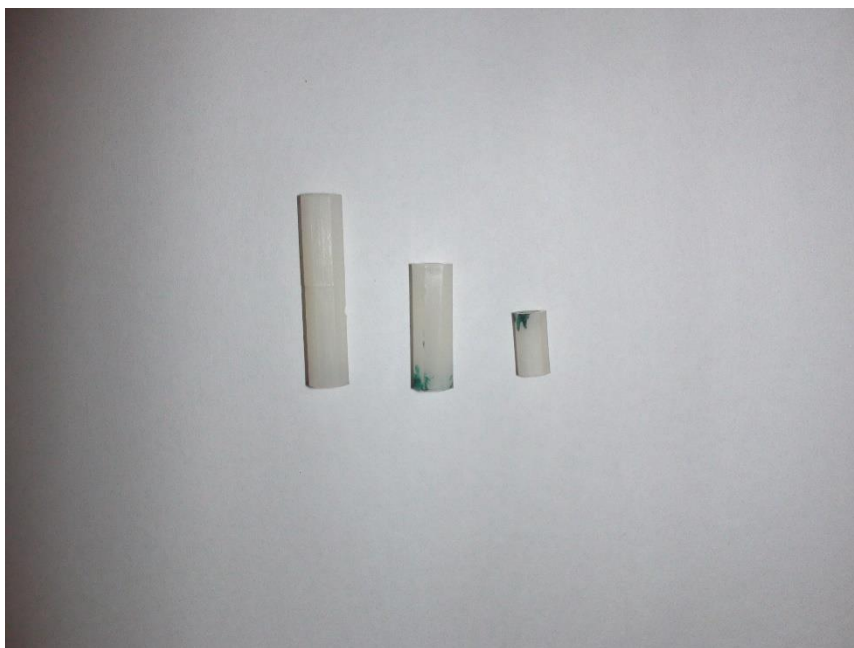
Παράρτημα



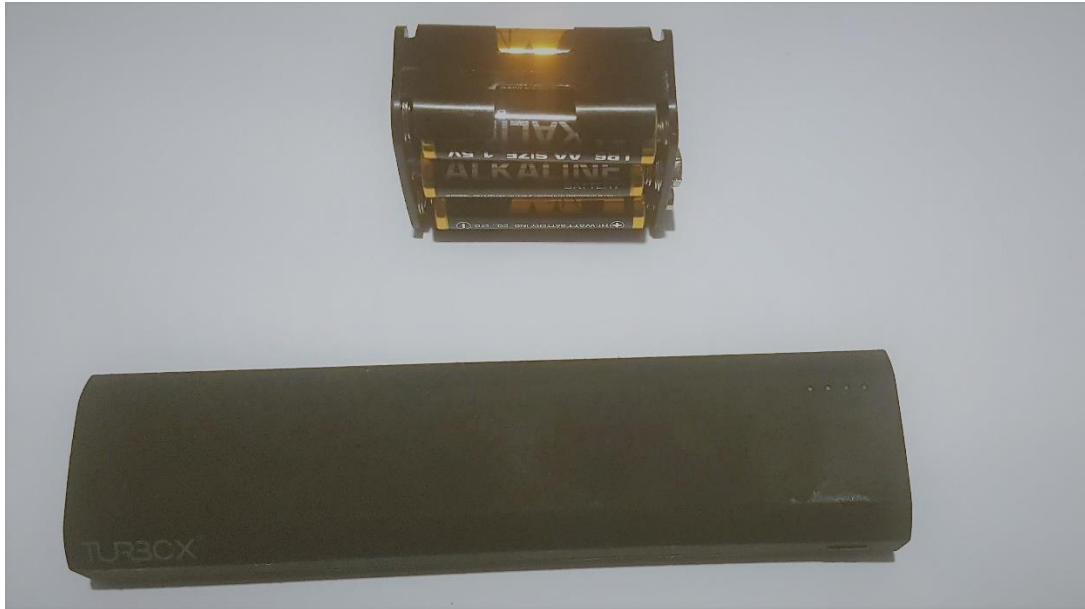
Σχήμα 1 - Ντίζες στήριξης, βίδες διαφόρων μεγεθών για στήριξη



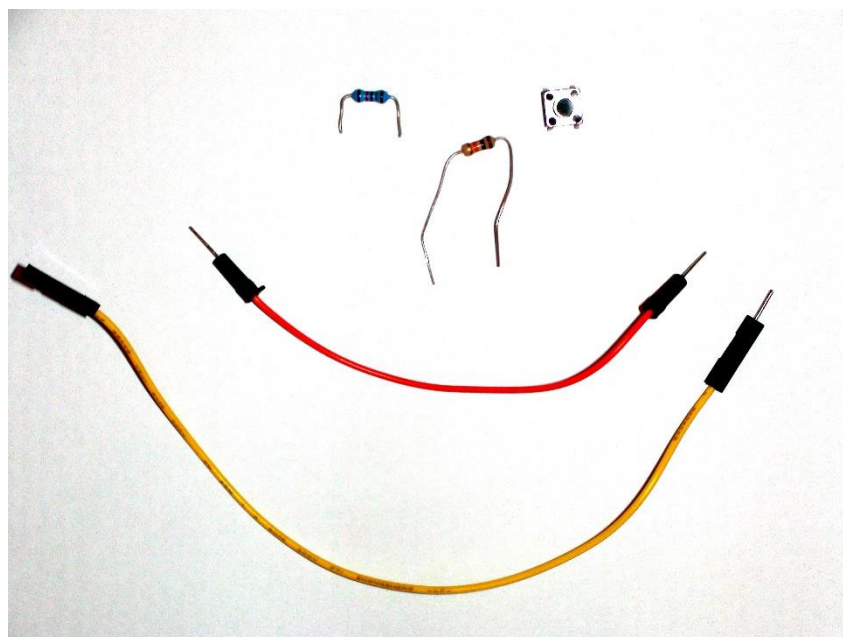
Σχήμα 2 - Μεταλλικά παξιμάδια



Σχήμα 3 - Πλαστικοί αποστάτες

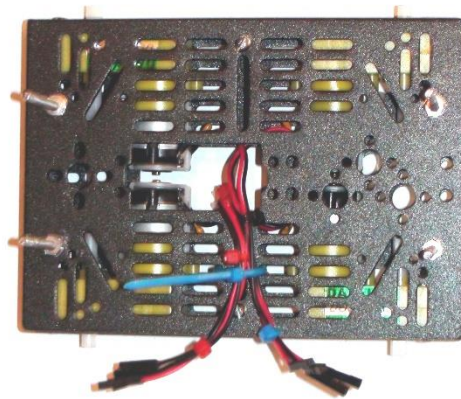


Σχήμα 4 – Πάνω: Power Bank, Κάτω: Μπαταρία 9Volts

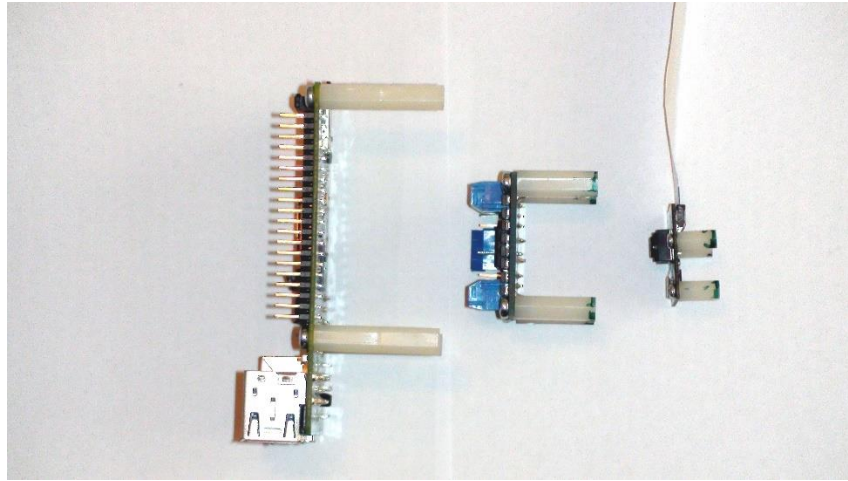


Σχήμα 5- Πάνω Αριστερά: 2 Αντιστάσεις Πάνω Δεξιά: Button

Κάτω: καλώδια



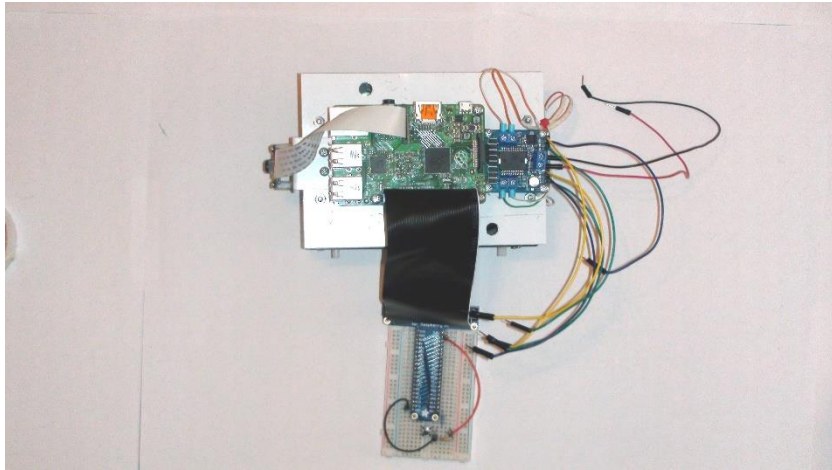
Σχήμα 6 – Το σασί με το καπάκι του και τις βίδες-αποστάτες



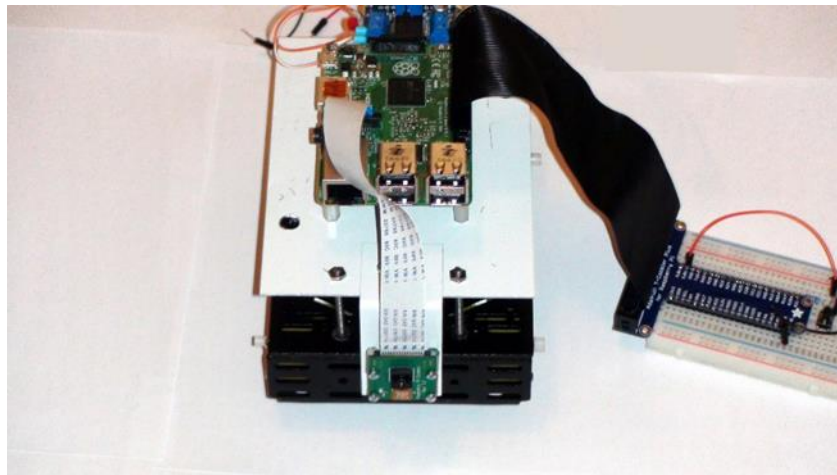
Σχήμα 7 – Από τα αριστερά στα δεξιά: 1. Raspberry Pi 2, 2. Κύκλωμα οδήγησης κινητήρων, 3. Raspberry Pi Camera Board



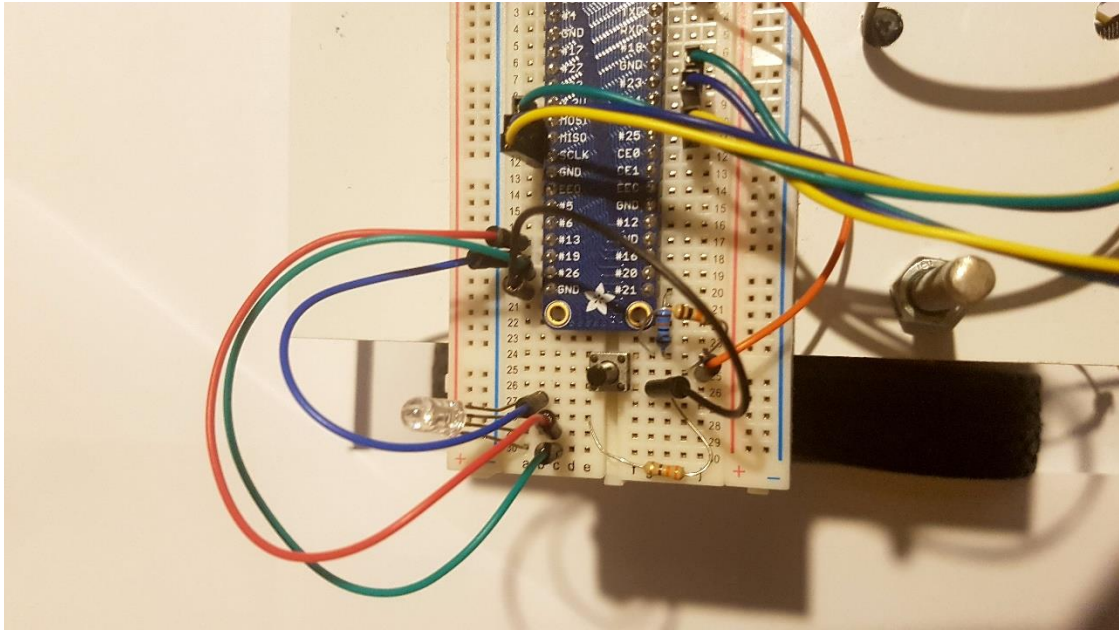
Σχέδιο 8 – GPIO T-cobbler με το breadboard



*Σχήμα 9 – Σύνδεση όλων των εξαρτημάτων με το Raspberry Pi 2
και τοποθέτηση τους πάνω στο σασί, από πάνω.*



*Σχήμα 10 – Σύνδεση όλων των εξαρτημάτων με το Raspberry Pi 2
και τοποθέτηση τους πάνω στο σασί, από μπροστά.*



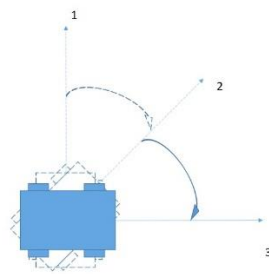
*Σχήμα 11 - Σύνδεση του κυκλώματος οδήγησης κινητήρων,
το κύκλωμα Pull Up Resistor και το RGB LED.*

```
#!/bin/bash  
  
echo "Running script"  
  
source ~/.profile  
  
workon cv  
  
echo "Starting the program"  
  
sudo python vehicle.py
```

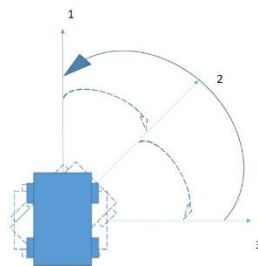
Σχήμα 12 – script startscript.sh



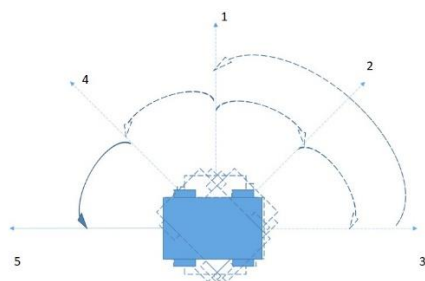
Σχήμα 13- Κινήσεις οχήματος για εντοπισμό μπάλας



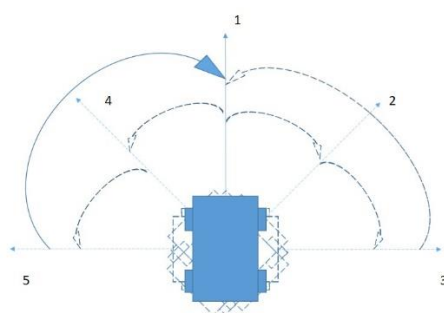
Σχήμα 14- Κινήσεις οχήματος για εντοπισμό μπάλας



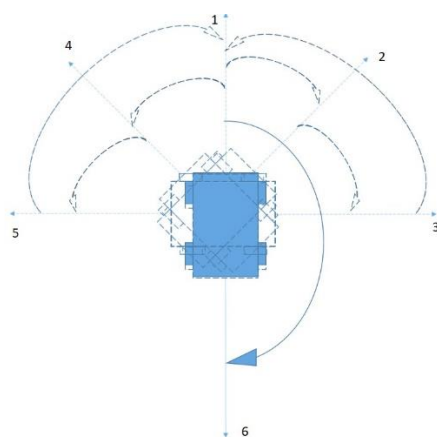
Σχήμα 15- Κινήσεις οχήματος για εντοπισμό μπάλας



Σχήμα 16- - Κινήσεις οχήματος για εντοπισμό μπάλας



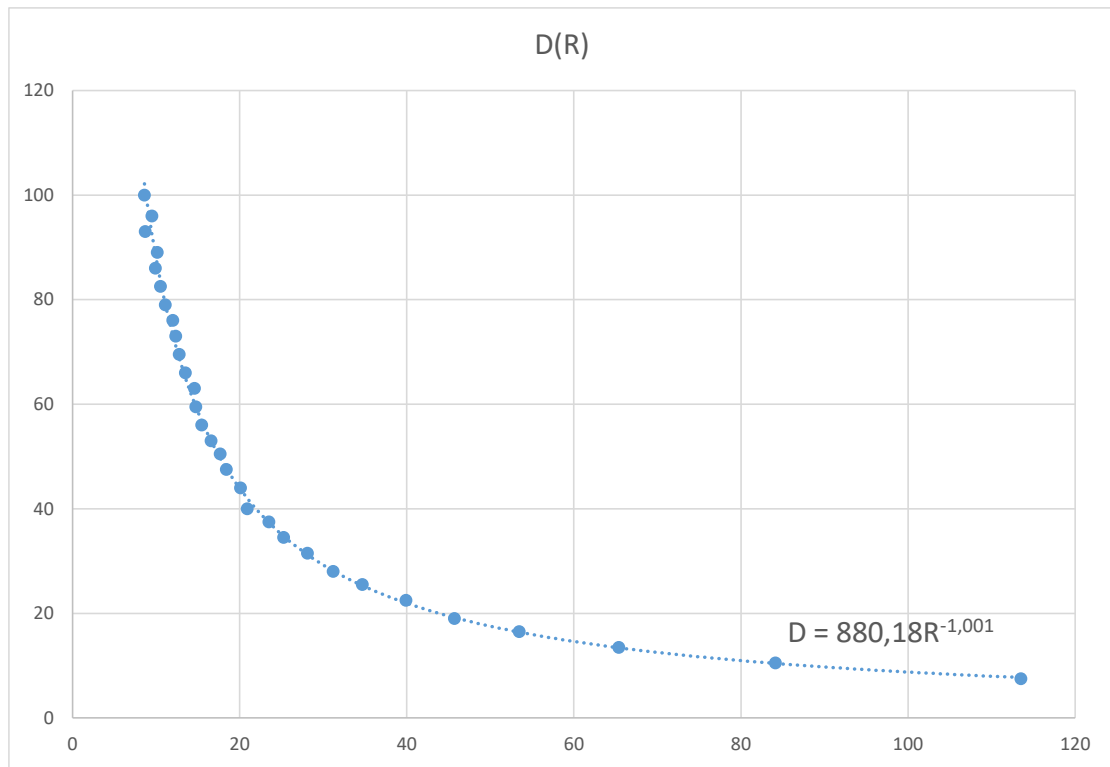
Σχήμα 17- Κινήσεις οχήματος για εντοπισμό μπάλας



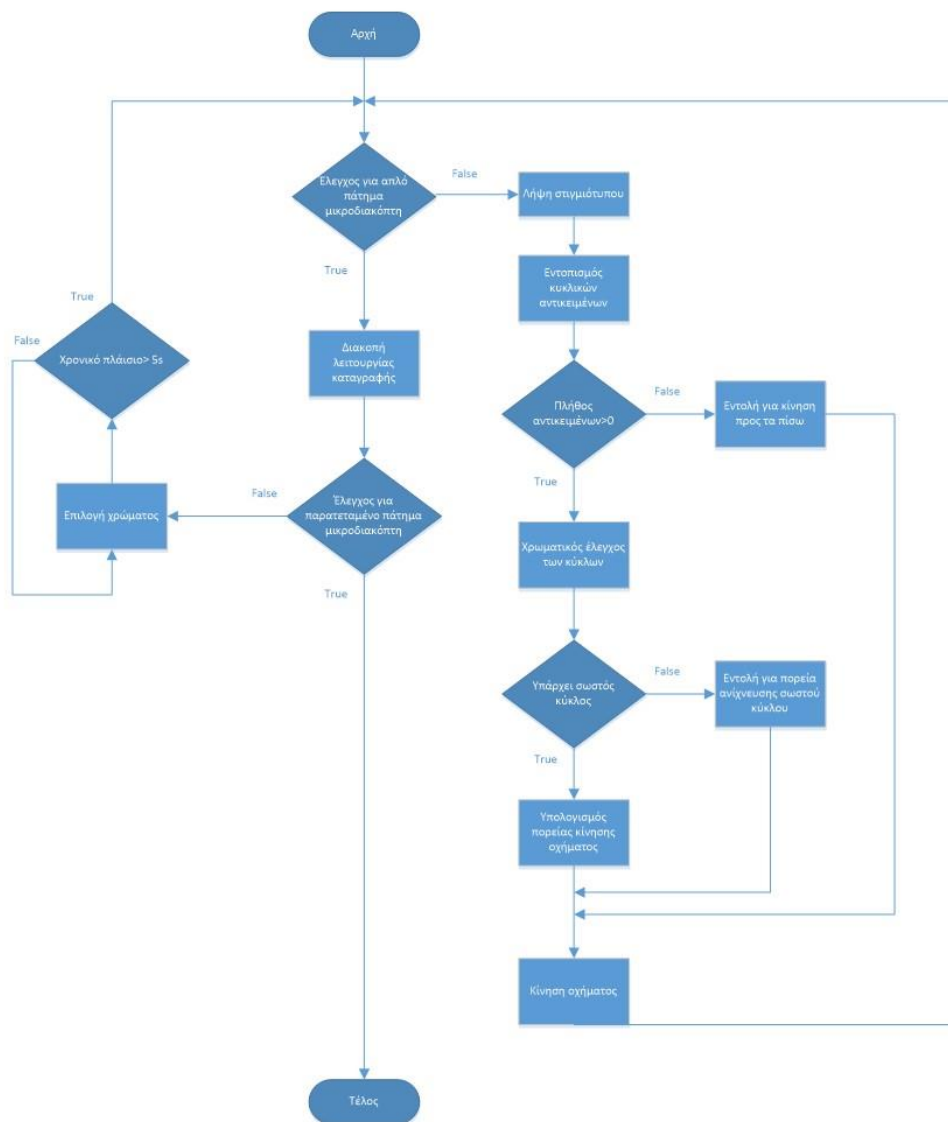
Σχήμα 18- Κινήσεις οχήματος για εντοπισμό μπάλας

α/α	D (cm)	R (cm)	mg 'focolor	heoritical valu	error
1	100	8,6	0	102,13	2,13%
2	96	9,5	1	92,44	3,71%
3	93	8,7	2	100,95	8,55%
4	89	10,12	3	86,77	2,50%
5	86	9,9	4	88,70	3,14%
6	82,5	10,51	5	83,55	1,27%
7	79	11,1	6	79,10	0,13%
8	76	12	7	73,17	3,73%
9	73	12,34	8	71,15	2,54%
10	69,5	12,74	9	68,91	0,85%
11	66	13,5	10	65,03	1,47%
12	63	14,57	12	60,25	4,37%
13	59,5	14,75	13	59,51	0,02%
14	56	15,44	14	56,85	1,52%
15	53	16,58	15	52,94	0,12%
16	50,5	17,67	16	49,67	1,65%
17	47,5	18,39	17	47,72	0,47%
18	44	20,1	18	43,66	0,78%
19	40	20,9	19	41,99	4,97%
20	37,5	23,5	20	37,34	0,44%
21	34,5	25,26	21	34,73	0,67%
22	31,5	28,1	22	31,22	0,89%
23	28	31,18	23	28,13	0,47%
24	25,5	34,67	24	25,30	0,79%
25	22,5	39,91	25	21,97	2,34%
26	19	45,7	26	19,19	0,98%
27	16,5	53,44	27	16,41	0,58%
28	13,5	65,36	28	13,41	0,66%
29	10,5	84,1	29	10,42	0,77%
30	7,5	113,5	30	7,72	2,91%
					1,85%

Σχήμα 19- Πίνακας μετρήσεων απόστασης ακτίνας



Σχήμα 20- Γραφική εξίσωση απόστασης-ακτίνας



Σχήμα 21- Διάγραμμα ροής του κώδικα