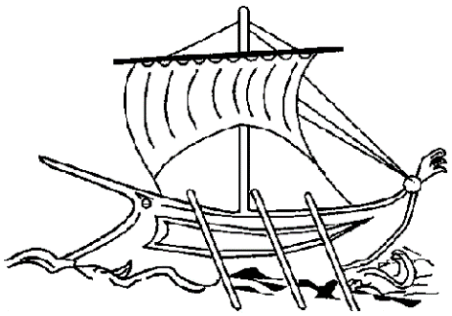


ΑΕΙ ΠΕΙΡΑΙΑ Τ.Τ.

ΤΜΗΜΑ

ΜΗΧΑΝΙΚΩΝ ΗΛΕΚΤΡΟΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΙΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

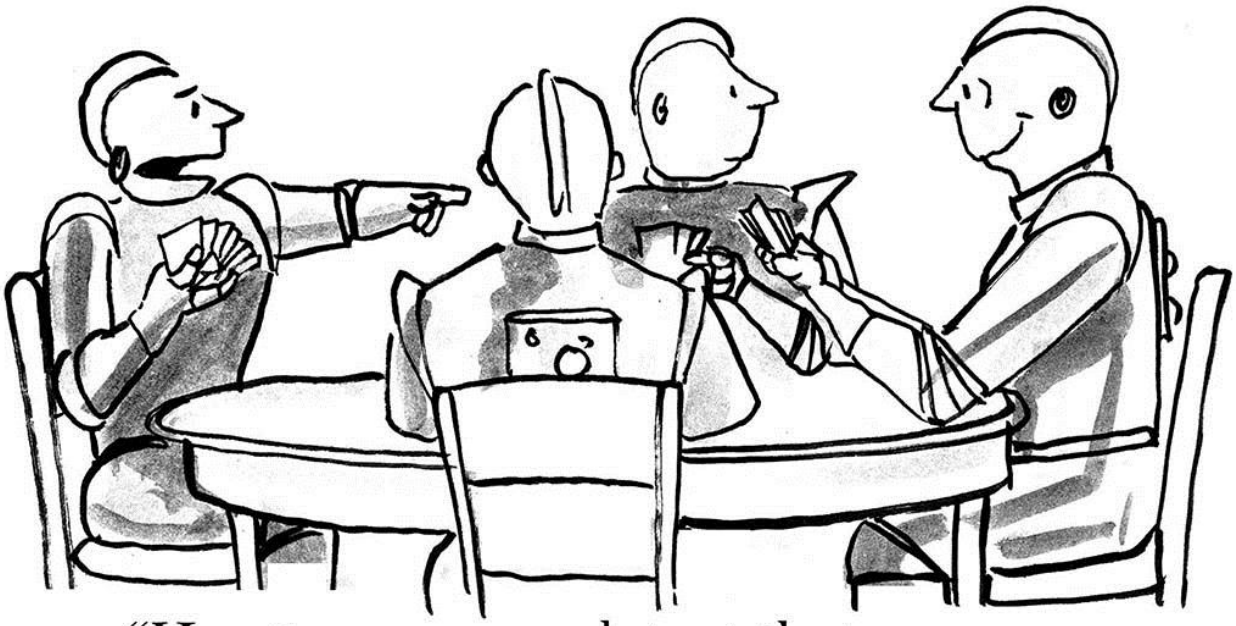
ΑΥΤΟΝΟΜΟ ΟΧΗΜΑ ΚΙΝΟΥΜΕΝΟ ΜΕ ΤΗΝ ΑΝΑΓΝΩΡΙΣΗ ΣΗΜΑΤΩΝ ΚΥΚΛΟΦΟΡΙΑΣ

ΞΕΝΙΚΑΚΗΣ ΓΕΩΡΓΙΟΣ – Α/Μ: 42035

ΕΙΣΗΓΗΤΗΣ: ΕΛΛΗΝΑΣ ΙΩΑΝΝΗΣ

ΑΙΓΑΛΕΩ

2017



"Hey, my sensors detect that you are scanning my cards!"

ΔΗΛΩΣΗ ΣΥΓΓΡΑΦΕΑ ΠΤΥΧΙΑΚΗΣ ΕΡΓΑΣΙΑΣ

Ο κάτωθι υπογεγραμμένος **Ξενικάκης Γεώργιος** του **Ελευθερίου**, φοιτητής του Τμήματος Μηχανικών Η/Υ Συστημάτων 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 απόφασή της, μετά από αίτηση του ενδιαφερομένου, του αναθέτει εκ νέου την εκπόνηση Π.Ε με άλλο θέμα και διαφορετικό επιβλέποντα καθηγητή. Η εκπόνηση της εν λόγω Π.Ε πρέπει να ολοκληρωθεί εντός τουλάχιστον ενός ημερολογιακού 6μήνου από την ημερομηνία ανάθεσής της. Κατά τα λοιπά εφαρμόζονται τα προβλεπόμενα στο άρθρου 18, παρ. 5 του ισχύοντος Εσωτερικού Κανονισμού.»

Ο Δηλών Ημερομηνία

ΕΥΧΑΡΙΣΤΙΕΣ

Θα ήθελα αρχικά, να εκφράσω τις ευχαριστίες μου για την οικογένειά μου και κυρίως για τον Πατέρα μου που με βοήθησε και με στήριξε για την ολοκλήρωση της πτυχιακής μου εργασίας.

Έπειτα, θα ήθελα να ευχαριστήσω τον Κύριο Ιωάννη Έλληνα, υπεύθυνο του τμήματος Μηχανικών Ηλεκτρονικών Υπολογιστικών Συστημάτων που μου ανέθεσε ένα πολύ όμορφο Project το οποίο μου έδωσε την δυνατότητα να ασχοληθώ με μια πρωτόγνωρη γλώσσα προγραμματισμού για εμένα, την Python η οποία αν και στην αρχή με δυσκόλεψε εν τέλει μου άρεσε πάρα πολύ και χάρηκα ιδιαίτερα που είχα μια «τριβή» μαζί της.

Τέλος, θα ήθελα να ευχαριστήσω και όλους τους συναδέλφους και Καθηγητές, που συνυπήρξαμε στα εργαστηριακά μαθήματα και με βοήθησαν να ολοκληρώσω των κύκλο σπουδών μου επιτυχώς, έχοντας λάβει ιδιαίτερες γνώσεις του αντικειμένου που με ενδιαφέρει.

Ημερομηνία:

Συγγραφέας: Ξενικάκης Γεώργιος

Περίληψη

Στη παρούσα πτυχιακή εργασία παρουσιάζεται η υλοποίηση ενός αυτόνομου οχήματος τεσσάρων τροχών το οποίο αναπτύχθηκε πάνω σε μια πλατφόρμα «Raspberry Pi 2 Model B». Πάνω στο όχημα υπάρχει μια κάμερα με την οποία γίνεται και μία πραγματικού χρόνου βίντεο-σύλληψη από την οπτική γωνία του οχήματος. Με την βοήθεια του Raspberry Pi και της βιβλιοθήκης μηχανικής όρασης OpenCV γίνεται η επεξεργασία των καρέ του βίντεο και μας δίνεται η δυνατότητα αναγνώρισης ορισμένων σημάτων του Κώδικα Οδικής Κυκλοφορίας. Αναγνωρίζοντας τα σήματα αυτά, το όχημα ανταποκρίνεται ανάλογα για το καθένα τους. Αυτό είναι και που το καθιστά αυτόνομο. Πάνω στο όχημα υπάρχουν και αισθητήρες υπερήχων οι οποίοι μας δίνουν την δυνατότητα να ανιχνεύουμε εμπόδια και να αποτρέπουμε το όχημα από τυχόν συγκρούσεις. Επίσης το όχημα, διαθέτει και τέσσερα Leds τα οποία προσομοιώνουν τα τέσσερα Flash των πραγματικών αυτοκινήτων και μας δείχνουν την πρόθεση του οχήματός μας.

Abstract

This thesis presents the implementation of an autonomous vehicle of four wheels which was developed on a platform “Raspberry Pi 2 Model B”. On the vehicle there is a camera that receives a real-time video from the perspective of the vehicle. The platform Raspberry Pi and the library of machine vision Open CV perform the processing of the video frames and as a result our vehicle has the ability to recognize certain signs of the Highway Code. By recognizing these signs, the vehicle responds accordingly to each of them. This is it that makes it autonomous. On the vehicle there are ultrasound sensors that enable us to detect obstacles and prevent the vehicle from any conflicts. Also, the vehicle has four Leds which simulate the four Flash of a real car and show us the intention of our vehicle.

Περιεχόμενα

OpenCV

Λογισμικά - Software

Υλικό - Hardware

Ορισμός προβλήματος και αλγόριθμοι επίλυσης

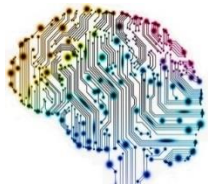
Λεπτομέρειες εγκατάστασης

Συμπεράσματα και μελλοντική εξέλιξη

Βιβλιογραφία - Αναφορές

ΚΕΦΑΛΑΙΟ 1

ΘΕΩΡΗΤΙΚΟ ΥΠΟΒΑΘΡΟ



1.1 Τεχνητή Νοημοσύνη

Ο όρος **τεχνητή νοημοσύνη** αναφέρεται στον κλάδο της πληροφορικής ο οποίος ασχολείται με τη σχεδίαση και την υλοποίηση υπολογιστικών συστημάτων που μιμούνται στοιχεία της ανθρώπινης συμπεριφοράς τα οποία υπονοούν έστω και στοιχειώδη ευφυΐα: μάθηση, προσαρμοστικότητα, εξαγωγή συμπερασμάτων, κατανόηση από συμφραζόμενα, επίλυση προβλημάτων κλπ. Ο Τζον Μακάρθι όρισε τον τομέα αυτόν ως «επιστήμη και μεθοδολογία της δημιουργίας νοούντων μηχανών».

Η τεχνητή νοημοσύνη αποτελεί σημείο τομής μεταξύ πολλαπλών επιστημών όπως της πληροφορικής, της ψυχολογίας, της φιλοσοφίας, της νευρολογίας, της γλωσσολογίας και της επιστήμης μηχανικών, με στόχο τη σύνθεση ευφυούς συμπεριφοράς, με στοιχεία συλλογιστικής, μάθησης και προσαρμογής στο περιβάλλον, ενώ συνήθως εφαρμόζεται σε μηχανές ή υπολογιστές ειδικής κατασκευής. Διαιρείται στη συμβολική τεχνητή νοημοσύνη, η οποία επιχειρεί να εξομοιώσει την ανθρώπινη νοημοσύνη αλγοριθμικά χρησιμοποιώντας σύμβολα και λογικούς κανόνες υψηλού επιπέδου, και στην υποσυμβολική τεχνητή νοημοσύνη, η οποία προσπαθεί να αναπαράγει την ανθρώπινη ευφυΐα χρησιμοποιώντας στοιχειώδη αριθμητικά μοντέλα που συνθέτουν επαγωγικά νοήμονες συμπεριφορές με τη διαδοχική αυτοοργάνωση απλούστερων δομικών συστατικών («συμπεριφορική τεχνητή νοημοσύνη»), προσομοιώνουν πραγματικές βιολογικές διαδικασίες όπως η εξέλιξη των ειδών και η λειτουργία του εγκεφάλου («υπολογιστική νοημοσύνη»), ή αποτελούν εφαρμογή στατιστικών μεθοδολογιών σε προβλήματα τεχνητής νοημοσύνης.

Η διάκριση σε συμβολικές και υποσυμβολικές προσεγγίσεις αφορά τον χαρακτήρα των χρησιμοποιούμενων εργαλείων, ενώ δεν είναι σπάνια η σύζευξη πολλαπλών προσεγγίσεων (διαφορετικών συμβολικών, υποσυμβολικών, ή ακόμα συμβολικών και υποσυμβολικών

μεθόδων) κατά την προσπάθεια αντιμετώπισης ενός προβλήματος. Με βάση τον επιθυμητό επιστημονικό στόχο η Τεχνητή Νοημοσύνη κατηγοριοποιείται σε άλλου τύπου ευρείς τομείς, όπως επίλυση προβλημάτων, μηχανική μάθηση, ανακάλυψη γνώσης, συστήματα γνώσης κλπ. Επίσης υπάρχει επικάλυψη με συναφή επιστημονικά πεδία όπως η μηχανική όραση, η επεξεργασία φυσικής γλώσσας ή η ρομποτική, τα οποία μπορούν να τοποθετηθούν μες στο ευρύτερο πλαίσιο της σύγχρονης τεχνητής νοημοσύνης ως ανεξάρτητα πεδία της.

1.1.1 Διαχωρισμός Τεχνητής από τη Φυσική Νοημοσύνη

Η εφαρμογή της Τεχνητής Νοημοσύνης σε μηχαζνές καθιστούν τις μηχανές αυτές ευφυείς και τις κάνουν να συμπεριφέρονται σχεδόν όπως ο άνθρωπος. Ο διαχωρισμός των μηχανών αυτών ως ευφυείς ή όχι καθορίζεται με τη βοήθεια του κριτηρίου που έθεσε ο Turing (δοκιμασία Turing).

Ο Alan Turing (1912-1953), ο οποίος θεωρείται ο πατέρας της ΤΝ, εμπνεύστηκε το 1950 μία δοκιμασία η οποία πήρε και το όνομα του (Turing Test - Δοκιμασία Turing), για το χαρακτηρισμό των μηχανών. Αυτό βασίζεται σε μία σειρά από ερωτήσεις που υποβάλει κάποιος ταυτόχρονα σε έναν άνθρωπο και μία μηχανή χωρίς να γνωρίζει εκ των προτέρων ποιος είναι τι. Αν στο τέλος δεν καταφέρει να ξεχωρίσει τον άνθρωπο από τη μηχανή, τότε η μηχανή πετυχαίνει στη δοκιμασία και θεωρείται ευφυής [1]. Αν και η αποτελεσματικότητα της Δοκιμασίας Turing εξαρτάται από πολλούς παράγοντες, θεωρείται μέχρι σήμερα ένα καλό μέτρο σύγκρισης της φυσικής με την τεχνητή νοημοσύνη και πολλοί διαγωνισμοί διοργανώνονται σε ετήσια βάση, χωρίς όμως ιδιαίτερα σοβαρά ή τουλάχιστο χρήσιμα αποτελέσματα.

Ο προγραμματισμός ενός υπολογιστή για να περάσει τη δοκιμασία Turing απαιτεί τη συμμετοχή αρκετών επιστημονικών περιοχών, όπως της επεξεργασίας φυσικής γλώσσας (natural language processing) για επικοινωνία σε φυσική γλώσσα, της αναπαράστασης γνώσης (knowledge representation) για την αποθήκευση της γνώσης πριν και κατά τη διάρκεια της δοκιμασίας, της αυτοματοποιημένης συλλογιστικής (automated reasoning) για τη χρήση της αποθηκευμένης

πληροφορίας και την εξαγωγή συμπερασμάτων, της μηχανικής μάθησης (machine learning) για προσαρμογή σε νέες περιπτώσεις, κλπ.

Στην αρχική της μορφή, η δοκιμασία Turing δεν προέβλεπε φυσική επαφή ανθρώπου μηχανής. Ωστόσο μια επέκταση της (Πλήρης Δοκιμασία Turing) περιλαμβάνει και την αναγνώριση εικόνων και αντικειμένων που ανταλλάσσονται μέσα από κάποια θυρίδα ώστε να μην υπάρχει οπτική επαφή με το δοκιμαζόμενο για τον έλεγχο των δυνατοτήτων αντίληψης του. Για το σκοπό αυτό απαιτείται η συμμετοχή και άλλων δύο επιστημονικών περιοχών, της μηχανικής όρασης (machine vision) για την αναγνώριση και της ρομποτικής (robotics) για τη μετακίνηση τους.

1.1.2 Ισχυρή Τεχνητή Νοημοσύνη

Η λογοτεχνία και ο κινηματογράφος επιστημονικής φαντασίας από τη δεκαετία του 1920 μέχρι σήμερα έχουν δώσει στο ευρύ κοινό την αίσθηση ότι η Τεχνητή Νοημοσύνη αφορά την προσπάθεια κατασκευής μηχανικών ανδροειδών ή αυτοσυνείδητων προγραμμάτων υπολογιστή (ισχυρή Τεχνητή Νοημοσύνη), επηρεάζοντας μάλιστα ακόμα και τους πρώτους ερευνητές του τομέα. Στην πραγματικότητα οι περισσότεροι επιστήμονες της τεχνητής νοημοσύνης προσπαθούν να κατασκευάσουν λογισμικό ή πλήρεις μηχανές οι οποίες να επιλύουν με αποδεκτά αποτελέσματα ρεαλιστικά υπολογιστικά προβλήματα οποιουδήποτε τύπου (ασθενής Τεχνητή Νοημοσύνη), αν και πολλοί πιστεύουν ότι η εξομοίωση ή η προσομοίωση της πραγματικής ευφυΐας, η **ισχυρή Τεχνητή Νοημοσύνη**, πρέπει να είναι ο τελικός στόχος.

Η σύγχρονη τεχνητή νοημοσύνη αποτελεί ένα από τα πλέον «μαθηματικοποιημένα» και ταχέως εξελισσόμενα πεδία της πληροφορικής. Σήμερα, ο τομέας αξιοποιεί περισσότερο υποσυμβολικές μεθόδους και εργαλεία καταγόμενα από τα εφαρμοσμένα μαθηματικά και τις επιστήμες μηχανικών, παρά από τη θεωρητική πληροφορική και τη μαθηματική λογική όπως συνέβαινε πριν το 1990. Σε ακαδημαϊκό επίπεδο η τεχνητή νοημοσύνη μελετάται επίσης από

την ηλεκτρονική μηχανική, ενώ συνιστά ένα από τα σημαντικότερα θεμελιακά συστατικά του διεπιστημονικού γνωστικού πεδίου της γνωσιακής επιστήμης.

1.1.3 Ιστορική Αναδρομή

Κατά τη δεκαετία του 1940 εμφανίστηκε η πρώτη μαθηματική περιγραφή τεχνητού νευρωνικού δικτύου, με πολύ περιορισμένες δυνατότητες επίλυσης αριθμητικών προβλημάτων. Καθώς ήταν εμφανές ότι οι ηλεκτρονικές υπολογιστικές συσκευές που κατασκευάστηκαν μετά τον Β' Παγκόσμιο Πόλεμο ήταν ένα τελείως διαφορετικό είδος μηχανής από ό,τι προηγήθηκε, η συζήτηση για την πιθανότητα εμφάνισης μηχανών με νόηση ήταν στην ακμή της. Το 1950 ο μαθηματικός Άλαν Τούρινγκ, πατέρας της θεωρίας υπολογισμού και προπάτορας της τεχνητής νοημοσύνης, πρότεινε τη δοκιμή Τούρινγκ· μία απλή δοκιμασία που θα μπορούσε να εξακριβώσει αν μία μηχανή διαθέτει ευφυΐα. Η τεχνητή νοημοσύνη θεμελιώθηκε τυπικά ως πεδίο στη συνάντηση ορισμένων επιφανών Αμερικανών επιστημόνων του τομέα το 1956 (Τζον Μακάρθι, Μάρβιν Μίνσκυ, Κλοντ Σάνον κλπ). Τη χρονιά αυτή παρουσιάστηκε για πρώτη φορά και το Logic Theorist, ένα πρόγραμμα το οποίο στηριζόταν σε συμπερασματικούς κανόνες τυπικής λογικής και σε ευρετικούς αλγορίθμους αναζήτησης για να αποδεικνύει μαθηματικά θεωρήματα.

Επόμενοι σημαντικοί σταθμοί ήταν η ανάπτυξη της γλώσσας προγραμματισμού LISP το 1958 από τον Μακάρθι, δηλαδή της πρώτης γλώσσας συναρτησιακού προγραμματισμού η οποία έπαιξε πολύ σημαντικό ρόλο στη δημιουργία εφαρμογών Τεχνητής Νοημοσύνης κατά τις επόμενες δεκαετίες, η εμφάνιση των γενετικών αλγορίθμων την ίδια χρονιά από τον Φρίντμπεργκ και η παρουσίαση του βελτιωμένου νευρωνικού δικτύου perceptron το '62 από τον Ρόσενμπλατ. Κατά τα τέλη της δεκαετίας του '60 όμως άρχισε ο χειμώνας της Τεχνητής Νοημοσύνης, μία εποχή κριτικής, απογοήτευσης και υποχρηματοδότησης των ερευνητικών προγραμμάτων καθώς όλα τα μέχρι τότε εργαλεία του χώρου ήταν κατάλληλα μόνο για την επίλυση εξαιρετικά απλών προβλημάτων. Στα μέσα του '70 ωστόσο προέκυψε μία αναθέρμανση του ενδιαφέροντος για τον τομέα λόγω των εμπορικών εφαρμογών που

απέκτησαν τα έμπειρα συστήματα, μηχανές Τεχνητής Νοημοσύνης με αποθηκευμένη γνώση για έναν εξειδικευμένο τομέα και δυνατότητα ταχείας εξαγωγής λογικών συμπερασμάτων, τα οποία συμπεριφέρονται όπως ένας άνθρωπος ειδικός στον αντίστοιχο τομέα. Παράλληλα έκανε την εμφάνισή της η γλώσσα λογικού προγραμματισμού Prolog η οποία έδωσε νέα ώθηση στη συμβολική Τεχνητή Νοημοσύνη, ενώ στις αρχές της δεκαετίας του '80 άρχισαν να υλοποιούνται πολύ πιο ισχυρά και με περισσότερες εφαρμογές νευρωνικά δίκτυα, όπως τα πολυεπίπεδα perceptron και τα δίκτυα Hopfield. Ταυτόχρονα οι γενετικοί αλγόριθμοι και άλλες συναφείς μεθοδολογίες αναπτύσσονταν πλέον από κοινού, κάτω από την ομπρέλα του εξελικτικού υπολογισμού.

Κατά τη δεκαετία του '90, με την αυξανόμενη σημασία του Internet, ανάπτυξη γνώρισαν οι ευφυείς πράκτορες, αυτόνομο λογισμικό Τεχνητής Νοημοσύνης τοποθετημένο σε κάποιο περιβάλλον με το οποίο αλληλεπιδρά, οι οποίοι βρήκαν μεγάλο πεδίο εφαρμογών λόγω της εξάπλωσης του Διαδικτύου. Οι πράκτορες στοχεύουν συνήθως στην παροχή βοήθειας στους χρήστες τους, στη συλλογή ή ανάλυση γιγάντιων συνόλων δεδομένων ή στην αυτοματοποίηση επαναλαμβανόμενων εργασιών (π.χ. βλέπε διαδικτυακό ρομπότ), ενώ στους τρόπους κατασκευής και λειτουργίας τους συνοψίζουν όλες τις γνωστές μεθοδολογίες Τεχνητής Νοημοσύνης που αναπτύχθηκαν με το πέρασμα του χρόνου. Έτσι σήμερα, όχι σπάνια, η Τεχνητή Νοημοσύνη ορίζεται ως η επιστήμη που μελετά τη σχεδίαση και υλοποίηση ευφυών πρακτόρων.

Επίσης τη δεκαετία του '90 η Τεχνητή Νοημοσύνη, κυρίως η μηχανική μάθηση και η ανακάλυψη γνώσης, άρχισε να επηρεάζεται πολύ από τη θεωρία πιθανοτήτων και τη στατιστική. Τα δίκτυα πεποιθήσεων υπήρξαν η αφετηρία αυτής της νέας μετακίνησης, που συνέδεσε τελικά την Τεχνητή Νοημοσύνη με τα πιο σχολαστικά μαθηματικά εργαλεία της στατιστικής και της επιστήμης μηχανικών, όπως τα κρυμμένα μαρκοβιανά μοντέλα και τα φίλτρα Κάλμαν. Αυτή η νέα πιθανοκρατική προσέγγιση έχει αυστηρά υποσυμβολικό χαρακτήρα, όπως και οι τρεις μεθοδολογίες οι οποίες κατηγοριοποιούνται κάτω από την ετικέτα της υπολογιστικής νοημοσύνης: τα νευρωνικά δίκτυα, ο εξελικτικός υπολογισμός και η ασαφής λογική.

1.2 Μηχανική Όραση

Η Μηχανική Όραση ή Τεχνητή Όραση ονομάζεται ο επιστημονικός τομέας της επιστήμης υπολογιστών, το οποίο περιλαμβάνει μεθόδους για την απόκτηση, την επεξεργασία, την ανάλυση και την κατανόηση εικόνων του πραγματικού κόσμου, στοχεύοντας στην παραγωγή αριθμητικής ή συμβολικής πληροφορίας. Με άλλα λόγια, προσπαθεί να ενσωματώσει τη λογική στην όραση, όπως ακριβώς συμβαίνει στον άνθρωπο. Οι πληροφορίες εισόδου αντλούνται από ψηφιακές φωτογραφίες, βίντεο και σαρωτές.

Η πρόοδος της τεχνολογίας έχει βοηθήσει κατά πολύ αυτό το επιστημονικό πεδίο, το οποίο έχει προσελκύσει τα τελευταία χρόνια όλο και περισσότερους επιστήμονες να ασχοληθούν με αυτό. Έτσι, οι εφαρμογές και τα συστήματα που προσομοιάζουν τη λειτουργία του ματιού έχουν γνωρίσει άνθιση. Παρ' όλα αυτά, η ανθρώπινη όραση είναι τόσο ανεπτυγμένη, που η μηχανική όραση απαιτεί πολύπλοκες διεργασίες και επεξεργασίες των εικόνων για να καταφέρει να τη φτάσει. Για παράδειγμα, ένας άνθρωπος μπορεί να αναγνωρίσει ένα αντικείμενο ως καρέκλα, χωρίς να έχει ξαναδεί την ίδια καρέκλα, απλά και μόνο από την έμφυτη ικανότητα της όρασης, μέσω της οποίας μπορεί να εξάγει συμπεράσματα για το χρώμα, το σχήμα, την κίνηση, την υφή και το μέγεθος ενός αντικειμένου.

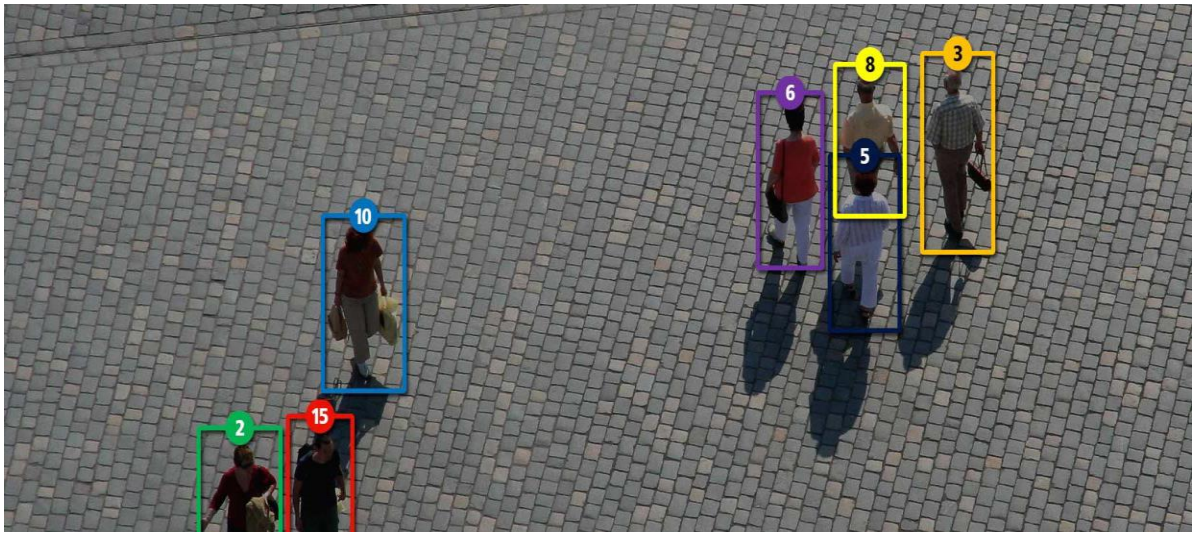
Σε αντίθεση με τον άνθρωπο, στη μηχανική όραση για να γίνει κάτι τέτοιο, απαιτείται η επεξεργασία πληροφοριών από μια ακολουθία από εικόνες. Μερικά από τα πιθανά προβλήματα που θα συναντήσει κάποιος που ασχολείται με αυτό τον επιστημονικό κλάδο είναι η απώλεια πληροφορίας κατά την επεξεργασία της εικόνας, οι ακολουθίες εικόνων που απαιτούν μεγάλο αποθηκευτικό χώρο, οι διαφορές κλίμακας των αντικειμένων, η εναλλαγή της φωτεινότητας, οι οπτικές γωνίες, η ύπαρξη θορύβου και άλλες πολλές δυσνόητες και δυσεπίλυτες έννοιες.

Η πολυπλοκότητα των προβλημάτων, των εννοιών και εν γένει ολόκληρου του πεδίου, έχει οδηγήσει στην ιεραρχική διάκριση των λειτουργιών της μηχανικής όρασης. Στη βάση της ιεραρχίας βρίσκεται η χαμηλού επιπέδου όραση, κατά την οποία γίνεται η εξαγωγή χαρακτηριστικών, όπως οι γωνίες και οι άκρες. Αμέσως από πάνω, βρίσκεται η όραση μεσαίου επιπέδου, κατά την οποία τα αντικείμενα αναγνωρίζονται και γίνεται η ερμηνεία της

τρισδιάστατης εικόνας, βάσει των χαρακτηριστικών που προέρχονται από το χαμηλότερο επίπεδο. Τέλος, έχουμε την όραση υψηλού επιπέδου, όπου ερμηνεύεται η εξελισσόμενη πληροφορία που προέρχεται από το μεσαίο επίπεδο. Η ερμηνεία αυτή, μπορεί να περιλαμβάνει την εννοιολογική περιγραφή μιας σκηνής, όπως την δραστηριότητα, την πρόθεση και τη συμπεριφορά.

Η Τεχνητή Νοημοσύνη, η Επεξεργασία Εικόνας και η Αναγνώριση Προτύπων είναι τα επιστημονικά πεδία από τα οποία μπορούμε να πούμε, ότι προήλθε η Τεχνητή Όραση. Τα πρώτα βήματα της Τεχνητής Όρασης βασίστηκαν στους ήδη υπάρχοντες αλγορίθμους της Τεχνητής Νοημοσύνης, από τη δεκαετία του ’70, που οι επιστήμονες του τομέα είχαν επικεντρωθεί στην ανάπτυξη υπολογιστών, με δυνατότητα όρασης. Από την Επεξεργασία Εικόνας φάνηκαν -και συνεχίζουν να είναι- χρήσιμοι οι μετασχηματισμοί, που βοηθούν στην εύρεση θορύβου, ακμών και στην περιστροφή των εικόνων. Οι τεχνικές Αναγνώρισης Προτύπων, όπως οι στατιστικές προσεγγίσεις και τα νευρωνικά δίκτυα, χρησιμοποιούνται συστηματικά στην Μηχανική Όραση.

Η σχέση, όμως, της Τεχνητής Όρασης με άλλους τομείς ή κλάδους της επιστήμης, είναι αναπόφευκτη. Η χρήση της είναι συχνό φαινόμενο σε εφαρμογές υψηλής τεχνολογίας, όπως στη βιομηχανία, την ιατρική, ακόμα και στην εξερεύνηση του διαστήματος.



Εικόνα 1.1: Παράδειγμα Μηχανικής Όρασης - Αναγνώριση κινούμενων ανθρώπων

Εικόνα 1.2: Παράδειγμα Μηχανικής Όρασης
Αναγνώριση Προσώπων

1.2.1 Μηχανική Όραση και προβλήματα

Παρακάτω αναφέρονται μερικά από τα πιο βασικά προβλήματα, τα οποία θα κληθεί να αντιμετωπίσει κάποιος που ασχολείται με τη Μηχανική Όραση.

Η αναγνώριση αντικειμένων (object recognition) πρόκειται για τον έλεγχο, σαρώνοντας την εικόνα, για το αν υπάρχουν γενικά ή ένα συγκεκριμένο αντικείμενο. Το αντικείμενο αυτό, θα είναι κάτι συγκεκριμένο-γνωστό αν πρόκειται για μια γραμμή παραγωγής, όπως συμβαίνει στο παρόν σύστημα. Μπορεί όμως να είναι και κάτι άγνωστο, όπως συμβαίνει στην περίπτωση της αναγνώρισης προσώπου.

Στην περίπτωση της αναγνώρισης προσώπου, αλλά και σε άλλες εφαρμογές που απαιτούν τον υπολογισμό ενός τρισδιάστατου μοντέλου, σημαντικό πρόβλημα αποτελεί η ανακατασκευή σκηνής, αφού υπάρχουν περισσότερες από μια εικόνες.

Τέλος, υπάρχει το πρόβλημα της κίνησης αντικείμενων (tracking). Στην περίπτωση των βιομηχανικών εφαρμογών, που έχουμε πολλά κινούμενα αντικείμενα για τα οποία πρέπει να γίνονται έλεγχοι, πρέπει να γίνει απαραίτητα επεξεργασία σε μια αλληλουχία εικόνων, για να εξαχθεί κάποιο συμπέρασμα.

1.2.2 Εφαρμογές Μηχανικής Όρασης

Όπως είδαμε και παραπάνω, η Μηχανική Όραση βρίσκει εφαρμογή από πολλές επιστήμες. Κάποιες περιπτώσεις, εφαρμογών ή συστημάτων, που χρησιμοποιούν την Μηχανική Όραση, αποτελούν αυτόνομα συστήματα, που λειτουργούν και εξάγουν πληροφορίες μόνα τους, ενώ κάποια άλλα αποτελούν υποσυστήματα ενός μεγαλύτερου συστήματος. Σε όλες, όμως, τις περιπτώσεις, ακολουθούνται κάποια βασικά βήματα. Το πρώτο βήμα, δεν θα μπορούσε να είναι άλλο από την ανάκτηση της πληροφορίας εισόδου, δηλαδή της εικόνας, της αλληλουχίας εικόνων ή του βίντεο. Επόμενο στάδιο, είναι η επεξεργασία της εισερχόμενης πληροφορίας για την εξαγωγή χαρακτηριστικών, τα οποία θα ελεγχτούν με τη σειρά τους, για την παραγωγή ακόμα πιο χρήσιμων πληροφοριών. Στη συνέχεια, σειρά έχει η κατάτμηση της εικόνας από την οποία παίρνουμε τις πληροφορίες που έχουν το περισσότερο ενδιαφέρον. Τελικό στάδιο, είναι η παραγωγή κάποιας απόφασης που ενεργοποιεί πιθανώς άλλα συστήματα ή προκαλεί κάποια αντίδραση του χρήστη. Φυσικά και το παρόν σύστημα που αναλύουμε έχει ακολουθήσει πιστά τα παραπάνω βήματα.

Παρακάτω αναφέρονται μερικά παραδείγματα εφαρμογών και περιπτώσεων που χρησιμοποιούν Μηχανική Όραση :

- Έλεγχος διεργασιών σε ρομποτικές εφαρμογές και αυτόνομα οχήματα.
- Ανίχνευση γεγονότων σε συστήματα παρακολούθησης.
- Οργάνωση πληροφοριών με τη δεικτοδότηση βάσεων εικόνων.
- Κατασκευή μοντέλων ή περιβάλλοντος, που χρησιμοποιείται στις ιατρικές εικόνες και στην τοπογραφία.
- Εφαρμογές αλληλεπίδρασης ανθρώπου-υπολογιστή με φυσικές κινήσεις και μεθόδους.
- Στρατιωτικούς σκοπούς για την καθοδήγηση πυραύλων και ανίχνευση αντίπαλων στρατευμάτων.
- Βιομηχανία στον έλεγχο των γραμμών παραγωγής.

Ένας από τους πιο διαδεδομένους τομείς χρήσης Μηχανικής Όρασης είναι τα στρατιωτικά- οπλικά συστήματα. Η ποικιλία των συστημάτων αυτών ξεπερνά τη φαντασία, όλα όμως έχουν κοινό σκοπό, ανωτερότητα του κατόχου έναντι των αντιπάλων του. Αναμφισβήτητα, τα πιο γνωστά παραδείγματα είναι η καθοδήγηση πυραύλων και η ανίχνευση εχθρικών στρατευμάτων. Από τα πιο σύγχρονα και τελευταία συστήματα, είναι οι αισθητήρες στο πεδίο μάχης. Οι αισθητήρες, εκτός των άλλων, είναι και αισθητήρες που προσφέρουν εικόνα από το σημείο της σύγκρουσης, συμβάλλοντας στην λήψη κρίσιμων στρατηγικών αποφάσεων.

Πολύ σημαντικός, επίσης, είναι ο ιατρικός τομέας. Η ιατρική υπολογιστική όραση (medical computer vision or medical image processing) βοηθάει στην (εξαγωγή) της ιατρικής γνωμάτευσης από τα δεδομένα της εικόνας. Η εικόνα μπορεί να προέρχεται από μικροσκόπια, από υπέρηχους, τομογράφους, ακτίνες Χ και άλλα μέσα. Η ανίχνευση όγκων και η αρτηριοσκλήρωση είναι παθήσεις που μπορούν να αντιμετωπιστούν με την βοήθεια της Μηχανικής Όρασης. Πέρα όμως από την πρόληψη και την διάγνωση, η Μηχανική Όραση έχει βοηθήσει την ιατρική και στον τομέα της έρευνας, δίνοντας πληροφορίες για την δομή του εγκεφάλου και την ποιότητα των ιατρικών θεραπειών.

Άλματα, όμως, έχουν γίνει και στον τομέα της βιομηχανίας με την ένταξη της Μηχανικής Όρασης στα συστήματα παραγωγής. Στο συγκεκριμένο τομέα, βρίσκει εφαρμογή και το σύστημα που αναλύεται στην παρούσα εργασία. Η ομαλή λειτουργία της γραμμής παραγωγής παρακολουθείται και εξασφαλίζεται μέσα από δεδομένα που εξάγονται από εικόνες ή αλληλουχίες εικόνας. Με αυτό τον τρόπο, απομακρύνονται τα ανεπιθύμητα προϊόντα ή τα προϊόντα που δεν πληρούν τις απαιτήσεις της παράγωγης (του ποιοτικού ελέγχου).

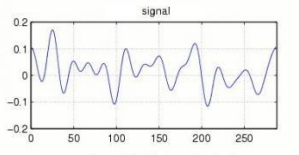
1.2.3 Ιστορική Αναδρομή

Ιστορικά, η υπολογιστική όραση αναδύθηκε μετά το 1980 ως αποτέλεσμα επέκτασης του πεδίου της πληροφορικής το οποίο καλείται ψηφιακή επεξεργασία εικόνας σε αλγορίθμους ανάλυσης και κατανόησης εικόνων. Είχαν προηγηθεί η μαθηματική μοντελοποίηση της φυσικής όρασης, έστω σε ένα βασικό επίπεδο, και οι πρώτες προσπάθειες για αναπαραγωγή της αίσθησης της όρασης σε αυτόνομα ρομπότ. Ως τότε ο όρος μηχανική όραση σχετιζόταν με την ηλεκτρολογία και τη ρομποτική, συνήθως σε βιομηχανικό πλαίσιο. Κατά τη δεκαετία του 1980, μετά την εμφάνιση της υπολογιστικής όρασης, οι δύο όροι σταδιακά συνέκλιναν και συγχωνεύθηκαν ως επιστημονικά πεδία, σαν διακριτός τομέας της τεχνητής νοημοσύνης με εφαρμογές όχι μόνο στη ρομποτική αλλά και σε δεκάδες ακόμα κλάδους.

Από τη δεκαετία του 1990 κι έπειτα η μηχανική όραση έχει γνωρίσει αλματώδη ανάπτυξη, έχει συνδεθεί με το γνωστικό πεδίο της μηχανικής μάθησης και έχει δώσει σημαντικά απτά αποτελέσματα, με αλγορίθμους όρασης πραγματικού χρόνου να υλοποιούνται ακόμα και σε φτηνά κινητά τηλέφωνα εξοπλισμένα με κάμερα. Στο εν λόγω πλαίσιο, η μηχανική όραση έχει διαδραματίσει θεμελιώδη ρόλο στην εξέλιξη της ενισχυμένης πραγματικότητας.

Μετά την ευρύτατη διάδοση του Kinect, ενός καινοτόμου περιφερειακού διασύνδεσης μεταξύ χρηστών και υπολογιστικών συστημάτων, και τη σχετική άνθιση του τριδιάστατου (στερεοσκοπικού) οπτικού περιεχομένου ύστερα από τη μεγάλη επιτυχία της κινηματογραφικής ταινίας Άβαταρ το 2009, η μηχανική όραση έχει αρχίσει να εξετάζει πιο ενδελεχώς και την αξιοποίηση δεδομένων βάθους (π.χ. από στερεοσκοπική κάμερα ή ξεχωριστούς αισθητήρες βάθους) για την επίτευξη των στόχων της.

1.3 Ψηφιακή Επεξεργασία Εικόνας



Η Ψηφιακή Επεξεργασία Εικόνας (Digital Image Processing - DIP) αποτελεί έναν σημαντικό κλάδο της Πληροφορικής, ο οποίος βρίσκει εφαρμογή σε πολυάριθμους και ποικίλους τομείς των ανθρωπίνων δραστηριοτήτων, επιστημονικούς και μη, όπως είναι η Ρομποτική, η Τεχνητή Νοημοσύνη, τα Νευρωνικά Δίκτυα, η Ιατρική, η Αστρονομία, τα Γεωγραφικά Συστήματα Πληροφοριών, η Βιομηχανία, η Ασφάλεια διαφόρων εγκαταστάσεων και συστημάτων, ο Στρατός, ο Κινηματογράφος.

Η ανάγκη επεξεργασίας και ερμηνείας οπτικής πληροφορίας παρουσιάζεται σε ολοένα και περισσότερες εφαρμογές κάθε είδους (συστήματα επισκόπησης, ρομποτικά συστήματα, πολυμέσα, τηλεματική, ηλεκτρονικές βιβλιοθήκες, ψηφιακή τηλεόραση, τηλεδιασκέψεις κλπ). Σε άλλες εφαρμογές η προς επεξεργασία πληροφορία δεν είναι από τη φύση της οπτική (δηλ. στο ορατό φάσμα) αλλά σκοπίμως οπτικοποιείται διότι με τον τρόπο αυτό καθίσταται ευκολότερη η ερμηνεία της (π.χ. διάφορες κατηγορίες ιατρικών εικόνων, συστήματα πολυφασματικής απεικόνισης κλπ). Σε πολλές από τις παραπάνω εφαρμογές η ψηφιακή επεξεργασία εικόνας είναι το πρώτο και ιδιαίτερα βασικό στάδιο. Από την αποτελεσματικότητά του εξαρτάται η επιτυχία του επόμενου σταδίου που είναι η ανάλυση εικόνας, η οποία σε αρκετές περιπτώσεις είναι ταυτόσημη με την λεγόμενη Τεχνητή Όραση. Η ψηφιακή επεξεργασία και ανάλυση εικόνας, έχει σχέση αλληλεπίδρασης με διάφορες άλλες περιοχές της επιστήμης και της τεχνολογίας των υπολογιστών, όπως την ψηφιακή επεξεργασία σήματος, την αναγνώριση προτύπων, την τεχνητή νοημοσύνη, την γραφική, τις δομές δεδομένων, τα πολυμεσικά συστήματα, τις τηλεπικοινωνίες κλπ. Η παρακολούθηση του εν λόγω μεταπτυχιακού μαθήματος απαιτεί βασικές γνώσεις σημάτων και συστημάτων καθώς και σχετικό μαθηματικό υπόβαθρο σε περιοχές όπως γραμμική άλγεβρα και στοχαστικές διαδικασίες.

Υπάρχει μία πληθώρα ερευνητικών κέντρων DIP ανά τον κόσμο, καθένα από τα οποία είναι εξειδικευμένο σε κάποιον από τους παραπάνω τομείς και προσπαθεί μέσω της έρευνας, της ανάλυσης και του πειραματισμού να χρησιμοποιήσει τις δυνατότητες που προσφέρει η DIP

έτσι, ώστε να βελτιώσει εφαρμογές, να επιλύσει προβλήματα, να αυτοματοποιήσει ενέργειες. Όλα αυτά καθιστούν την DIP ένα πολύτιμο εργαλείο, που συμβάλλει, μαζί με πλήθος άλλων, στην ολοκληρωμένη ανάπτυξη διαφόρων ανθρωπίνων εγχειρημάτων

1.3.1 Τι είναι Ψηφιακή Εικόνα;

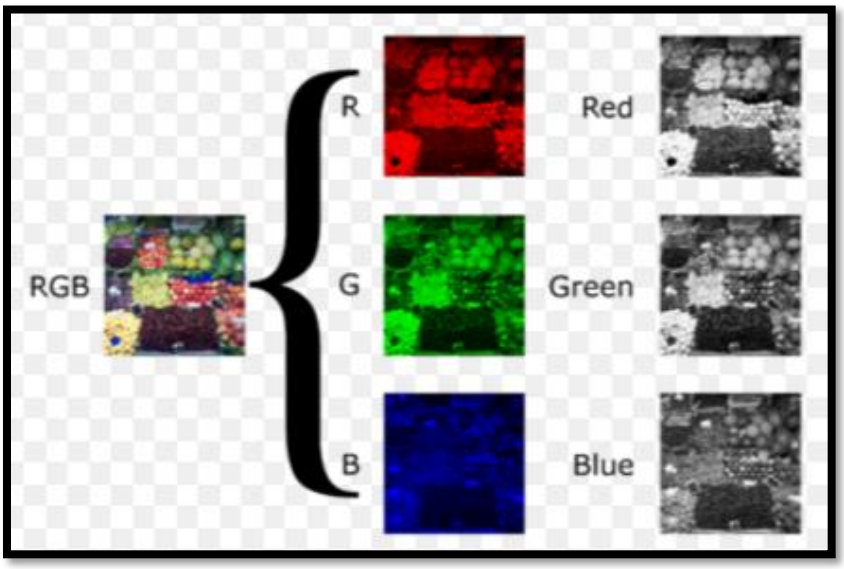
Με τον όρο ψηφιακή εικόνα, εννοούμε την μετάβαση από τον αναλογικό κόσμο στον ψηφιακό, δηλαδή τη μετατροπή των αναλογικών σημάτων σε ψηφιακά. Έτσι, μια πραγματική εικόνα μεταφέρεται στον ψηφιακό κόσμο με τη μορφή διακεκριμένου σήματος που έχει τη μορφή ψηφιακών πινάκων. Μια ψηφιακή εικόνα μπορεί να είναι δυαδική (binary images), μονοχρωματική αποχρώσεων του γκρι (gray-level ή grayscale images) ή έγχρωμη (color image).

Η απλούστερη μορφή μιας εικόνας, είναι η δυαδική μορφή. Μια δυαδική εικόνα, έχει μόνο δυο στάθμες φωτεινότητας που συνήθως είναι το μαύρο και το άσπρο. Το μαύρο αντιστοιχεί στην τιμή 0 και το άσπρο στην τιμή 1 (ή σε αντίστοιχη με grayscale εικόνες στην τιμή 255). Μια δυαδική εικόνα, επίσης, καταλαμβάνει μικρότερη μνήμη και η επεξεργασία της απαιτεί μικρότερο υπολογιστικό κόστος. Σε δυαδική μορφή μπορούν να απεικονισθούν σημαντικές πληροφορίες, όπως είναι το εμβαδόν και η θέση αντικειμένων, η μορφή αντικειμένων κ.α.

Οι έγχρωμες εικόνες, αποτελούν το μέσο για την απεικόνιση του πραγματικού κόσμου. Μια έγχρωμη ψηφιακή εικόνα, αποτελείται από τρεις gray-level εικόνες. Δηλαδή, το χρώμα κάθε εικονοστοιχείου, έχει τρεις συνιστώσες, οι οποίες αντιστοιχούν στις γκρι αποχρώσεις των αντίστοιχων εικονοστοιχείων των τριών γκρι εικόνων.

1.3.2 Είδη Ψηφιακών Εικόνων

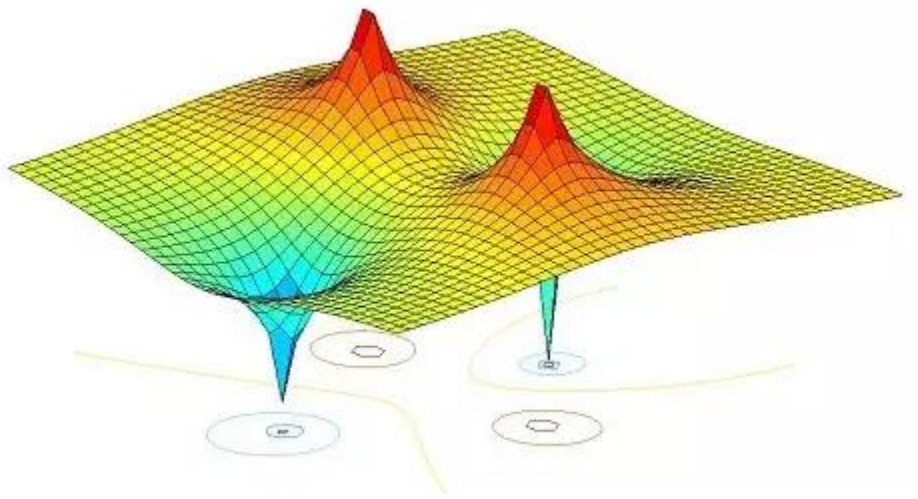
Υπάρχουν τρία είδη ψηφιακών εικόνων που χαρακτηρίζονται από το πλήθος των χρωμάτων που περιέχουν. Χωρίζονται λοιπόν: Στις δυαδικές εικόνες (binary images), έχουμε μόνο δύο αποχρώσεις: μαύρα (0) και λευκά (1) pixels. Κάθε εικονοστοιχείο δηλαδή, μπορεί να χρωματιστεί με ένα από τα δυο χρώματα (μαύρο ή άσπρο), όπως επίσης απαιτείται ένα bit πληροφορίας, π.χ. με τιμή 0 για το μαύρο και 1 για το άσπρο. Στις εικόνες αποχρώσεων του γκρι (gray-level ή gray-scale images), έχω 2^8 =256 αποχρώσεις του γκρι. Από αυτές τις αποχρώσεις, συνήθως λαμβάνονται 256 αντιπροσωπευτικές, οι οποίες κωδικοποιούνται με τιμές από 0,1,2,…,255. Η απόχρωση κάθε εικονοστοιχείου, απαιτεί πληροφορία 8 bit. Στις έγχρωμες εικόνες (color images), έχω 28 διαφορετικές αποχρώσεις του κόκκινου, 28 διαφορετικές αποχρώσεις του πράσινου και 28 διαφορετικές αποχρώσεις του μπλε (Red,Green, Blue:RGB). Δηλαδή, για κάθε ένα από αυτά τα τρία χρώματα, λαμβάνονται 256 αποχρώσεις, δηλαδή πληροφορία 8 bit. Συνεπώς, κάθε εικονοστοιχείο της έγχρωμης εικόνας, απαιτεί 24 bit.



Εικόνα 1.3: Σύνθεση μιας RGB εικόνας από τρεις Grayscale εικόνες

1.3.3 Ανάλυση και Επεξεργασία Ψηφιακής Εικόνας

Η ανάλυση εικόνας, χρησιμοποιείται για την ποσοτικοποίηση των χαρακτηριστικών - τα οποία βρίσκονται σε διασπορά (σκόρπια) – ώστε να προσδιοριστεί εάν και σε τι βαθμό διασποράς είναι ομογενείς ή ανομοιογενείς. Στην ανάλυση εικόνας, η τμηματοποίηση είναι μέρος μιας ψηφιακής εικόνας μέσα σε πολλαπλά πεδία (τμήματα pixel). Σκοπός της τμηματοποίησης είναι να εντοπίσει αντικείμενα τα οποία μας ενδιαφέρουν. Δυστυχώς, πολλοί σημαντικοί αλγόριθμοι είναι αρκετά απλοί ώστε να λύσουν αυτό το πρόβλημα επακριβώς: αντισταθμίζουν για τον συγκεκριμένο περιορισμό με την προβλεψιμότητα, την γενικότητα και την αποδοτικότητά τους



Εικόνα 1.4: Ψηφιακή αναπαράσταση επεξεργασίας εικόνας.

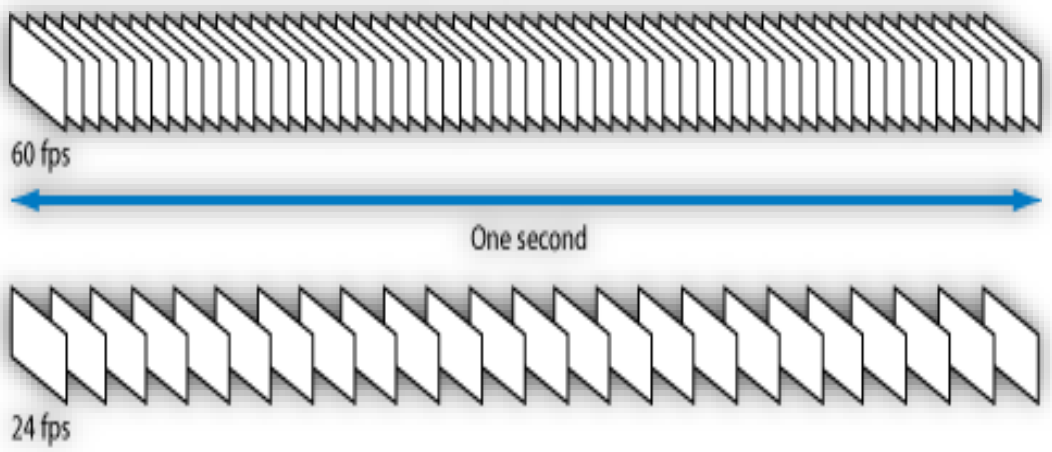
1.3.4 Το Βίντεο ως Σήμα

Τα βίντεο όπως γνωρίζουμε είναι μία σειρά από εικόνες που διαδέχονται η μία την άλλη. Η διαδοχή αυτή γίνεται αρκετά γρήγορα και με σταθερό ρυθμό δίνοντας έτσι στο θεατή τους την ψευδαίσθηση της κινούμενης εικόνας. Οι εικόνες από τις οποίες αποτελείται ένα βίντεο ονομάζονται frames ή καρέ.

Μια εικόνα γενικά μπορεί να οριστεί ως μία δισδιάστατη συνάρτηση f(x,y). Τα x και y αποτελούν τις χωρικές συντεταγμένες και η τιμή της f σε ένα σημείο (x,y) του χώρου ορίζεται ως ένταση της φωτεινότητας στο συγκεκριμένο σημείο. Αν τα x, y και f(x,y) μιας εικόνας είναι πεπερασμένες διακριτές τιμές τότε είναι μία ψηφιακή εικόνα. Είναι φανερό λοιπόν ότι μια ψηφιακή εικόνα αποτελείται από ένα πεπερασμένο πλήθος στοιχείων, το καθένα με συγκεκριμένες συντεταγμένες και τιμή. Τα στοιχεία αυτά όπως αναφέρθηκε και πιο πάνω ονομάζονται Pixels.

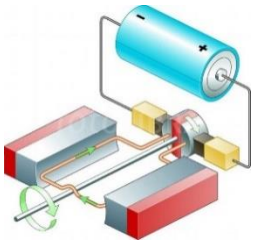
Τα βίντεο επίσης, μπορούν να οριστούν και ως σήματα τριών διαστάσεων. Απλά προστίθεται στη συνάρτηση μίας εικόνας f(x,y) μία ακόμα παράμετρος, ο χρόνος t. Ας θεωρήσουμε λοιπόν ότι έχουμε το βίντεο F(x,y,t). Η τιμή της F είναι η φωτεινότητα του εικονοστοιχείου με χωρικές συντεταγμένες (x,y) στο καρέ t του βίντεο. Τα x και y παίρνουν τιμές, από 0 έως το πλάτος και ύψος αντίστοιχα του κάθε καρέ του βίντεο, ενώ ο χρόνος t παίρνει διακριτές τιμές από 0, που αποτελεί το πρώτο καρέ, έως T που αποτελεί το τελικό.

Αν η παράμετρος t είναι σταθερή και ίση με t0, αναφερόμαστε σε ένα συγκεκριμένο καρέ. Έτσι το σήμα F(x,y,t0) μπορούμε να θεωρήσουμε ότι αποτελεί ένα δισδιάστατο σήμα G0(x,y). Αυτό αναπαριστά το συγκεκριμένο καρέ με αριθμό t0, δηλαδή μία στατική εικόνα. Έτσι μπορούμε να εφαρμόσουμε πάνω στο G0 κάθε γνωστό αλγόριθμο από την επεξεργασία εικόνας. Αυτή είναι γενικά από μαθηματική πλευρά η διαδικασία με την οποία προχωράμε στην ψηφιακή επεξεργασία του βίντεο βασιζόμενοι σε τεχνικές της ψηφιακής επεξεργασίας ει



Εικόνα 1.5: Αναπαράσταση καρέ βίντεο

1.4 Ηλεκτροκινητήρες



Ο ηλεκτρικός κινητήρας ή ηλεκτροκινητήρας, είναι διάταξη που χρησιμοποιείται για την μετατροπή της ηλεκτρικής ενέργειας σε μηχανική. Η αρχή λειτουργίας του ηλεκτρικού κινητήρα είναι η δύναμη [1] Laplace. Όταν ένας αγωγός από τον οποίο διαρρέει ηλεκτρικό ρεύμα βρεθεί μέσα σε ένα μαγνητικό πεδίο ασκείται πάνω του δύναμη ίση με:

$$\mathbf{F = I * \lambda * B * \eta\mu\varphi}$$

Όπου: • I = Ένταση Ρεύματος

• λ = Μήκος Αγωγού

• B = Ένταση Μαγνητικού πεδίου

• φ = η γωνία που σχηματίζει ο αγωγός με τη διεύθυνση των δυναμικών γραμμών (B)

Οι ηλεκτρικοί κινητήρες αποτελούνται από:

1. **Τον Δρομέα.**

Ο **Δρομέας** αποτελείται από τον ηλεκτροφόρο αγωγό ο οποίος είναι τοποθετημένος σε πυκνές περιελίξεις (σπείρες) ώστε να περιέχει όσο μεγαλύτερο μήκος αγωγού γίνεται για δεδομένο όγκο.

2. **Τον Στάτη.**

Ο **Στάτης** αποτελείται από μόνιμους ή τεχνητούς μαγνήτες οι οποίοι δημιουργούν το μαγνητικό πεδίο.

3. **Τις Ψήκτρες.**

Οι **Ψήκτρες** έρχονται σε επαφή με τον δρομέα τροφοδοτώντας τον με ρεύμα.

Τα απαραίτητα στοιχεία για κάθε ηλεκτροκινητήρα τα οποία και τον προσδιορίζουν εμπορικά είναι:

a) Η απαιτούμενη τάση για την τροφοδοσία του σε βολτ (V),
b) Το είδος της απαιτούμενης τάσης, συνεχές ή εναλλασσόμενο ρεύμα (DC ή AC) και στη 2η περίπτωση, μονοφασικό (1PH) ή τριφασικό (3PH),
c) Η συχνότητα του εναλλασσόμενου ρεύματος, εφόσον πρόκειται για ηλεκτροκινητήρα AC και προφανώς σε κύκλους ανά δευτερόλεπτο κ/δ (c/s) ή Hertz. Πολλές φορές χρησιμοποιείται το σύμβολο (~) αντί του κ/δ.

d) Η ισχύς του κινητήρα σε Βατ ή ίππους (W ή HP)

e) Η ένταση του ρεύματος σε αμπέρ που διαρρέει τον κινητήρα, και

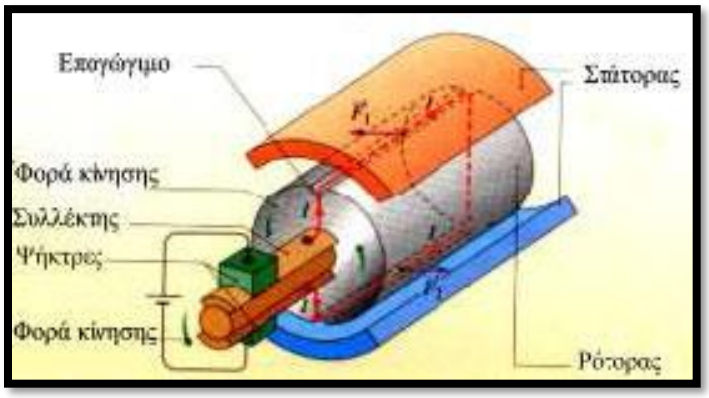
f) Η αποκτώμενη ταχύτητα περιστροφής του άξονα του κινητήρα σε στροφές ανά λεπτό (RPM).

Όλα τα παραπάνω στοιχεία φέρονται χαραγμένα, από τους κατασκευαστές, σε ειδική ενσωματωμένη στον ηλεκτροκινητήρα πινακίδα, καθώς και ο αριθμός της έγκρισης του Υπουργείου Βιομηχανίας για εμπορική διάθεση ή άλλα σύμβολα πιστοποίησης ασφαλούς λειτουργίας.

Οι ηλεκτροκινητήρες διακρίνονται σε "συνεχούς ρεύματος" (DC motors) και σε "εναλλασσόμενου ρεύματος" (AC motors).

Η χρήση των ηλεκτροκινητήρων σήμερα είναι ευρύτατα διαδεδομένη, ξεκινώντας από τις οικιακές συσκευές (ψυγείο, κλιματιστικό, μπλέντερ) συνεχίζοντας στις βιομηχανικές (τόρνοι, ανυψωτικές διατάξεις, αντλίες) και καταλήγοντας στις μετακινήσεις (ηλεκτρικοί σιδηρόδρομοι, υβριδικά αυτοκίνητα) μπορούμε εύκολα να κατανοήσουμε ότι είναι άρρηκτα συνδεδεμένες με την καθημερινή ζωή, σε πολλές περιπτώσεις ακόμα και σε βιοτικής σημασίας πρωτογενείς ανάγκες (τεχνητή καρδιά, ρομποτικοί βραχίονες νευροχειρουργικής, αντλίες

συσκευών αιμοκάθαρσης, αυτοκινούμενες αναπηρικές καρέκλες). Οι δε χρησιμοποιούμενοι στις βιομηχανίες είναι εκατοντάδων ίππων.



Εικόνα 1.6: Περιγραφή Ηλεκτροκινητήρα

1.4.1 Χαρακτηριστικά Ηλεκτρικών Κινητήρων

- **Τάση** λειτουργίας :

Μετράτε σε Volts και είναι η τάση που πρέπει να έχει το ηλεκτρικό ρεύμα τροφοδοσίας ώστε ο κινητήρας να λειτουργεί σωστά.

- **Ένταση** του ρεύματος :

Μετράτε σε Amps και είναι η ένταση του ρεύματος που διαρρέει τον κινητήρα όταν αυτός λειτουργεί. Η ένταση αυτή είναι ανάλογη του φορτίου του κινητήρα. Η ελάχιστη τιμή της αντιστοιχεί στην ελεύθερη περιστροφή του κινητήρα. Εάν ο κινητήρας έχει φορτίο η τιμή αυτή

αυξάνεται. Για κάποια τιμή του φορτίου ο κινητήρας σταματά να περιστρέφεται, λόγω της μεγάλης αντίστασης, οπότε και η τιμή της έντασης μεγιστοποιείται. Η μέγιστη αυτή τιμή είναι ένα από τα σημαντικά χαρακτηριστικά του κινητήρα που πρέπει να είναι γνωστό για τη σωστή επιλογή της τροφοδοσίας του.

- **Ταχύτητα** :

Στους σέρβο κινητήρες μετράτε σε στροφές ανά λεπτό και στους βηματικούς κινητήρες σε βήματα ανά δευτερόλεπτο. Πρόκειται για την ταχύτητα περιστροφής του άξονα του κινητήρα, όταν ο κινητήρας λειτουργεί υπό κανονική ηλεκτρική τάση και με δεδομένο φορτίο.

- **Ροπή** :

Μετράτε σε Nm (Newton meters) και είναι η ροπή που παράγει ο κινητήρας στις διάφορες ταχύτητες περιστροφής του. Η μέγιστη τιμή ροπής κάθε κινητήρα, που ονομάζεται ροπή ακινητοποίησης, είναι η ροπή που παράγει όταν το φορτίο που αντιμετωπίζει είναι τόσο μεγάλο, ώστε να τον ακινητοποιεί.

1.4.2 Ηλεκτρικοί Βηματικοί Κινητήρες (Stepper Motors)

Ο βηματίκος κινητήρας είναι ένας σταθερός μετατροπέας [2] παραγωγής δύναμης, όπου η δύναμη ορίζεται ως το γινόμενο της ροπής με την ταχύτητα. Αυτό σημαίνει ότι η ροπή των μηχανών είναι αντίστροφη της ταχύτητας. Για να βοηθήσουμε να καταλάβουμε γιατί η δύναμη ενός βηματίκου κινητήρα είναι ανεξάρτητη από την ταχύτητα, πρέπει να κατασκευάσουμε (μεταφορικά) έναν ιδανικό βηματικό κινητήρα.

Ένας ιδανικός βηματίκος κινητήρας θα είχε μη μηχανική τριβή, η ροπή του θα ήταν ανάλογη προς τις αμπέρ-στροφές και το μόνο ηλεκτρικό χαρακτηριστικό του θα ήταν η αυτεπαγωγή. Οι αμπέρ- στροφές απλά σημαίνουν ότι η ροπή είναι ανάλογη προς τον αριθμό στροφών του καλωδίου στο Στάτη της μηχανής πολλαπλασιαζόμενο με το ρεύμα που περνά μέσω εκείνων των στροφών του καλωδίου.

Οποτεδήποτε υπάρχουν στροφές ενός καλωδίου γύρο από ένα μαγνητικό υλικό όπως ο σίδηρος στο Στάτη της μηχανής, τότε αποκτάει μια ηλεκτρική ιδιότητα, την αυτεπαγωγή. Η αυτεπαγωγή περιγράφει την ενέργεια που αποθηκεύεται σε ένα μαγνητικό πεδίο όταν περάσει ρεύμα μέσω μιας σπείρας αυτού του καλωδίου.

Η αυτεπαγωγή (L) έχει μια ιδιότητα που ονομάζετε επαγωγική άεργη αντίσταση, η οποία για τους σκοπούς αυτής της συζήτησης μπορεί να θεωρηθεί ως αντίσταση ανάλογη προς τη συχνότητα και επομένως της ταχύτητας του κινητήρα.

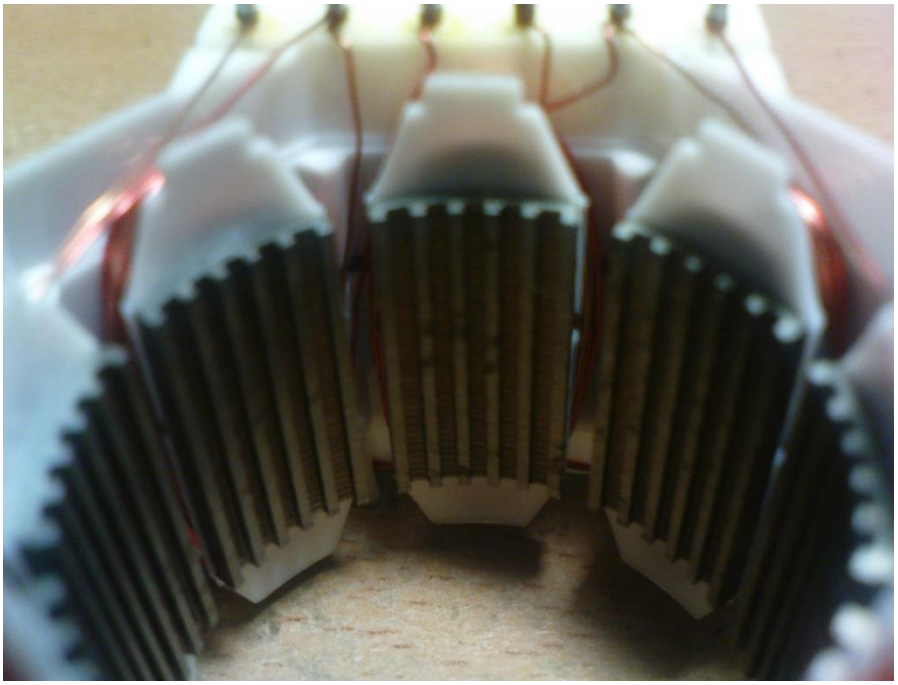
Σύμφωνα με το νόμο του Ohm, το ρεύμα είναι ίσο με την τάση που διαρρέει μια αντίσταση προς την αντίσταση αυτή. Σε αυτήν την περίπτωση αντικαθιστούμε την επαγωγική άεργη αντίσταση με την αντίσταση στο νόμο του Ohm και καταλήγουμε πως το ρεύμα των κινητήρων είναι αντίστροφο της ταχύτητας τους. Δεδομένου ότι η ροπή είναι ανάλογη προς τις αμπέρ-στροφές (ρεύμα επί των αριθμό στροφών του καλωδίου στο τύλιγμα), και το ρεύμα είναι αντίστροφο της ταχύτητας, η ροπή πρέπει επίσης να είναι αντίστροφη της ταχύτητας.

Σε έναν ιδανικό βηματικό κινητήρα , όσο η ταχύτητα προσεγγίζει το μηδέν τόσο η ροπή θα προσέγγιζε το άπειρο ενώ με άπειρη ταχύτητα η ροπή θα ήταν μηδέν. Επειδή το ρεύμα είναι ανάλογο προς τη ροπή, το ρεύμα θα ήταν επίσης άπειρο σε μηδενική ταχύτητα.

Ηλεκτρικά, μια πραγματική μηχανή διαφέρει από ένα ιδανικό κινητήρα καταρχήν από την ύπαρξη μιας διάφορης του μηδενός αντίστασης τυλίγματος. Επίσης ο σίδηρος στη μηχανή υπόκειται στο μαγνητικό κορεσμό, καθώς επίσης και απώλεια ρεύματος στον στροβιλισμό και λόγω υστέρησης.

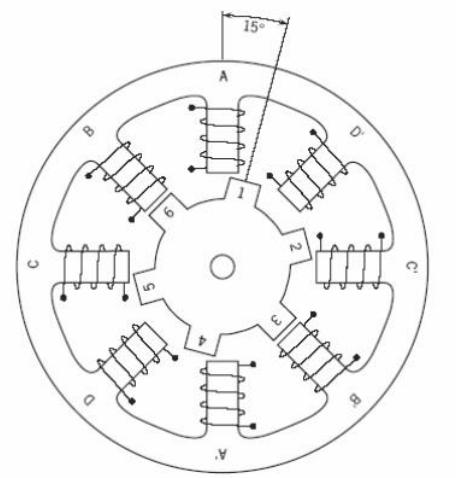
Ο μαγνητικός κορεσμός θέτει ένα όριο στο ρεύμα και στην αναλογικότητα της ροπής ενώ ο στροβιλισμός και η υστέρηση (απώλειες σιδήρου) μαζί με την αντίσταση του τυλίγματος (απώλειες χαλκού) προκαλούν έκλυση θερμότητας.

Εικόνα 1.7: Μέρη Βηματικού Κινητήρα

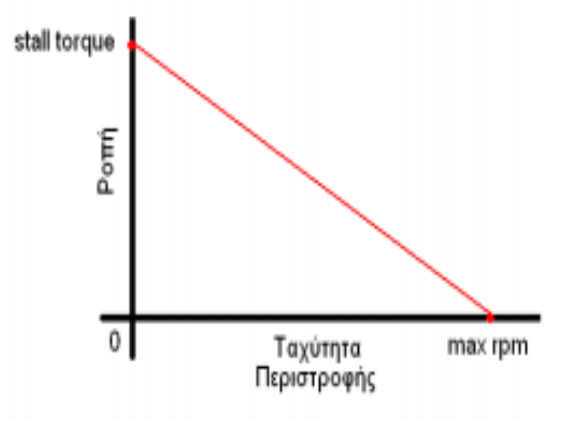


Εικόνα 1.8: Στάτης Βηματικού Κινητήρα

Ο κινητήρας αυτός αποτελείται από έναν ρότορα μαλακού σιδήρου με οδοντώσεις και έναν στάτορα με τέσσερα ζεύγη ηλεκτρομαγνητών: A και A΄, B και B΄, C και C΄, D και D΄. Για να κινηθεί ο ρότορας εφαρμόζεται διαδοχικά σε κάθε ένα από τα παραπάνω ζεύγη μια τάση ηλεκτρικού ρεύματος. Όταν το ηλεκτρικό ρεύμα μεταφέρεται από το ένα ζεύγος ηλεκτρομαγνητών στο διπλανό του ο ρότορας μετατοπίζεται κατά 15 μοίρες λόγω των μαγνητικών δυνάμεων που εφαρμόζονται σε αυτόν. Η γωνία αυτή ονομάζεται **βήμα** του κινητήρα.



Εικόνα 1.9: Διατομή Βηματικού Κινητήρα



Εικόνα 1.10: Σχέση Ταχύτητας-Ροπής

Πλεονεκτήματα Βηματικού Κινητήρα

1. Σε αντίθεση με τους κινητήρες συνεχούς ρεύματος, δεν χρειάζεται φρένα για να μένει ακίνητος ή για να επιβραδυνθεί.
2. Στις μικρές ταχύτητες περιστροφής, αλλά και κατά την εκκίνησή του, παράγει μεγάλες τιμές ροπής.
3. Είναι πολύ αξιόπιστος καθώς για τη λειτουργία του δεν απαιτούνται κινούμενες ηλεκτρικές επαφές, όπως στον κινητήρα συνεχούς ρεύματος και έτσι η διάρκεια ζωής του εξαρτάται μόνο από την αξιοπιστία του εδράνου κύλισης.

4. Δεν απαιτείται χρήση αισθητήρων και κυκλωμάτων ανάδρασης για τον προσδιορισμό της θέσης του άξονα κίνησης.
5. Ο βηματικός κινητήρας μπορεί να επιτύχει μεγάλο εύρος ταχυτήτων περιστροφής.
6. Ο βηματικό κινητήρας μπορεί να επιτύχει πολύ χαμηλές ταχύτητες περιστροφής.

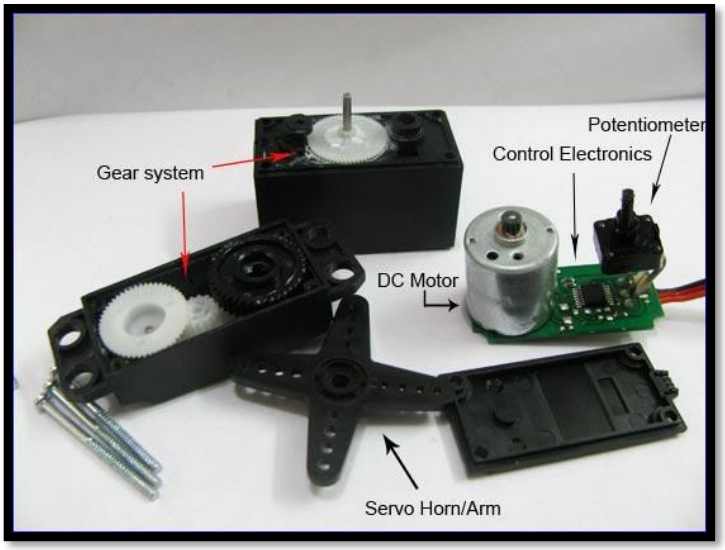
Μειονεκτήματα Βηματικού Κινητήρα

1. Θορυβώδης λειτουργία.
2. Αδυναμία περιστροφής σε υψηλές ταχύτητες.
3. Κατά τη μετακίνηση φορτίων μεγάλης μάζας μπορεί να μη σταματήσει ακαριαία ο κινητήρας, λόγω της αυξημένης αδράνειας.

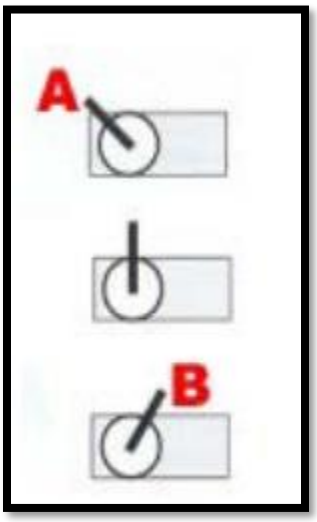
1.4.3 Ηλεκτρικοί Κινητήρες Servo (R/C Motors)

Το σέρβο είναι συσκευή που αποτελείται από έναν ηλεκτροκινητήρα συνεχούς ρεύματος, ένα ηλεκτρονικό κύκλωμα που ελέγχει τη θέση του τελικού άξονα κίνησης και ένα κιβώτιο υποβιβασμού της σχέσης μετάδοσης του κινητήρα. Ο τελικός άξονας κίνησης δεν εκτελεί πλήρεις περιστροφές, αλλά περιστρέφεται μεταξύ δύο ακραίων θέσεων Α, Β.

Για τη λειτουργία του σέρβο απαιτείται η παροχή της κατάλληλης ηλεκτρικής τάσης αλλά και ενός σήματος ελέγχου που καθορίζει τη θέση περιστροφής του τελικού άξονα.



Εικόνα 1.11: Εξαρτήματα ενός Servo Motor



Εικόνα 1.12: Κίνηση του Servo

Τα σέρβο χρησιμοποιούνται σε τηλεκατευθυνόμενα μοντέλα αλλά και σε πολλές ρομποτικές εφαρμογές.

Πλεονεκτήματα

• Χαμηλό κόστος

• Μικρές διαστάσεις και εύχρηστο σχήμα: όλα τα τμήματά ενός σέρβο περιβάλλονται από ένα συμπαγές περίβλημα από το οποίο εξέχει μόνο ο τελικός άξονας κίνησης

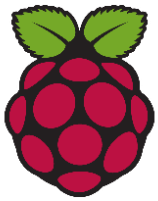
• Παράγουν υψηλές τιμές ροπής

• Δεν απαιτείται χρήση αισθητήρων και κυκλωμάτων ανάδρασης για τον προσδιορισμό της θέσης του άξονα κίνησης.

Μειονεκτήματα

• Αδυναμία εκτέλεσης πλήρους και συνεχούς περιστροφής

ΚΕΦΑΛΑΙΟ 2



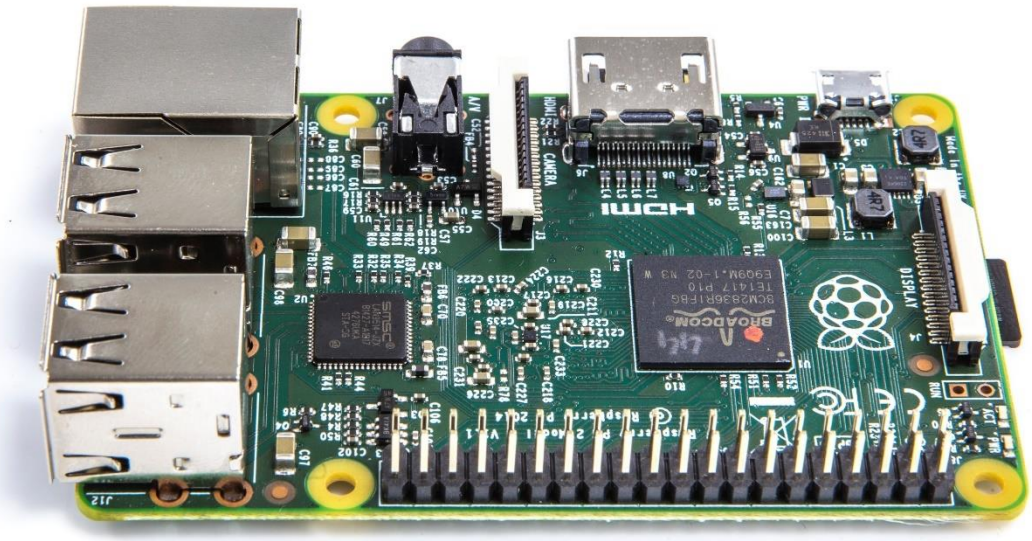
RASPBERRY PI

2.1 Εισαγωγή

Μέχρι τη δεκαετία του 1970, οι υπολογιστές αποτελούσαν αποκλειστικό προνόμιο ενδοπανεπιστημιακών ερευνητικών εργαστηρίων καθώς και συγκεκριμένων μεγάλων εταιριών. Από το 1970 όμως, και μετά η κατάσταση αυτή άλλαξε άρδην. Η αλλαγή αυτή υποκινήθηκε κυρίως από την κυκλοφορία της TVT (TV Typewriter) και του Apple I και στη συνέχεια του Apple II. Έτσι, η δεκαετία του 1980 έφερε χαμηλού κόστους προσωπικούς υπολογιστές για οικιακή χρήση όπως ο ZX Spectrum και ο Commodore 64 και ταυτόχρονη ανάπτυξη στον κλάδο των προγραμματιστών. Μέχρι το 1990, ο προγραμματισμός είχε ήδη γίνει αναπόσπαστο κομμάτι της επιστήμης των υπολογιστών τόσο σε ακαδημαϊκό όσο και σε βιομηχανικό επίπεδο, γεγονός που οδήγησε και στη διαδικτυακή επανάσταση που συντελέστηκε την εποχή εκείνη.

Εμπνευσμένη από αυτή την πορεία της επιστήμης του προγραμματισμού και θέλοντας να παρέχει έναν οικονομικό μικρο-υπολογιστή με δυνατότητα να υποστηρίξει ελαφριά λειτουργικά συστήματα στους ερασιτέχνες προγραμματιστές, τους φοιτητές και τα παιδιά, μια ομάδα επιστημόνων των υπολογιστών υπό την καθοδήγηση του Eben Upton στο Εργαστήριο Υπολογιστών του Πανεπιστημίου του Cambridge ξεκίνησε το 2006 την προσπάθεια κατασκευής του Raspberry Pi. Το 2008, το εγχείρημα αυτό άρχισε να παίρνει σάρκα και οστά με αποτέλεσμα το λανσάρισμα του Raspberry Pi την 29η Φλεβάρη του 2012. Σημαντική βοήθεια στην επιτυχία αυτής της προσπάθειας προσέφερε η φοβερή ανάπτυξη που είχε την εποχή εκείνη η κινητή τηλεφωνία, αφού χρησιμοποιήθηκαν μονάδες CPU της βρετανικής εταιρίας ARM που αρχικά προορίζονταν για κινητά τηλέφωνα. Αυτό είχε σημαντική επίπτωση στη μείωση της τιμής του Raspberry Pi αφού μπορούσε να χρησιμοποιήσει ήδη υπάρχουσα τεχνογνωσία. Έτσι, ιδρύθηκε και επίσημα το «Ίδρυμα Raspberry Pi» το οποίο έχει σαν κύριο στόχο την κατασκευή και βελτίωση των Raspberry Pi υπολογιστών.

Το πρώτο μοντέλο Raspberry Pi, το Model A, κατασκευάστηκε το 2011 και δόθηκε στο κοινό για να γνωρίσει αυτό το νέο είδος μικροϋπολογιστή και να τροφοδοτήσει την κατασκευάστρια εταιρία με πολύτιμη ανάδραση. Οι δοκιμές αυτές κατέληξαν στο συμπέρασμα πως επρόκειτο για έναν μικροσκοπικό υπολογιστή ο οποίος μπορούσε να προσφέρει πάρα πολλά σε σύγκριση με το πολύ μικρό του κόστος. Τελικά, το 2012 ο Raspberry Pi ήταν έτοιμος για την πρώτη έξοδο του στο εμπόριο για ευρεία κατανάλωση με τη μορφή δύο μοντέλων, το A model και το B model. Τα δύο μοντέλα διέφεραν μεταξύ τους μόνο στο ότι το A δε διέθετε θύρα Ethernet, διαφοροποίηση που μεταφράστηκε σε μια διαφορά στην τιμή του ύψους των $10. Η κατασκευή τους ξεκίνησε στην Κίνα ενώ συνεχίστηκε επί βρετανικού εδάφους όταν τη διαδικασία ανέλαβε η Sony. Μετά από αρκετές βελτιώσεις και συμμορφώσεις σε κανονισμούς, ο Raspberry Pi είναι πλέον στη διάθεση του καταναλωτικού κοινού και απολαμβάνει σημαντικής αποδοχής.

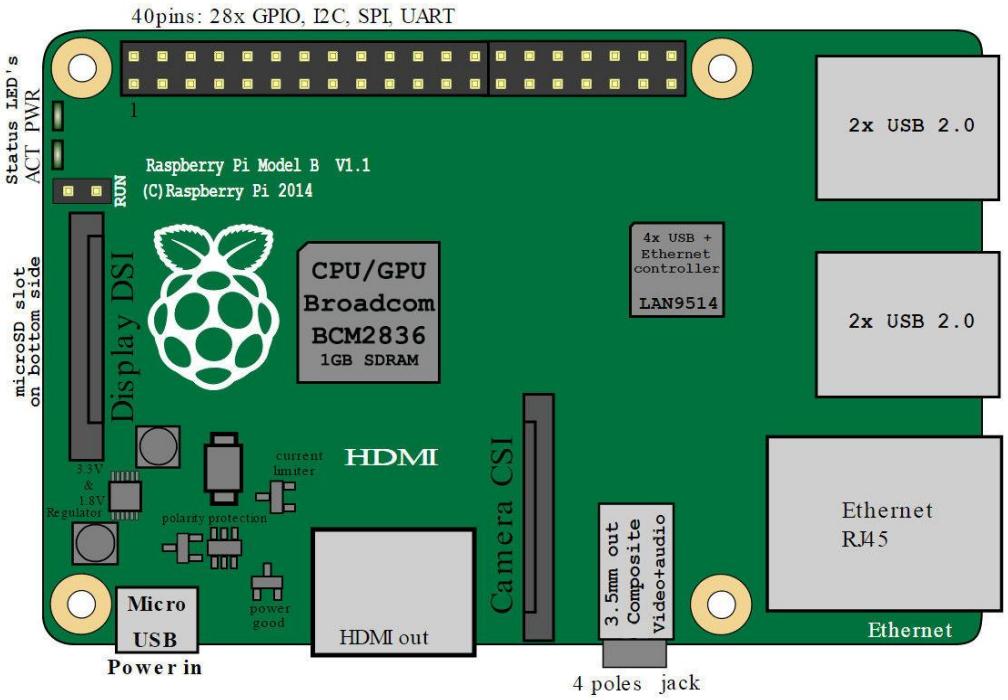


Εικόνα 2.1: Το model B του Raspberry Pi 2

2.2 Το Hardware του Raspberry Pi

Η κατασκευή του Raspberry Pi έχει βασιστεί στον Broadcom BCM2835, έναν επεξεργαστή για εφαρμογές πολυμέσων που προορίζεται για κινητές συσκευές. Πάνω σε αυτόν έχουν προστεθεί διάφορα άλλα στοιχεία ώστε να στηριχθούν τεχνολογίες όπως η τοποθέτηση κάρτας SD, η ύπαρξη θύρας USB, θύρας RCA ή HDMI.

Ο υπολογιστής Raspberry Pi έχει την παρακάτω μορφή:



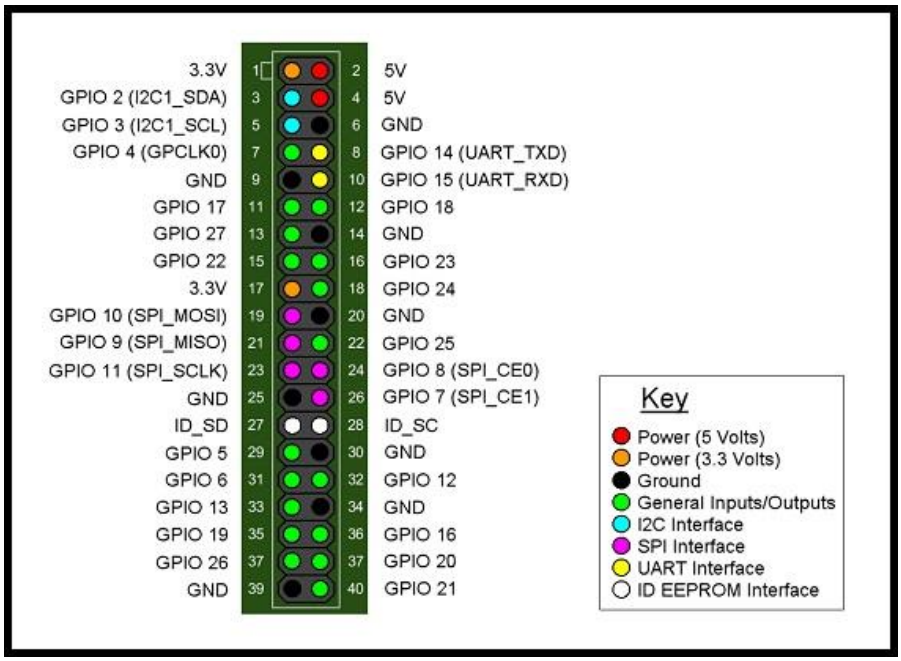
Εικόνα 2.2: Σχηματική αναπαράσταση του υπολογιστή Raspberry Pi

Οι διαστάσεις του είναι αρκετά μικρές, 85.60mm x 53.98mm x 17mm με βάρος μόλις 45gr. Οι διαστάσεις του αυτές είναι που τον καθιστούν ιδανικό στοιχείο συστημάτων αυτοματισμού μικρών εγκαταστάσεων αφού στις συγκεκριμένες εφαρμογές χρειαζόμαστε πολύ μικρές διατάξεις οι οποίες θα τοποθετηθούν εντός ήδη υπάρχουσας συσκευής ή διάταξης.

Ο υπολογιστής Raspberry Pi διαθέτει θύρες εισόδου/εξόδου για διαφόρων τύπων επικοινωνίες του συστήματος έτσι ώστε να είναι δυνατή η υλοποίηση διαφορετικών ειδών εφαρμογών. Διαθέτει αναλογική θύρα ήχου 3.5mm για τη χρησιμοποίηση ακουστικών και μικροφώνων, ιδιαίτερα χρήσιμη για τις εφαρμογές πολυμέσων. Επίσης, διαθέτει θύρα RCA έτσι ώστε να μπορεί να χρησιμοποιηθεί η τηλεόραση ως monitor. Αναφορικά με τη USB συνδεσιμότητα του, ο Raspberry Pi διαθέτει από δύο ως τέσσερις θύρες USB 2.0 και μια micro USB οι οποίες επιτρέπουν τη σύνδεση πληκτρολογίου και ποντικιού και σύνδεση με πηγή ηλεκτρικού ρεύματος αντίστοιχα. Υπάρχει ακόμα μια θύρα HDMI (High Definition Multimedia Interface) η οποία καθιστά δυνατή τη σύνδεση του συστήματος με τηλεοράσεις και monitor υψηλής ευκρίνειας, Αυτή η θύρα χρησιμοποιείτε όταν επιθυμείτε η αναπαραγωγή πολυμέσων με ήχο και εικόνα ταυτόχρονα. Η επικοινωνία του υπολογιστή με άλλες συσκευές καθώς και η σύνδεση του στο διαδίκτυο γίνονται μέσω μιας θύρας Ethernet. Μέσω αυτής ο Raspberry Pi συνδέεται με τον κοινό οικιακό δρομολογητή (Router) που χρησιμοποιείται για τη σύνδεση οποιουδήποτε προσωπικού υπολογιστή στο διαδίκτυο. Ωστόσο, θύρα Ethernet διαθέτει μόνο το Model B, το Model A συνδέεται στο διαδίκτυο μέσω μιας εξωτερικής θύρας Ethernet συνδεδεμένης σε μια από τις USB θύρες είτε με προσαρμογέα Wi-Fi πάλι συνδεδεμένο σε κάποια από τις θύρες USB. Ο κύριος τρόπος αποθήκευσης δεδομένων του υπολογιστή Raspberry Pi είναι μέσω χρήσης κάρτας SD ή οποία τοποθετείται στην κατάλληλη SD θύρα. Αυτός ακριβώς ο αποθηκευτικός χώρος είναι το σημείο όπου θα εγκατασταθεί το λειτουργικό σύστημα του υπολογιστή και κατ' επέκταση θα λειτουργεί ως ο σκληρός του δίσκος. Επειδή όμως μια κάρτα SD προσφέρει περιορισμένο αποθηκευτικό χώρο, υπάρχει πάντα η δυνατότητα επέκτασης αυτού μέσω εξωτερικού σκληρού δίσκου συνδεδεμένου με το σύστημα μέσω των θυρών USB. Τα προηγούμενα μοντέλα Raspberry Pi είναι εφοδιασμένα με προ εγκατεστημένη μνήμη SDRAM 256MB ενώ τα πιο σύγχρονα μοντέλα με SDRAM 512MB και 1 GB. Αυτή η προ εγκατεστημένη μνήμη συγκρινόμενη με τη μνήμη που διαθέτει ένας σύγχρονος προσωπικός υπολογιστής είναι σχετικά μικρή. Ωστόσο, για το εύρος των

εφαρμογών για τις οποίες προορίζεται ένας Raspberry Pi υπολογιστής τόσο η μνήμη των 512MB όσο η μνήμη του 1GB θεωρούνται κάτι παραπάνω από αρκετές. Αξίζει να σημειωθεί, βέβαια, πως στις 15 Οκτώβρη του 2012, το ίδρυμα Raspberry Pi ανακοίνωσε πως όλα τα μοντέλα του πλέον θα κατασκευάζονται με μνήμη τουλάχιστον 512MB.

Όπως προαναφέρθηκε ο κεντρικός επεξεργαστής (CPU) ενός υπολογιστή Raspberry Pi είναι βρετανικής κατασκευής και πιο συγκεκριμένα είναι το μοντέλο ARM1176JZF-S - Core στα 700 Mhz της σειράς 32 bit πολυεπεξεργαστών ARM 11 της εταιρίας ARM. Ο CPU είναι υπεύθυνος για την εκτέλεση των εντολών κάθε προγράμματος που φέρει ο Raspberry Pi μέσω μαθηματικών και λογικών εντολών. Τέλος, έχει ενσωματωθεί η κάρτα γραφικών Broadcom VideoCore IV, η οποία είναι ειδικά σχεδιασμένη ώστε να επιταχύνεται η διαχείριση εικόνων.Η δυνατότητα αυτή είναι ιδιαίτερα σημαντική για τις εφαρμογές 3D γραφικών, η για χρησιμοποίηση του υπολογιστή ως κονσόλα παιχνιδιών.



Εικόνα 2.3: Σχηματική αναπαράσταση των GPIO Pins

Συγκεντρωτικά για τις προδιαγραφές του υπολογιστή Raspberry Pi ανά μοντέλο μπορούμε να δούμε τον παρακάτω πίνακα:

	Raspberry Pi 3 Model B	Raspberry Pi 2 Model B	Raspberry Pi Model B+
Processor Chipset	**Broadcom BCM2837 64Bit Quad Core ARM Cortex A53 at 1.2GHz**	Broadcom BCM2836 32Bit Quad Core ARMv7 at 900MHz	Broadcom BCM2835 32Bit ARMv6k at 700MHz
GPU	**Videocore IV @ 400MHz**	Videocore IV @ 250MHz	Videocore IV @ 250MHz
Processor Speed	**QUAD Core @1.2 GHz**	QUAD Core @900 MHz	Single Core @700 MHz
RAM	1GB SDRAM @ 400 MHz	1GB SDRAM @ 400 MHz	512 MB SDRAM @ 400 MHz
Storage	MicroSD	MicroSD	MicroSD
USB 2.0	4x USB Ports	4x USB Ports	4x USB Ports
Max Power Draw/voltage	2.5A @ 5V	1.8A @ 5V	1.8A @ 5V
GPIO	40 pin	40 pin	40 pin
Ethernet Port	Yes	Yes	Yes
WiFi	**Built in**	No	No
Bluetooth LE	**Built in**	No	No
Video Output	HDMI/Composite via RCA Jack	HDMI/Composite via RCA Jack	HDMI/Composite via RCA Jack
Audio Output	3.5mm Jack	3.5mm Jack	3.5mm Jack

Εικόνα 2.4: Προδιαγραφές Raspberry Pi ανά μοντέλο

2.3 Το Software του Raspberry Pi

Κατά το σχεδιασμό του υπολογιστή Raspberry Pi στοχεύθηκε η δημιουργία ενός συστήματος το οποίο θα δύναται να λειτουργεί με πολύ ελαφριά λειτουργικά συστήματα. Έτσι, ο υπολογιστής αυτός μπορεί να χρησιμοποιεί κατά κύριο λόγο λειτουργικά συστήματα βασισμένα στον πυρήνα του Linux.

Το ίδρυμα Raspberry Pi δημιούργησε έναν ιδιαίτερα εύχρηστο installer προγραμμάτων, τον NOOBS (New Out Of Box System), ο οποίος διευκολύνει ιδιαίτερα την εγκατάσταση λειτουργικών συστημάτων και από τον Ιούλιο του 2013 συνιστά τη χρήση αυτού για την εγκατάσταση Raspbian στα υπολογιστικά του συστήματα. Με αυτή τη διαδικασία, το ίδρυμα προσπαθεί να ξεκινήσει την ανάπτυξη ενός καταστήματος εφαρμογών όπου οι χρήστες θα μπορούν να ανταλλάσσουν προγράμματα.

Το Raspbian είναι ένα λειτουργικό σύστημα βασισμένο σε βελτιστοποίηση Debian ώστε να μπορεί να υποστηριχθεί από έναν υπολογιστή Raspberry Pi. Είναι το λειτουργικό σύστημα το οποίο -προς το παρόν- συνιστά το ίδρυμα και λανσαρίστηκε επίσημα τον Ιούλιο του 2012 παρά το γεγονός ότι ήταν ακόμα υπό ανάπτυξη. Είναι ελεύθερο λογισμικό του οποίου η ανάπτυξη γίνεται εξολοκλήρου από το ίδρυμα και βασίζεται στην αρχιτεκτονική του Debian 7 σε περιβάλλον LXDE. Το Raspbian διαθέτει μερικά deb πακέτα προγραμμάτων και απαιτεί μια ελάχιστη μνήμη SD κάρτας των 2GB. Ωστόσο, συνίσταται η χρήση SD κάρτας από 4GB και πάω.

Εκτός, όμως από το Raspbian, ο NOOBS εγκαθιστά μερικά ακόμα λειτουργικά συστήματα, ανάλογα με την προτίμηση του χρήστη. Αυτά είναι το Archlinux ARM, το OpenELEC, το Pidora (Λειτουργικό Σύστημα βασισμένο στο Fedora) , το Raspbmc και το RISC OS. Ωστόσο, το ίδρυμα συνιστά τη χρήση Raspbian.

Επίσης, υπάρχει διαθέσιμο λογισμικό και από άλλες εταιρείες που έχουν ασχοληθεί με τα συστήματα Raspberry Pi. Έτσι, διατίθεται το Raspbian Server Edition το οποίο είναι μια

έκδοση του Raspbian με ενσωματωμένα πακέτα λογισμικού. Το Slackware ARM είναι συμβατό με τον Raspberry Pi χωρίς καμία τροποποίηση. Το λογισμικό αυτό χρειάζεται μνήμη 64MB σε έναν ARM όπως αυτός που χρησιμοποιείται στον Raspberry Pi με αποτέλεσμα τα 128-496MB που είναι διαθέσιμα στη μνήμη του συγκεκριμένου υπολογιστή να είναι παραπάνω από αρκετά. Αυτή η εξοικονόμηση μνήμης προέρχεται από το γεγονός ότι, ενώ τα υπόλοιπα λειτουργικά συστήματα που βασίζονται σε πυρήνα Linux φορτώνονται μέσω διεπαφής με το χρήστη με απαιτήσεις από την κάρτα γραφικών, το Slackware ARM λειτουργεί στο περιβάλλον του ίδιου του φλοιού και σε παράθυρο εντολών. Επιπροσθέτως, αρκετές ομάδες εργάζονται πάνω σε ελαφριές εκδόσεις Linux όπως το IPFire, το OpenELEC, το Raspbmc καθώς και το ελεύθερου κώδικα κέντρο πολυμέσων XBMC.

Ιδιαίτερης σημασίας ήταν η ανακοίνωση που εξέδωσε το ίδρυμα στις 24 Οκτώβρη 2012 όπου το κοινό ενημερώθηκε πως εφεξής το συμβατό με ARM VideoCore λανσάρεται ως ελεύθερο λογισμικό με άδεια τύπου BSD. Έτσι, πλέον, το καταναλωτικό κοινό έχει στα χέρια του το πρώτο SoC πολυμέσων βασισμένο σε επεξεργαστή ARM με πλήρως λειτουργικό σύστημα ελεύθερου κώδικα

2.4 Εξαρτήματα για το Raspberry Pi

Το Μάϊο του 2012, το ίδρυμα Raspberry Pi ανακοίνωσε πως έχει ήδη δοκιμαστεί μια πρωτότυπη κάμερα για σύνδεση με τον υπολογιστή. Η κάμερα αυτή είχε ανάλυση 14 Megapixels, ωστόσο η κάμερα που θα λανσαριστεί αρχικά αναμένεται να είναι ανάλυσης 5 Megapixels και είναι κατασκευής των εταιρειών RS Components και Premier Farnell / Element 14. Συνδέεται μέσω επίπεδου καλωδίου στη θύρα CSI ανάμεσα στις Ethernet kai HDMI, υποστηρίζει βίντεο 1080p , 720p και 640 x 480p ενώ η τιμή της στην Ευρώπη είναι στα 20 Euro.

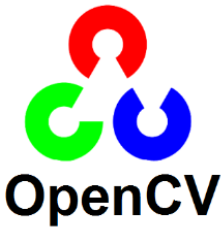
Εικόνα 2.5: Η πλακέτα Gertboard συνδεδεμένη με το Raspberry Pi

Επίσης, υπάρχει μεγάλη γκάμα περιφερειακών συσκευών και θηκών. Σε αυτές περιλαμβάνεται το Gertboard κατασκευής του ιδρύματος Raspberry Pi το οποίο έχει αναπτυχθεί για εκπαιδευτικούς σκοπούς και επεκτείνει τις δυνατότητες των GPIO Pins του υπολογιστή έτσι ώστε να επιτρέπεται η διασύνδεση με LEDS, διακόπτες, αισθητήρες και λοιπές συσκευές και τον πλήρη έλεγχο αυτών.

Εικόνα 2.6: Παράδειγμα σύνδεσης περιφερειακών συσκευών με Raspberry Pi

ΚΕΦΑΛΑΙΟ 3

OpenCV



3.1 Εισαγωγή

Η OpenCV είναι μια βιβλιοθήκη μηχανικής όρασης (computer vision) ανοικτού λογισμικού που αναπτύχθηκε αρχικά από την Intel και είναι ελεύθερη για εμπορική και ερευνητική χρήση υπό την άδεια ανοικτού λογισμικού BSD. Αυτό σημαίνει ότι μπορεί να γίνει ελεύθερη χρήση όλης η μέρους της βιβλιοθήκης για την ανάπτυξη ενός εμπορικού η ερευνητικού προϊόντος από τον οποιοδήποτε. Η βιβλιοθήκη είναι ανεξάρτητη πλατφόρμας και είναι γραμμένη σε optimized C. Αποτελείται από περισσότερες από 500 συναρτήσεις σε C και διάφορες κλάσεις σε C++. Έχει σχεδιαστεί με στόχο την αποτελεσματική εκμετάλλευση των υπολογιστικών πόρων και με έμφαση στις εφαρμογές πραγματικού χρόνου. Η Intel έχει επίσης αναπτύξει ένα σύνολο ρουτινών χαμηλού προγραμματιστικού επιπέδου, το Integraded Performance Primitives(IPP), το οποίο χρησιμοποιείται σε πολλές αλγοριθμικές περιοχές και αν είναι εγκατεστημένο η OpenCV.θα το χρησιμοποιήσει για να επιταχύνει τα προγράμματα που τη χρησιμοποιούν. Πρέπει να σημειωθεί ότι αυτό το σύνολο ρουτινών δεν είναι απαραίτητο για τη χρήση της OpenCV

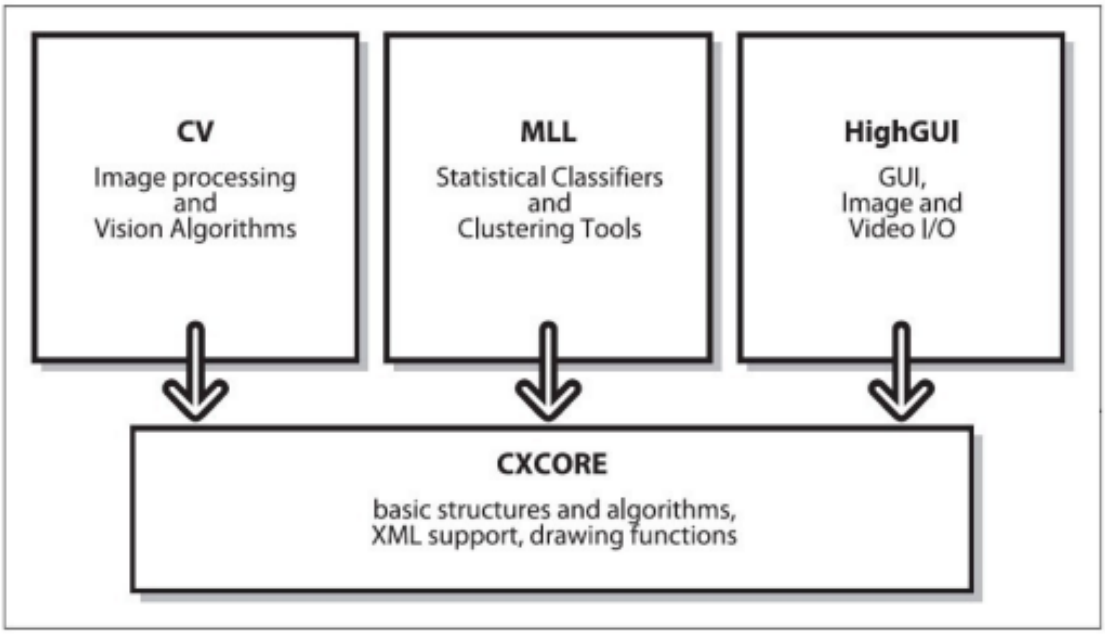
3.2 Υποστηριζόμενα Λειτουργικά Συστήματα και διαθέσιμες εκδόσεις

Η OpenCV είναι διαθέσιμη σε Linux, Mac OS και Windows. Η τελευταία της έκδοση, 3.2, μπορεί να βρεθεί στο Sourceforge και στο GitHub. Η πρώτη έκδοση ανακοινώθηκε το 2000 στο συνέδριο Computer Vision and Pattern Recognition και από τότε ενημερώνεται συνεχώς.Η βιβλιοθήκη OpenCV είναι γραμμένη στην γλώσσα προγραμματισμού C++.Έχουν κυκλοφορήσει διάφορες εκδόσεις της OpenCV και σε άλλες γλώσσες προγραμματισμού όπως

Python, Java και MATLAB.Στην παρούσα εργασία θα δούμε την ανάπτυξη της βιβλιοθήκης αυτής με χρήση της γλώσσας Python.

3.3 Τα δομικά στοιχεία της βιβλιοθήκης OpenCV

Η OpenCV περιέχει πάνω από 500 συναρτήσεις κατάλληλες για πολλές διαφορετικές εφαρμογές στους τομείς της οπτικής επεξεργασίας, όπως ιατρικές εφαρμογές που αφορούν εικόνα και βίντεο, κλειστό κύκλωμα παρακολούθησης, ασφάλεια, ρομποτική, ρύθμιση κάμερας, διεπαφή χρήστη και υπολογιστή. Επειδή υπάρχει ισχυρή συσχέτιση μεταξύ μηχανικής όρασης και μηχανικής εκμάθησης, η OpenCV περιέχει και μια βιβλιοθήκη μηχανικής εκμάθησης γενικού σκοπού (MLL - Machine Learning Library). Αυτή η ειδική βιβλιοθήκη εστιάζει στην αναγνώριση προτύπων με στατιστικές μεθόδους και στην ομαδοποίηση. Χρησιμοποιείται σε οποιοδήποτε πρόβλημα μηχανικής όρασης λόγω της γενικής μορφής της. Η OpenCV δομείται κυρίως από 5 συστατικά μέρη, 4 από τα οποία παρουσιάζονται στο σχήμα 4.1. Το πρώτο στο σχήμα με την ονομασία CV, αναπαριστά τους βασικούς αλγορίθμους για επεξεργασία εικόνας αλλά και τους υψηλού επιπέδου για τη μηχανική όραση. Το δεύτερο, ML, αναπαριστά τη βιβλιοθήκη μηχανικής μάθησης που περιλαμβάνει εργαλεία για ομαδοποίηση και στατιστική ταξινόμηση. Το τρίτο, HighGUI, αναπαριστά τις ρουτίνες εισόδου-εξόδου, τις συναρτήσεις για τη προβολή και την αποθήκευση των βίντεο και γενικά ό,τι έχει να κάνει με τη διεπαφή χρήστη-εφαρμογής. Τέλος, το CxCore περιέχει τις βασικές δομές δεδομένων όπως το IplImage, τη βασική δομή στην OpenCV που περιγράφει με λεπτομέρεια το κάθε καρέ προς επεξεργασία. Το τελευταίο δομικό μέρος της OpenCV είναι το CVAux που περιέχει αλγορίθμους που δε χρησιμοποιούνται πολύ γιατί είναι σε πειραματικό στάδιο ή που έχουν απλά εγκαταλειφθεί.



Εικόνα 3.1: Τα συστατικά μέρη της OpenCV

3.4 Τα πλεονεκτήματα της βιβλιοθήκης OpenCV

Το βασικότερο πλεονέκτημα της OpenCV είναι η ταχύτητα. Αυτός είναι και ο βασικός λόγος που δημιουργήθηκε εξ' αρχής. Η Intel ήθελε να δείξει πόσο προχωρημένοι ήταν οι επεξεργαστές τις που μπορούσαν να επεξεργάζονται βίντεο σε πραγματικό χρόνο. Το δεύτερο είναι ότι ενώ είναι ελεύθερο λογισμικό υποστηρίζεται από εταιρίες όπως η Intel, η IBM πανεπιστήμια όπως το Stanford, το MIT και πλήθος άλλων οργανισμών και εργαστηρίων που βοηθούν συνεχώς στην ανάπτυξή της. Ένα άλλο πλεονέκτημα είναι ότι δεν απαιτεί περίπλοκα μηχανήματα και εξοπλισμούς, αφού και η πιο απλή webcam είναι αρκετή για είσοδο εικόνας και ο πιο απλός προσωπικός υπολογιστής είναι ικανός για την επεξεργασία της.

3.5 Εφαρμογές της OpenCV

Οι επιστήμονες και οι ακαδημαϊκοί που ασχολούνται με τη μηχανική όραση γνωρίζουν πραγματικά τις δυνατότητες της και το ευρύ φάσμα στο οποίο χρησιμοποιείται. Οι περισσότεροι γνωρίζουν κάποια από τα πλεονεκτήματα της μηχανικής όρασης και λίγες εφαρμογές της. Οι πιο γνωστές εφαρμογές της είναι στα συστήματα παρακολούθησης και σε εικόνες και βίντεο στο διαδίκτυο. Η πραγματική δύναμή της φαίνεται σε περιοχές που δεν είναι ευρέως γνωστό ότι χρησιμοποιείται όπως σε διεπαφές παιχνιδιών και σε άλλες εντυπωσιακές όπως στους περισσότερους εναέριους χάρτες και χάρτες δρόμων (όπως στο Google Street View) όπου χρησιμοποιούν κατά κόρον τεχνικές συγκόλλησης εικόνων και ευθυγράμμισης και διόρθωσης της εικόνας από παραμορφώσεις που εισαγάγει ο φακός. Άλλες χρήσεις της είναι σε μη-επανδρωμένα οχήματα (γενικά στη ρομποτική) ή βιοϊατρικές αναλύσεις. Ένας σημαντικός κλάδος στον οποίο κερδίζει έδαφος είναι η βιομηχανία. Σχεδόν όλα τα προϊόντα που παράγονται μαζικά έχουν επιθεωρηθεί σε κάποιο στάδιο της παραγωγής από συστήματα μηχανικής όρασης. Από την πρώτη της έκδοση τον Ιανουάριο του 1999, η OpenCV έχει χρησιμοποιηθεί σε πολλές εφαρμογές, προϊόντα και ερευνητικές μελέτες. Αυτές οι εφαρμογές περιλαμβάνουν τον συγκερασμό εικόνων σε δορυφορικούς και διαδικτυακούς χάρτες, τη μείωση του θορύβου σε ιατρικές εικόνες, τα αυτόματα συστήματα ελέγχου και ασφαλείας, στρατιωτικές εφαρμογές και μη-επανδρωμένα οχήματα (εναέρια, υποβρύχια, επίγεια). Έχει χρησιμοποιηθεί ακόμα σε αναγνώριση ήχου και μουσικής. Η OpenCV ήταν ένα βασικό συστατικό του συστήματος όρασης του ρομπότ από το πανεπιστήμιο Stanford, του "Stanley", το οποίο κέρδισε το DARPA Grand Challenge αγώνα δρόμου ρομπότ στην έρημο.

Εικόνα 3.2: Παραγωγή πληροφορίας συγκριτικά με τον Άνθρωπο

3.6 Εγκατάσταση OpenCV σε Raspberry Pi

Η εγκατάσταση της βιβλιοθήκης OpenCV είναι μια χρονοβόρα διαδικασία καθώς χρειάζονται πολλές προϋποθέσεις και εξαρτήσεις από άλλες βιβλιοθήκες για να καταφέρουμε να γίνει επιτυχώς.

Σημείωση: Τα Timings που αναγράφονται παρακάτω είναι ο χρόνος διάρκειας της κάθε εντολής.

Βήμα 1ο : Εγκατάσταση εξαρτήσεων.

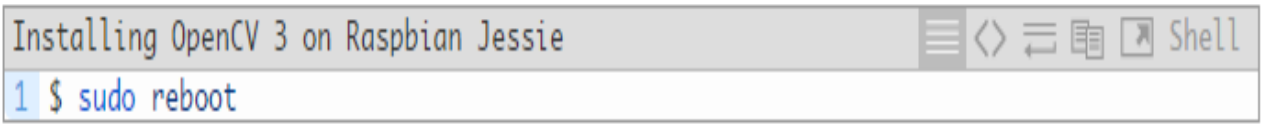
Το πρώτο πράγμα που πρέπει να κάνουμε είναι να αναβαθμίσουμε όλα τα υπάρχοντα πακέτα που υπάρχουν στο σύστημά μας και έπειτα να αναβαθμίσουμε το λογισμικό του Raspberry Pi μας.

Installing OpenCV 3 on Raspbian Jessie Shell

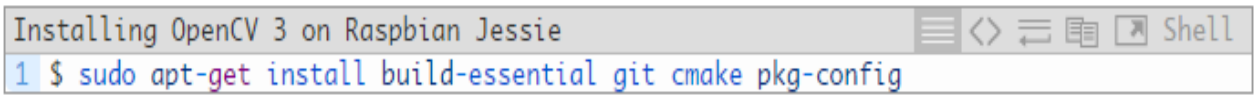
```
1 $ sudo apt-get update
2 $ sudo apt-get upgrade
3 $ sudo rpi-update
```

Timing: 3m 33s

Θα πρέπει να γίνει επανεκκίνηση του συστήματος μετά την αναβάθμιση του λογισμικού.



Τώρα θα εγκαταστήσουμε κάποια προγραμματιστικά εργαλεία.

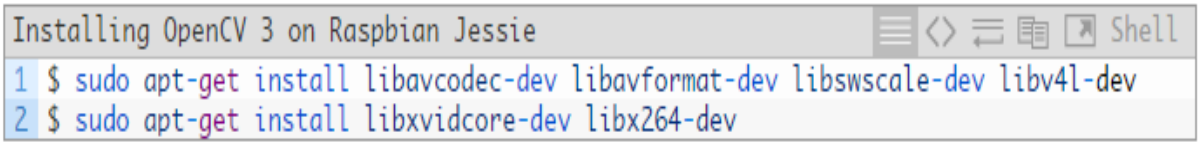


Timing: 51s

Τώρα μπορούμε να προχωρήσουμε στην εγκατάσταση των πακέτων που θα μας δώσουν την δυνατότητα να μπορούμε να φορτώσουμε διάφορες μορφές εικόνων όπως JPEG,PNG,TIFF κ.α.

Timing: 42s

Όπως κάναμε την εγκατάσταση των πακέτων για την είσοδο και έξοδο των φωτογραφιών.Το ίδιο πρέπει να κάνουμε και για τα βίντεο.Αυτό το πακέτο θα μας δώσει την δυνατότητα να μπορούμε να φορτώσουμε διάφορες μορφές βίντεο ακόμα και για την χρήση του βίντεο Stream.



Timing: 58s

Τώρα πρέπει να εγκαταστήσουμε την GTK βιβλιοθήκη ώστε να μπορούμε να μεταγλωτίσουμε το highgui το οποίο είναι μια υποενότητα της βιβλιοθήκης OpenCV και μας δίνει την δυνατότητα να εμφανίζουμε εικόνες στην οθόνη μας και να χτίζουμε διάφορα GUI (Graphical User Interface).

```
Installing OpenCV 3 on Raspbian Jessie                    Shell
1 $ sudo apt-get install libgtk2.0-dev
```

Timing: 2m 48s

Κάποιες λειτουργίες του OpenCV μπορούν να βελτιστοποιηθούν εγκαθιστώντας το παρακάτω πακέτο.

```
Installing OpenCV 3 on Raspbian Jessie                    Shell
1 $ sudo apt-get install libatlas-base-dev gfortran
```

Timing: 50s

Τελειώνοντας με το πρώτο βήμα της εγκατάστασης θα πρέπει να γίνει η εγκατάσταση των Header Files της γλώσσας προγραμματισμού Python 2.7 και Python 3 έτσι ώστε να έχουμε την δυνατότητα μεταγλώτισης των προγαμμάτων που πρόκειται να αναπτύξουμε.

```
Installing OpenCV 3 on Raspbian Jessie                    Shell
1 $ sudo apt-get install python2.7-dev python3-dev
```

Βήμα 2ο : Κατέβασμα πηγαίου κώδικα OpenCV.

Σε αυτό το το σημείο που βρισκόμαστε έχωντας εγκαταστήσει όλα τα απαραίτητα εργαλεία, συνεχίζουμε κατεβάζοντας το πηγαίο κώδικα της βιβλιοθήκης OpenCV απευθείας από το GitHub Repository.

Installing OpenCV 3 on Raspbian Jessie — Shell

```
1 $ cd ~
2 $ wget -O opencv.zip https://github.com/Itseez/opencv/archive/3.0.0.zip
3 $ unzip opencv.zip
```

Timing: 2m 29s

Για την πλήρη εγκατάσταση του OpenCV 3 (το οποίο περιέχει χαρακτηριστικά όπως το SIFT και το SURF) θα πρέπει να κατεβάσουμε και το opencv_contrib Repository επίσης.

Installing OpenCV 3 on Raspbian Jessie — Shell

```
1 $ wget -O opencv_contrib.zip https://github.com/Itseez/opencv_contrib/archive/3.0.0
2 $ unzip opencv_contrib.zip
```

Timing: 1m 54s

Βήμα 3ο : Εγκατάσταση Γλώσσας Προγραμματισμού Python.

Το πρώτο πράγμα που πρέπει να κάνουμε για να εγκαταστήσουμε την γλώσσα προγραμματισμού Python στο σύστημά μας είναι να εγκαταστήσουμε το pip το οποίο είναι ένας διαχειριστής πακέτων.

Installing OpenCV 3 on Raspbian Jessie — Shell

```
1 $ wget https://bootstrap.pypa.io/get-pip.py
2 $ sudo python get-pip.py
```

Timing: 26s

Το επόμενο βήμα που θα ακολουθήσει είναι προαιρετικό αλλά θα μπορούσε να βοηθήσει καθώς μας δίνει την δυνατότητα να δημιουργούμε μεμονωμένα Python περιβάλλοντα το οποίο θα μας ωφελούσε αν θέλαμε να χρησιμοποιήσουμε διάφορες εκδόσεις της Python στα εκάστοτε προγράμματα που επρόκειτο να αναπτύξουμε στο μέλλον.

Installing OpenCV 3 on Raspbian Jessie — Shell

```
1 $ sudo pip install virtualenv virtualenvwrapper
2 $ sudo rm -rf ~/.cache/pip
```

Timing: 17s

Αφού έγινε η εγκατάσταση των virtualenv και virtualenvwrapper, πρέπει να ανανεώσουμε το **~/.profile** αρχείο μας και να προσθέσουμε τις παρακάτω γραμμές στο τέλος του αρχείου.

Installing OpenCV 3 on Raspbian Jessie — Shell

```
1 # virtualenv and virtualenvwrapper
2 export WORKON_HOME=$HOME/.virtualenvs
3 source /usr/local/bin/virtualenvwrapper.sh
```

Μπορούμε να χρησιμοποιήσουμε όποιον File Editor θελήσουμε για κάνουμε αυτή την αλλαγή στο αρχείο.Το μόνο που έχουμε να κάνουμε είναι αρχικά να ανοίξουμε το αρχείο μας το οποίο

βρίσκεται στο /home/pi/.profile και να εισάγουμε τις παραπάνω γραμμές στο τέλος του αρχείου.

Τώρα που το ~/.profile έχει ανανεωθεί θα πρέπει να το φορτώσομε ξανά έτσι ώστε να λειτουργήσουν οι αλλαγές οι οποίες κάναμε.Αυτό γίνεται με την παρακάτων εντολή.

Installing OpenCV 3 on Raspbian Jessie — Shell

```
1 $ source ~/.profile
```

Σημείωση: Θα πρέπει να τρέχουμε την παραπάνω εντολή στο σύστημα μας κάθε φορά που ανοίγουμε έναν νέο Terminal για να είμαστε σίγουροι ότι το περιβάλλον μας έχει στηθεί σωστά.

Το επόμενο βήμα μας είναι να δημιουργήσουμε το Python περιβάλλον μας όπου θα χρησιμοποιούμε για να αναπτύσουμε προγράμματα μελλοντικά.

Installing OpenCV 3 on Raspbian Jessie — Shell

```
1 $ mkvirtualenv cv
```

Η παραπάνω εντολή δημιουργεί ένα εικονικό περιβάλλον εν ονόματι cv χρησιμοποιόντας την έκδοση 2.7 της Python.

Αν θέλουμε να κάνουμε το ίδιο αλλά να χρησιμοποιήσουμε την έκδοση 3 της Python θα πρεέπει να τρέξουμε την παρακάτω εντολή.

Installing OpenCV 3 on Raspbian Jessie — Shell

```
1 $ mkvirtualenv cv -p python3
```

Θα πρέπει να σημειωθεί ξανά ότι το cv περιβάλλον της Python που δημιουργήσαμε είναι εντελώς ανεξάρτητο με την προκαθορισμένη έκδοση της Python που υπάρχει ήδη στο σύστημα μας.

Όποτε κάνουμε επανεκίννηση του συστήματός μας ή ανοίξουμε νέο Terminal θα πρέπει να χρησιμοποιούμε την εντολή workon για να έχουμε μια εκ νέου πρόσβαση στο cv εικονικό περιβάλλον, αλλιώς το σύστημα μας θα χρησιμοποιεί την προκαθορισμένη έκδοση της Python που έχει το σύστημά μας.

```
Installing OpenCV 3 on Raspbian Jessie                    Shell
1 $ source ~/.profile
2 $ workon cv
```

Μπορούμε να βεβαιωθούμε ότι βρισκόμαστε στο εικονικό περιβάλλον μας βλέποντας στο Terminal μας τους χαρακτήρες (cv) όπως στην παρακάτων εικόνα.

Εικόνα 3.3: Σιγουρευόμαστε ότι το (cv) υπάρχει σαν πρόσθετο μπροστά από το Username.

Αν δεν υπάρχουν αυτοί οι χαρακτήρες σαν πρόθεμα αυτό σημαίνει ότι δεν είμαστε στο εικονικό περιβάλλον μας.

Εικόνα 3.4: Αν δεν υπάρχει το (cv) υπάρχει σαν πρόσθετο μπροστά από το Username τότε δεν είμαστε στο εικονικό μας περιβάλλον.

Σε αυτή την περιίπτωση θα πρέπει να χρησιμοποιήσουμε τις εντολές source kai workon που είδαμε παραπάνω.

Αν υποθέσουμε ότι βρισκόμαστε στο εικονικό μας περιβάλλον,μπορούμε να εγκαταστήσουμε το Numpy, ένα πάρα πολύ σημαντικό εργαλείο που μας βοηθάει όταν μεταγλώτίζουμε Python προγράμματα για την OpenCV.

```
Installing OpenCV 3 on Raspbian Jessie                    Shell
1 $ pip install numpy
```

Βήμα 4ο : Μεταγλώττιση και Εγκατάσταση της βιβλιοθήκης OpenCV.

Σε αυτό το σημείο είμαστε έτοιμοι να μεταγλωττίσουμε το πακέτο της OpenCV.

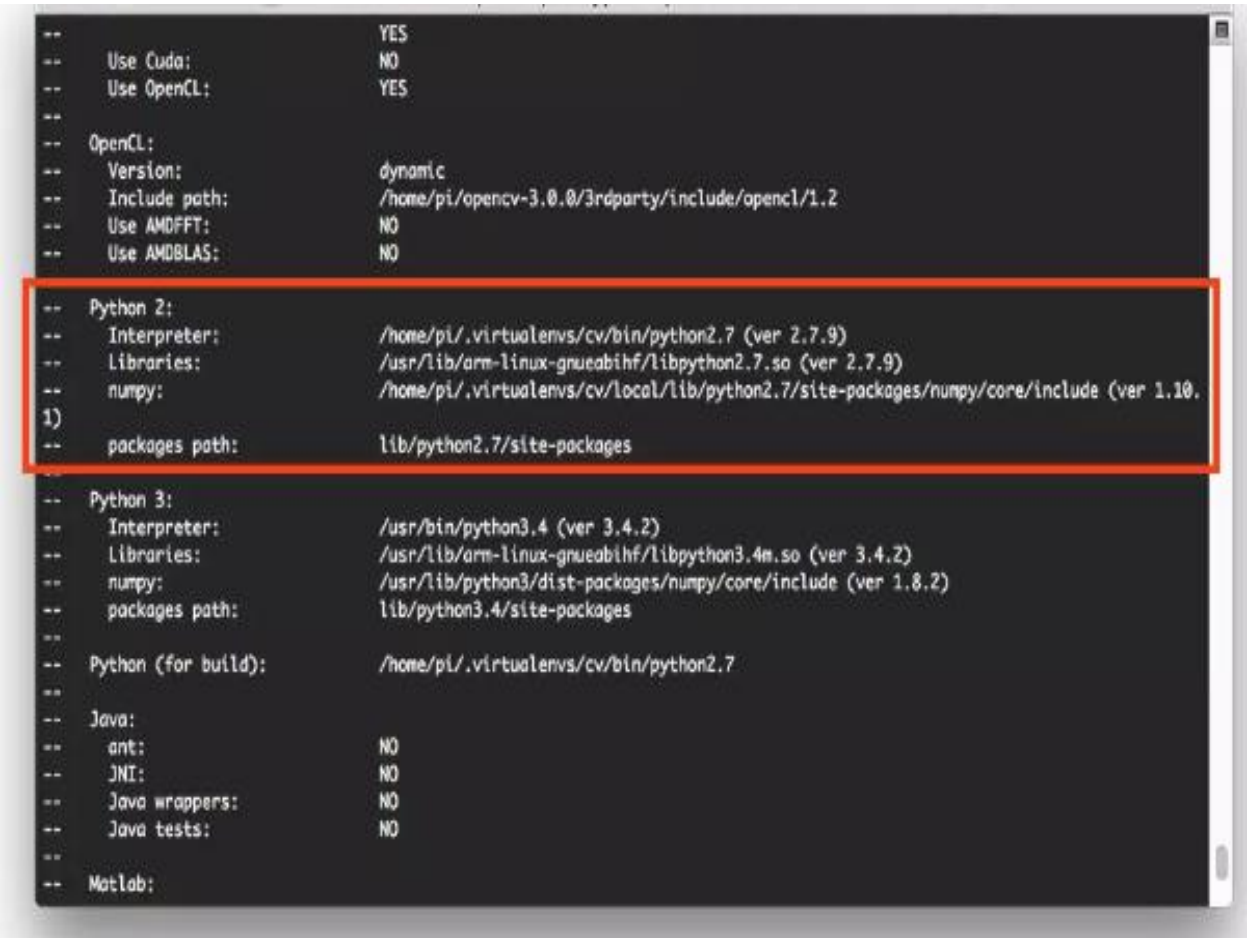
Ας σιγουρευτούμε αρχικά ότι βρισκόμαστε στο cv εικονικό περιβάλλον μας.

```
Installing OpenCV 3 on Raspbian Jessie                    Shell
1 $ workon cv
```

Και έπειτα ας εισάγουμε τις παρακάτων εντολές.

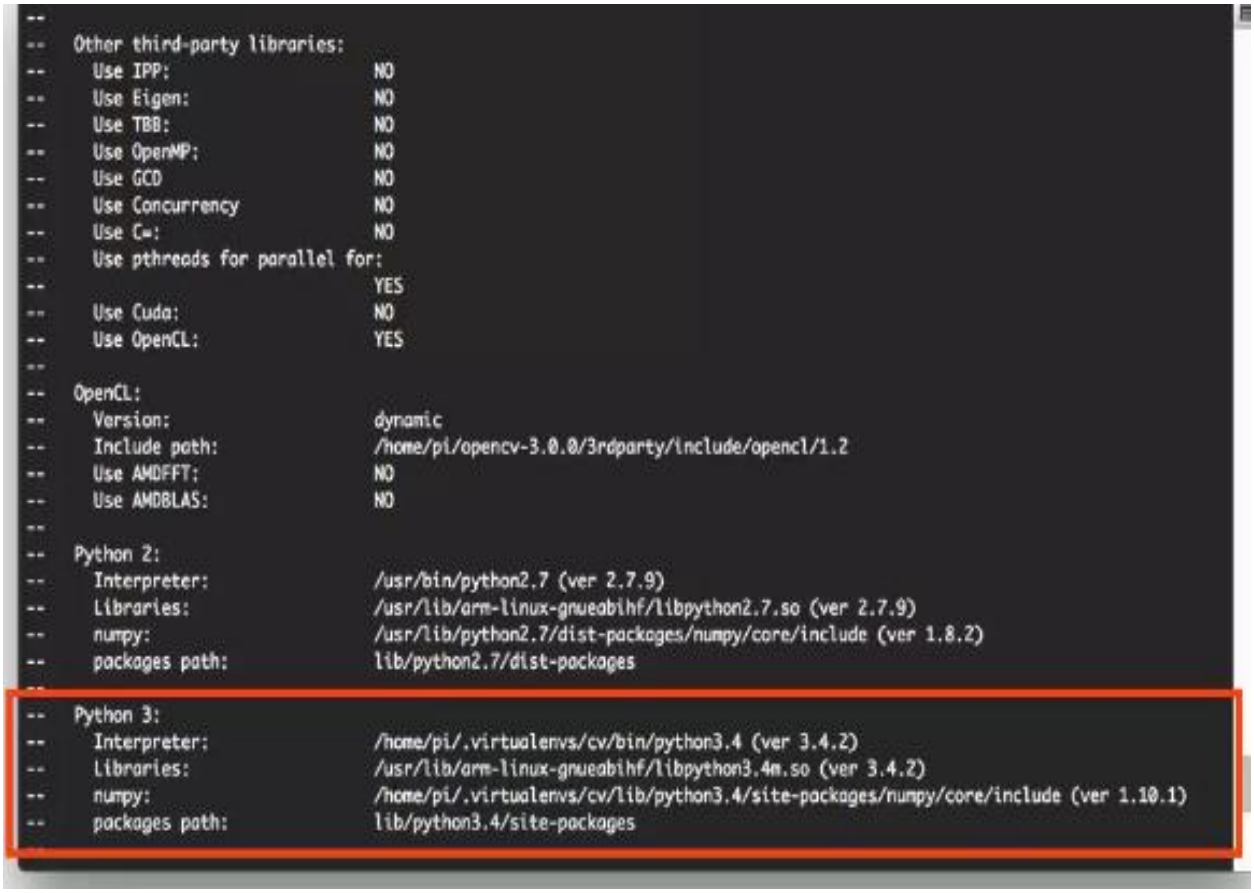
```
Installing OpenCV 3 on Raspbian Jessie                    Shell
1 $ cd ~/opencv-3.0.0/
2 $ mkdir build
3 $ cd build
4 $ cmake -D CMAKE_BUILD_TYPE=RELEASE \
5     -D CMAKE_INSTALL_PREFIX=/usr/local \
6     -D INSTALL_C_EXAMPLES=ON \
7     -D INSTALL_PYTHON_EXAMPLES=ON \
8     -D OPENCV_EXTRA_MODULES_PATH=~/opencv_contrib-3.0.0/modules \
9     -D BUILD_EXAMPLES=ON ..
```

Αφού τελειώσει αυτή η διαδικασία θα πρέπει να τσεκάρουμε στον Terminal το Output του CMake. Αν μεταγλωττίσαμε το OpenCV 3 για την έκδοση 2.7 της Python θα πρέπει να δούμε τα παρακάτω στο τερματικό μας.



Εικόνα 3.5.: Αποτέλεσμα του Cmake για την έκδοση 2.7 της Python.

Αν όμως μεταγλωττίσαμε το OpenCV 3 για την έκδοση 3 της Python θα πρέπει να δούμε τα παρακάτω στο τερματικό μας.



Εικόνα 3.6: Αποτέλεσμα του Cmake για την έκδοση 3 της Python.

Έχοντας φτάσει σε αυτό το σημείο επιτυχώς είμαστε έτοιμοι να μεταγλωττίσουμε τη βιβλιοθήκη του OpenCV στο σύστημά μας χρησιμοποιώντας την παρακάτω εντολή.

Installing OpenCV 3 on Raspbian Jessie Shell

```
1 $ make -j4
```

Timing: 1h 35m

Το -j4 μας υποδεικνύει πόσους πυρήνες του συστήματος μας θα χρησιμοποιήσουμε για να μεταγλωττίσουμε την βιβλιοθήκη του OpenCV.Εφόσον χρησιμοποιούμε Raspberry Pi θα χρησιμοποιήσουμε και τους 4 πυρήνες μας να γίνει η μεταγλώττιση πιο γρήγορα.

Αν υπάρξουν τυχόν λάθη κατά την μεταγλώττιση μπορούμε να χρησιμοποιήσουμε την παρακάτω εντολή και να ξεκινήσουμε την μεταγλώττιση ξανά από την αρχή χρησιμοποιώντας έναν μόνο πυρήνα του συστήματος.

Installing OpenCV 3 on Raspbian Jessie — Shell

```
1 $ make clean
2 $ make
```

Χρησιμοποιώντας έναν μόνο πυρήνα λογικό είναι η μεταγλώττιση να αργήσει πολύ περισσότερο.

Ας υποθέσουμε ότι τελείωσε η μεταγλώττιση χωρίς λάθη. Το μόνο που μένει πλέον είναι να γίνει η εγκατάσταση στο σύστημα μας.

Installing OpenCV 3 on Raspbian Jessie — Shell

```
1 $ sudo make install
2 $ sudo ldconfig
```

Βήμα 5ο Τελειώνοντας την Εγκατάσταση.

Για την έκδοση 2.7 της Python.

Έχοντας τελειώσει το βήμα 4ο χωρίς κανένα πρόβλημα. Το OpenCV θα έχει εγκατασταθεί επιτυχώς στο παρακάτω directory:

/usr/local/lib/python2.7/site-packages

Με την παρακάτω εντολή μπορούμε να σιγουρευτούμε ότι έχουν πάει όλα καλά.

```
Installing OpenCV 3 on Raspbian Jessie                    Shell
1 $ ls -l /usr/local/lib/python2.7/site-packages/
2 total 1636
3 -rw-r--r-- 1 root staff 1675144 Oct 17 15:25 cv2.so
```

Σημείωση: Σε ορισμένες περιπτώσεις υπάρχει περίπτωση το OpenCV να έχει εγκατασταθεί σε αυτό το directory: /usr/local/lib/python2.7/dist-packages.

Το τελευταίο βήμα που μας έχει απομείνει είναι να συνδέσουμε την βιβλιοθήκη OpenCV με το εικονικό μας CV περιβάλλον.

```
Installing OpenCV 3 on Raspbian Jessie                    Shell
1 $ cd ~/.virtualenvs/cv/lib/python2.7/site-packages/
2 $ ln -s /usr/local/lib/python2.7/site-packages/cv2.so cv2.so
```

Για την έκδοση 3 της Python.

Το OpenCV θα έχει εγκατασταθεί επιτυχώς στο παρακάτω directory:

/usr/local/lib/python3.4/site-packages

Με την παρακάτω εντολή μπορούμε να σιγουρευτούμε ότι έχουν πάει όλα καλά.

```
Installing OpenCV 3 on Raspbian Jessie                    Shell
1 $ ls /usr/local/lib/python3.4/site-packages/
2 cv2.cpython-34m.so
```

Όπως βλέπουμε παραπάνω, για κάποιον άγνωστο λόγο το αρχείο cv2.so στην συγκεκριμένη περίπτωση έχει διαφορετική ονομασία και λέγεται cv2.cpython-32m.so

Για να το διορθώσουμε αυτό, επειδή μελλοντικά θα μας δημιουργούσε πρόβλημα στις μεταγλωττίσεις μας απλά τρέχουμε την παρακάτω εντολή η οποία αλλάζει το όνομα του cv2.cpython-32m.so στην σωστή του ονομασία η οποία είναι cv2.so.

Installing OpenCV 3 on Raspbian Jessie — Shell

```
1 $ cd /usr/local/lib/python3.4/site-packages/
2 $ sudo mv cv2.cpython-34m.so cv2.so
```

Το τελευταίο βήμα που μας έχει απομείνει είναι να συνδέσουμε την βιβλιοθήκη OpenCV με το εικονικό μας CV περιβάλλον.

Installing OpenCV 3 on Raspbian Jessie — Shell

```
1 $ cd ~/.virtualenvs/cv/lib/python3.4/site-packages/
2 $ ln -s /usr/local/lib/python3.4/site-packages/cv2.so cv2.so
```

Βήμα 6ο Επαληθεύοντας την Εγκατάσταση του OpenCV.

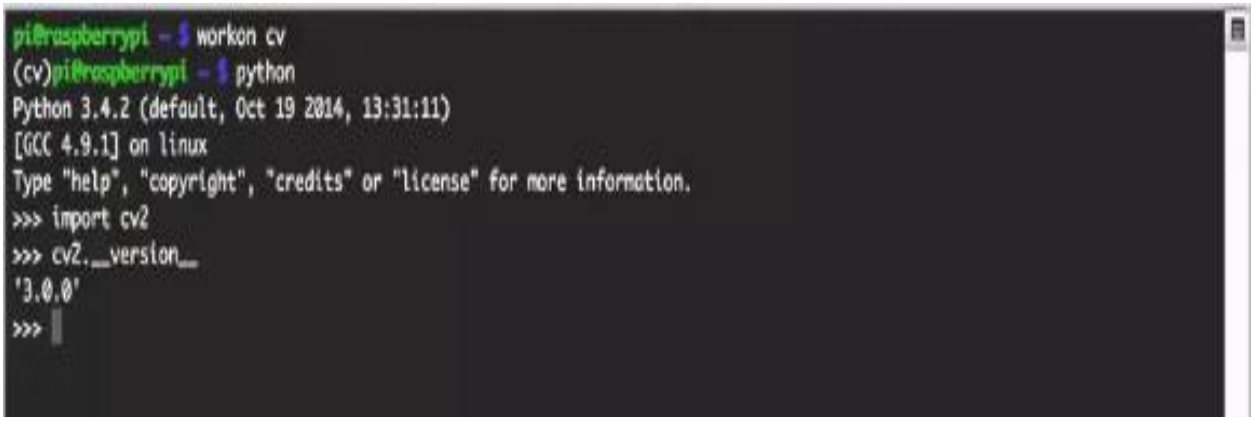
Σε αυτό το σημείο, το OpenCV 3 έχει εγκατασταθεί επιτυχώς στο σύστημά μας και το τελευταίο βήμα που μας απομένει είναι να επαληθεύσουμε ότι η βιβλιοθήκη του OpenCV, λειτουργεί κανονικά. Για να το επαληθεύσουμε αυτό πρέπει αρχικά να αποκτήσουμε

Installing OpenCV 3 on Raspbian Jessie — Shell

```
1 $ workon cv
2 $ python
3 >>> import cv2
4 >>> cv2.__version__
5 '3.0.0'
```

πρόσβαση στο εικονικό μας περιβάλλον CV που είχαμε δημιουργήσει στα προηγούμενα βήματα και να κάνουμε εισαγωγή το αρχείο cv2.so.

Στη παρακάτω εικόνα θα δούμε σε ένα τερματικό ποιο θα είναι το αποτέλεσμα των παραπάνω εντολών εφόσον όλα έχουν πάει καλά.



Εικόνα 3.7: Αποτέλεσμα τερματικού επιτυχόντας εγκατάστασης OpenCV.

ΚΕΦΑΛΑΙΟ 4

ΛΟΓΙΣΜΙΚΑ - SOFTWARE

4.1 Putty

Το Putty είναι ένας δωρεάν και ανοιχτού κώδικα προσομοιωτής τερματικού ο οποίος μπορεί να χρησιμοποιηθεί επίσης ως σειριακή κονσόλα για μια εφαρμογή μεταφοράς αρχείων μέσω του δικτύου. Έχει την δυνατότητα να υποστηρίξει διάφορα πρωτόκολλα δικτύων, συμπεριλαμβανομένων των SCP, SSH, Telnet και Rlogin. Έχει επίσης την δυνατότητα να συνδεθεί σε μία σειριακή θύρα.

Το Putty δημιουργήθηκε αρχικά για τα Microsoft Windows, αλλά εφόσον καινούρια λειτουργικά συστήματα δημιουργούνταν με τον καιρό δημιουργήθηκε και η ανάγκη επέκτασης του σε αυτά. Στην συγκεκριμένη εργασία θα το χρησιμοποιήσουμε για την επικοινωνία μας με το Raspberry Pi μας μέσω του πρωτοκόλλου SSH.

4.1.1 Το πρωτόκολλο SSH

Το **SSH** (Secure Shell) είναι ένα ασφαλές δικτυακό πρωτόκολλο το οποίο επιτρέπει τη μεταφορά δεδομένων μεταξύ δύο υπολογιστών. Το SSH όχι μόνο κρυπτογραφεί τα δεδομένα που ανταλλάσσονται κατά τη συνεδρία, αλλά προσφέρει ένα ασφαλές σύστημα αναγνώρισης καθώς και άλλα χαρακτηριστικά όπως ασφαλή μεταφορά αρχείων (SSH File Transfer Protocol, SFTP), κλπ.

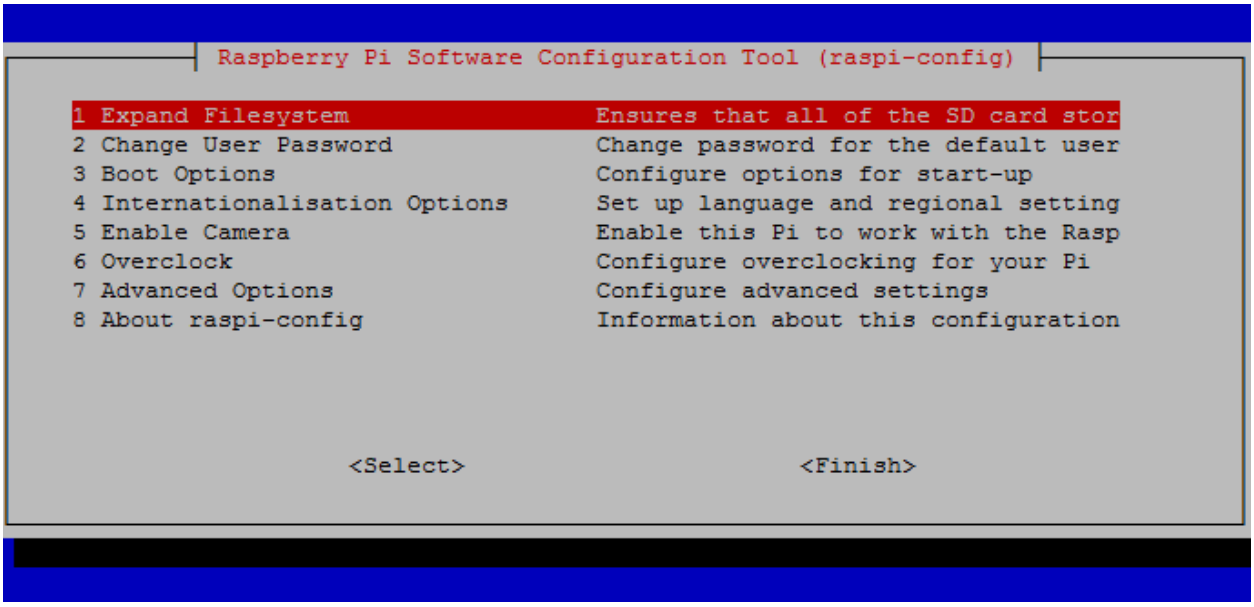
4.1.2 Γιατί επιλέξαμε το Putty

Το Putty χρησιμοποιήθηκε στην παρούσα εργασία διότι είναι ένα δωρεάν εργαλείο που διανείμετε στο διαδίκτυο και μας έδινε την δυνατότητα να προγραμματίζουμε απομακρυσμένα (OTA – Over The Air) το Raspberry Pi μας από όποιον υπολογιστή εμείς θέλαμε. Ήταν πάρα πολύ χρήσιμο κατά την διάρκεια πειραματισμού του οχήματος μας.

4.1.3 Προαπαιτούμενες Ρυθμίσεις

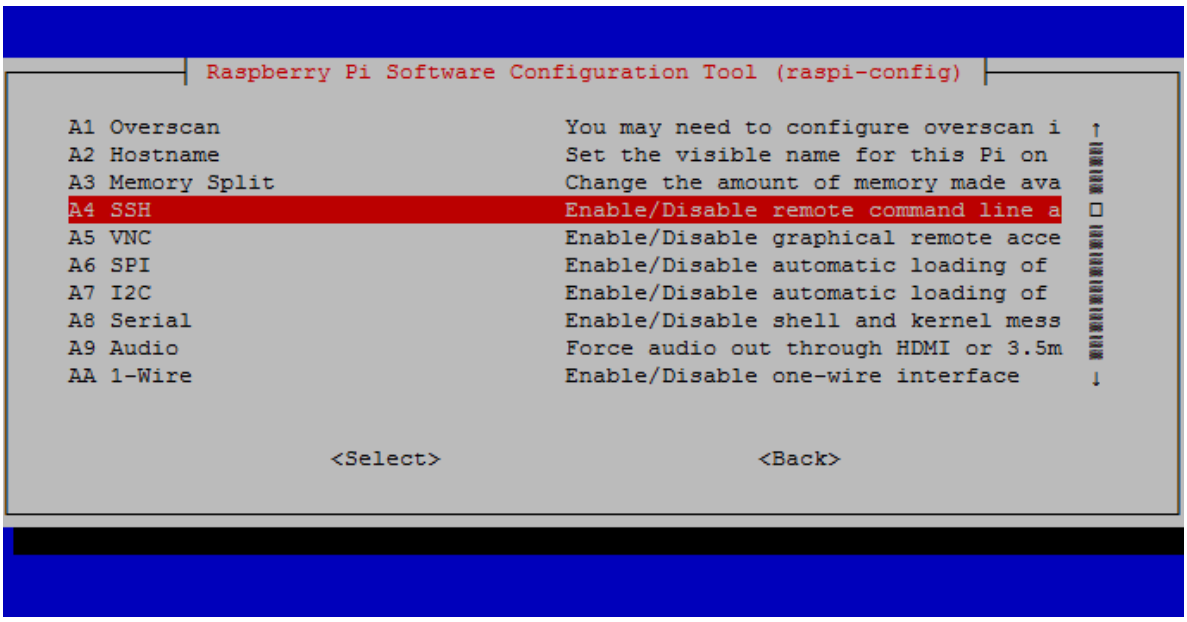
Για να πραγματοποιηθεί επιτυχώς η σύνδεσή μας με το Raspberry Pi θα πρέπει αρχικά να εκκινήσουμε για μια και μοναδική φορά το Raspberry Pi με γραφικό περιβάλλον έτσι ώστε να ενεργοποιήσουμε από τις ρυθμίσεις του Raspberry Pi το πρωτόκολλο SSH και οποιοδήποτε άλλο πρωτόκολλο θα μας χρειαστεί μελλοντικά.

Έχοντας ήδη εκκινήσει το Raspberry Pi μας ανοίγουμε ένα τερματικό και εισάγουμε την παρακάτω εντολή:



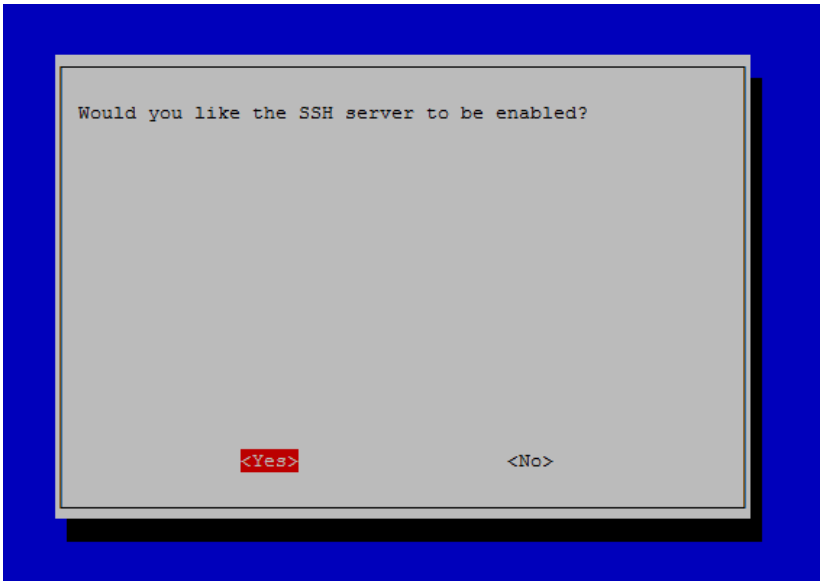
Εικόνα 4.1: Αποτέλεσμα εντολής **sudo raspi-config**.

Αφού μας εμφανιστεί στην οθόνη αυτό που είδαμε στην παραπάνω εικόνα, χρησιμοποιώντας τα βελάκια κατεβαίνουμε στις 7η επιλογή που έχει την ονομασία «**Advanced Options**» και πατάμε Enter.



Εικόνα 4.2: Αποτέλεσμα της επιλογής **«Advanced Options».**

Έπειτα, επιλέγουμε την 4η επιλογή από το μενού που θα εμφανιστεί στην οθόνη μας η οποία έχει την ονομασία **«SSH».**. Πατώντας Enter το σύστημα θα μας ρωτήσει αν θέλουμε να ενεργοποιήσουμε το πρωτόκολλο SSH στο σύστημα μας και εμείς θα επιλέξουμε **«Yes»**



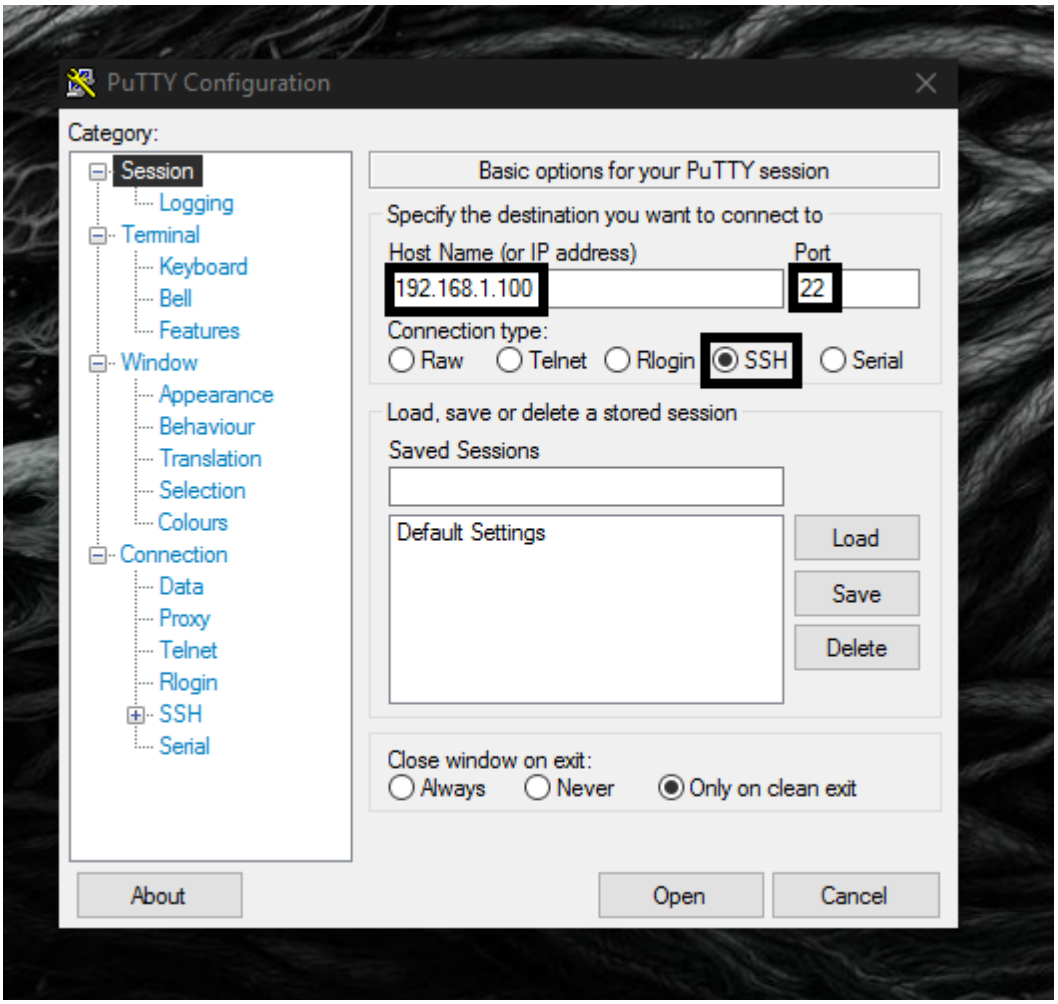
Εικόνα 4.3: Αποτέλεσμα της επιλογής **«SSH».**

Αμέσως το σύστημα θα μας ενημερώσει ότι το πρωτόκολλο SSH μόλις ενεργοποιήθηκε στο σύστημά μας.

Εικόνα 4.4: Μήνυμα επιτυχής ενεργοποίησης πρωτοκόλλου SSH στο Raspberry Pi.

Πατώντας το κουμπί OK που μας έχει εμφανιστεί στη οθόνη είμαστε έτοιμοι για να προχωρήσουμε στη σύνδεση οποιαδήποτε υπολογιστή που βρίσκεται στο δίκτυό μας, με το Raspberry Pi.

4.1.4 Πραγματοποίηση σύνδεσης μέσω Putty

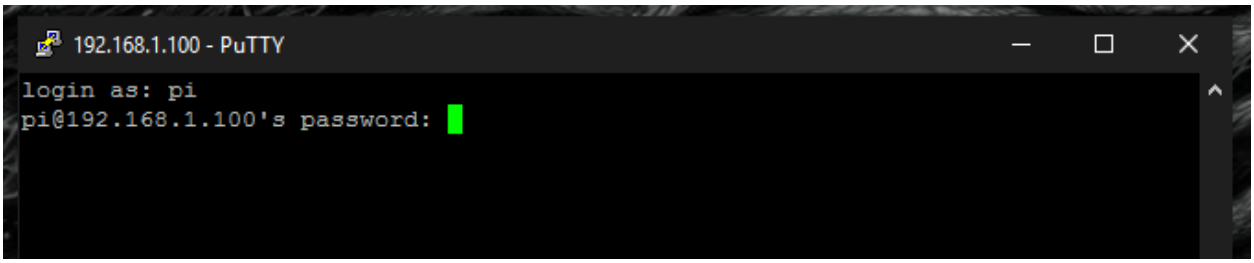


Εικόνα 4.5: Πραγματοποίηση σύνδεσης Desktop με Raspberry Pi χρησιμοποιώντας το Putty.

Για να συνδεθούμε με το Raspberry Pi μας θα πρέπει αρχικά να εισάγουμε τα στοιχεία του Raspberry Pi στα εκάστοτε επισημασμένα πεδία. Στο πεδίο Host Name(or IP address) εισάγουμε την IP Address του Raspberry Pi μας. Στην προκειμένη περίπτωση χρησιμοποιούμε μια Static IP η οποία είναι το 192.168.1.100. Στο πεδίο Port εισάγουμε τον αριθμό 22 η οποία είναι η default τιμή του SSH πρωτοκόλλου. Στο πεδίο Connection Type επιλέγουμε μέσω

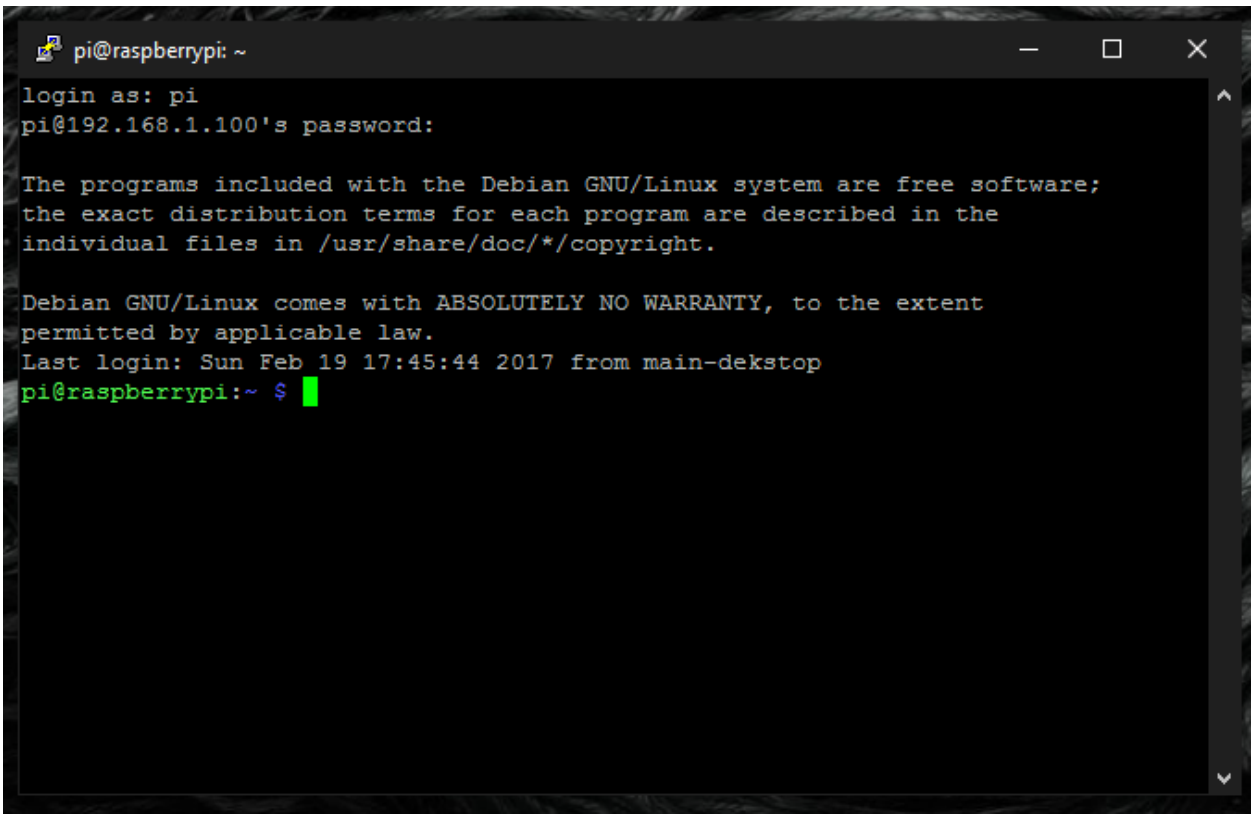
ποιανού πρωτοκόλλου θέλουμε να συνδεθούμε, επιλέγουμε το SSH και πατάμε το κουμπί Open.

Πατώντας το κουμπί Open μας εμφανίζεται ένα τερματικό στο οποίο θα μας ζητηθεί ένα Username και ένα Password.



Εικόνα 4.6: Εισαγωγή στοιχείων Username και Password στο Putty.

Αφού εισάγουμε σωστά το Username και το Password είμαστε πλέον συνδεδεμένοι απομακρυσμένα με το Raspberry Pi μας.



Εικόνα 4.7:Επιτυχής σύνδεση με Raspberry Pi.

4.2 FileZilla

Το λογισμικό FileZilla είναι μια FTP(File Transfer Protocol) εφαρμογή ανοιχτού κώδικα, η οποία μας επιτρέπει να μεταφέρουμε αρχεία από έναν τοπικό υπολογιστή σε έναν απομακρυσμένο. Το FileZilla είναι διαθέσιμο σε δύο εκδόσεις. Η μία έκδοση είναι τύπου Client(ένα τερματικό το οποίο μπορεί να λάβει αρχεία και πληροφορίες από έναν Server) και η άλλη είναι τύπου Server.

Το FileZilla έχει διάφορα χαρακτηριστικά τα οποία αξίζει να επισημάνουμε:

- Διαχειριστή Ιστοσελίδας (Για την δημιουργία και την αποθήκευση μίας λίστας με διακομιστές FTP και τα σχετικά δεδομένα σύνδεσης).
- Σύγκριση καταλόγου (επιτρέπει σε έναν Χρήστη να συγκρίνει τα περιεχόμενα μεταξύ του τοπικού και του απομακρυσμένου καταλόγου.
- Όψη αρχείων και φακέλων (παρόμοιο με τον διαχειριστή αρχείων, επιτρέπει την τροποποίηση των αρχείων και των φακέλων και παρέχει ικανότητα “drag and drop” μεταξύ των τοπικών και απομακρυσμένων καταλόγων).
- Προτεραιότητα μεταφοράς (εμφανίζει την κατάσταση της μεταφοράς των αρχείων σε εξέλιξη ή σε αναμονή για την επεξεργασία).

4.2.1 Το πρωτόκολλο FTP

Το **File Transfer Protocol** (**FTP**) είναι ένα ευρέως χρησιμοποιούμενο πρωτόκολλο σε δίκτυα τα οποία υποστηρίζουν το πρωτόκολλο TCP/IP (δίκτυα όπως internet ή intranet). Ο υπολογιστής που τρέχει εφαρμογή FTP client μόλις συνδεθεί με τον server μπορεί να εκτελέσει

ένα πλήθος διεργασιών όπως ανέβασμα αρχείων στον server, κατέβασμα αρχείων από τον server, μετονομασία ή ακόμα και διαγραφή αρχείων από τον server. Το πρωτόκολλο είναι ένα ανοιχτό πρότυπο. Είναι δυνατό κάθε υπολογιστής που είναι συνδεδεμένος σε ένα δίκτυο, να διαχειρίζεται αρχεία σε ένα άλλο υπολογιστή του δικτύου, ακόμη και εάν ο δεύτερος διαθέτει διαφορετικό λειτουργικό σύστημα.

4.2.2 Γιατί επιλέξαμε το FileZilla

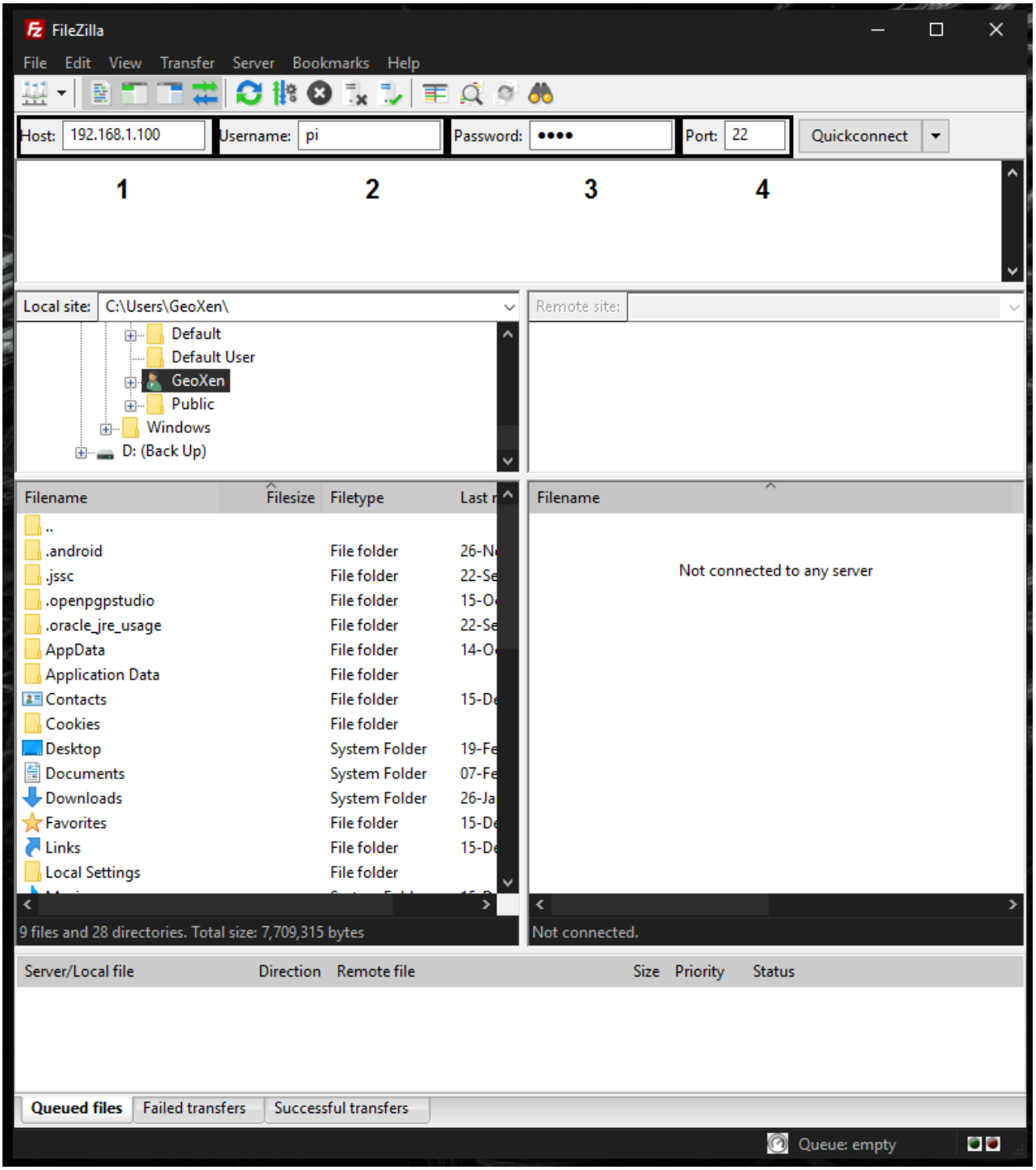
Το FileZilla χρησιμοποιήθηκε στην παρούσα εργασία διότι είναι ένα εργαλείο που διανείμετε δωρεάν στο διαδίκτυο και μας έδινε την δυνατότητα να μεταφέρουμε αρχεία απομακρυσμένα, από έναν υπολογιστή της επιλογής μας στο Raspberry Pi μας ή και το αντίστροφο με τεράστια ταχύτητα και ευκολία.

Εικόνα 4.8: Λογότυπο FileZilla

4.2.3 Πραγματοποίηση FTP σύνδεσης μέσω FileZilla

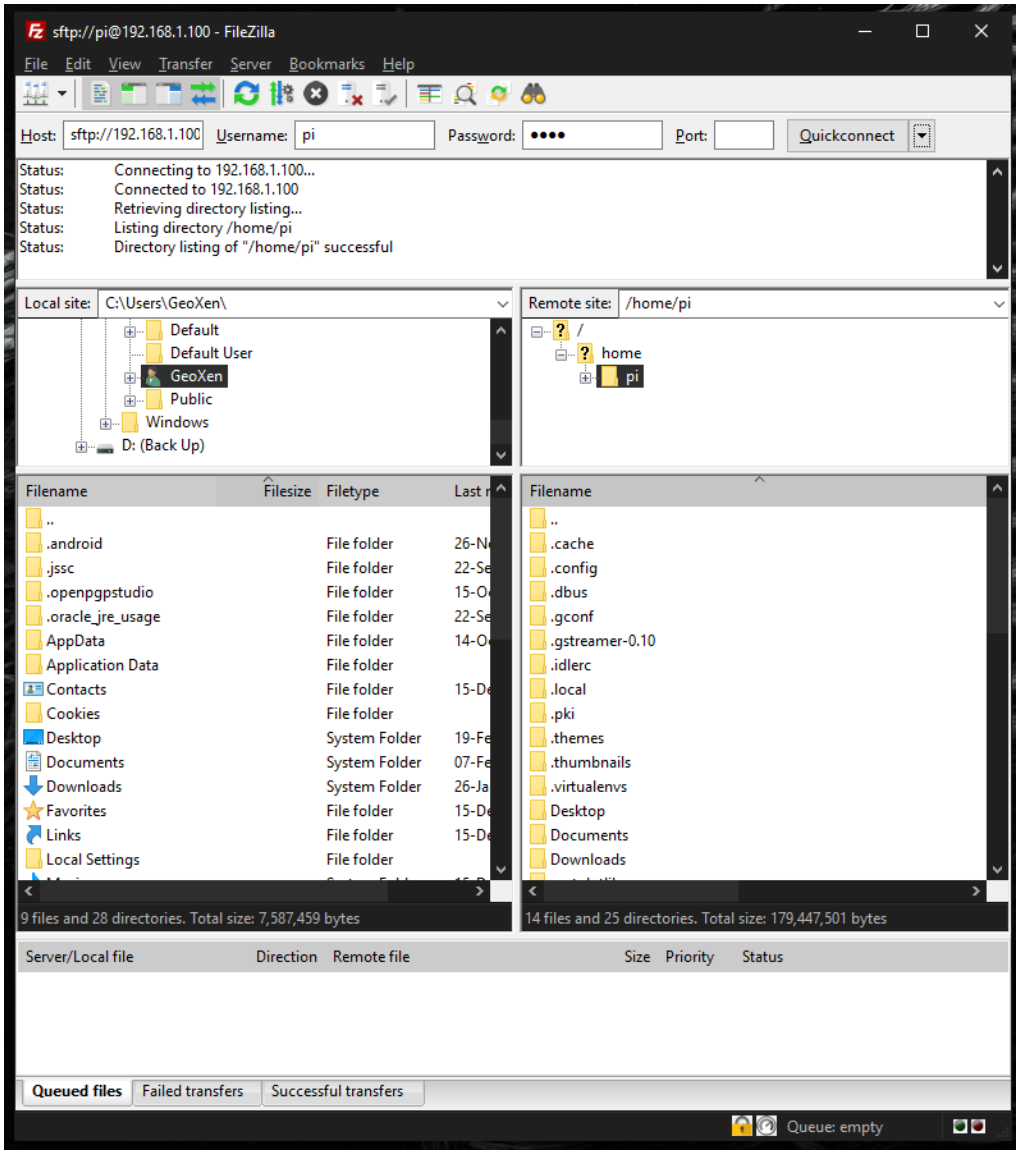
Για την πραγματοποίηση της FTP σύνδεσης ενός υπολογιστή με το Raspberry Pi μας δεν χρειάζεται κάποια προαπαιτούμενη ρύθμιση όπως το Putty. Εφόσον το Raspberry Pi μας είναι ενημερωμένο με την τελευταία έκδοση.Το FTP είναι ήδη εγκατεστημένο και ενεργοποιημένο στο σύστημά μας.

Εκκινώντας την εφαρμογή, όπως θα δούμε και στην παρακάτω εικόνα, για να έχουμε μια επιτυχή σύνδεση μέσω του Raspberry Pi και του υπολογιστή που θα χρησιμοποιήσουμε θα πρέπει να εισάγουμε σωστά τα στοιχεία στα 4 επισημασμένα πεδία. Το πρώτο πεδίο είναι το Host, στο πεδίο αυτό βάζουμε την IP του Raspberry Pi. Το δεύτερο και το τρίτο πεδίο είναι το Username και το Password, όπου βάζουμε το Username και το Password του Raspberry Pi. Στο τέταρτο και τελευταίο πεδίο που είναι το Port, βάζουμε τον αριθμό 22 ο οποίος είναι ο προεπιλεγμένος αρι θμός για το πρωτόκολλο FTP του συστήματος.



Εικόνα 4.9: Εκκίνηση εφαρμογής FileZilla και κατηγοριοποίηση πεδίων.

Εφόσον τα στοιχεία που δώσαμε είναι σωστά πατάμε το πλήκτρο Quickconnect για να συνδεθούμε με τον FTP server του Raspberry Pi.



Εικόνα 4.10: Επιτυχής σύνδεση με Desktop με τον FTP server του Raspberry Pi.

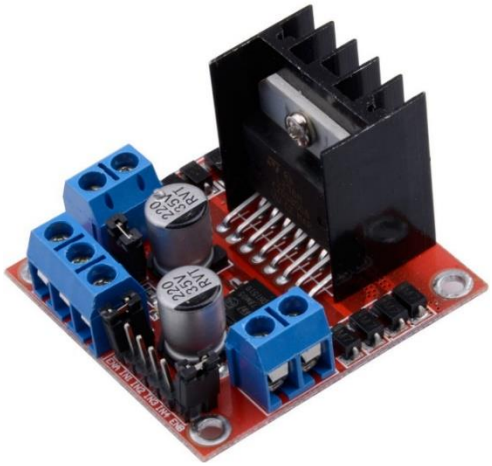
Έχοντας συνδεθεί επιτυχώς με τον FTP server του Raspberry Pi όπως βλέπουμε στην παραπάνω εικόνα μπορούμε να περιηγηθούμε στα αρχεία του Raspberry Pi μας έχοντας την δυνατότητα να τα τροποποιήσουμε, να στείλουμε ή ακόμα και να λάβουμε αρχεία.

ΚΕΦΑΛΑΙΟ 5

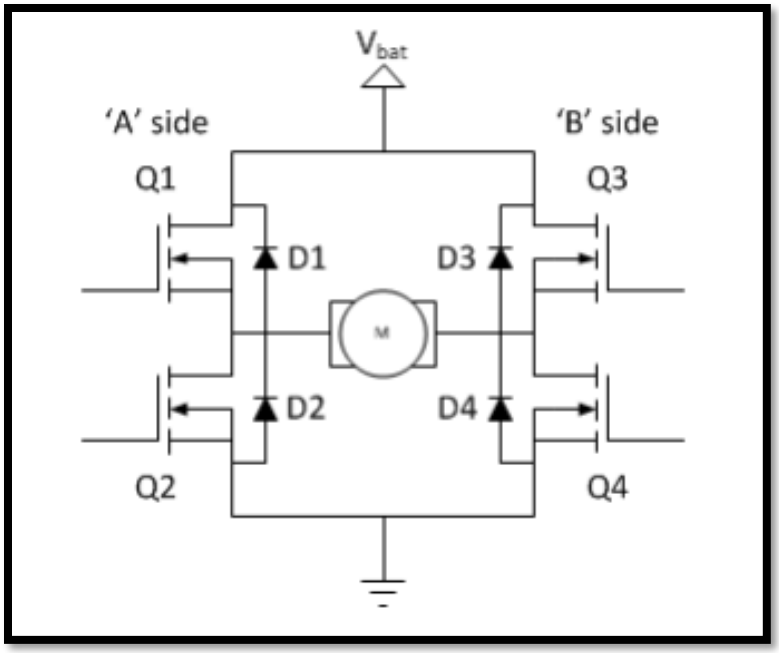
ΥΛΙΚΟ - HARDWARE

5.1 Οδηγός Κινητήρων (LN298N)

Για να ελέγξουμε τους τέσσερις κινητήρες του συστήματος μας χρησιμοποιήσαμε το Module Motor Shield LN298N. Το συγκεκριμένο Module, βασίζεται στο τσιπ LN298N. Το LN298N, αποτελείται από δύο γέφυρες ή γνωστές και ως H-Bridge. Αυτή η κυκλωματική διάταξη επιτρέπει στο τσιπ να οδηγήσει επαγωγικά φορτία όπως ρελέ, DC motor, κ.α. Για τον λόγο αυτόν η H-Bridge, είναι μια ευρέως γνωστή συνδεσμολογία και χρησιμοποιείται σε πολλές ρομποτικές κατασκευές. Έχει πάρει το όνομα της από την διάταξη της καθώς μοιάζει με ¨H¨, αφού έχει στα αριστερά και δεξιά από ένα ζευγάρι διακοπές, οπού συνήθως είναι διπολικά, ή FET transistor, ενώ στο κέντρο έχει το φορτίο. Οι δίοδοι που χρησιμοποιούνται για προστασία, είναι συνήθως Schotky.



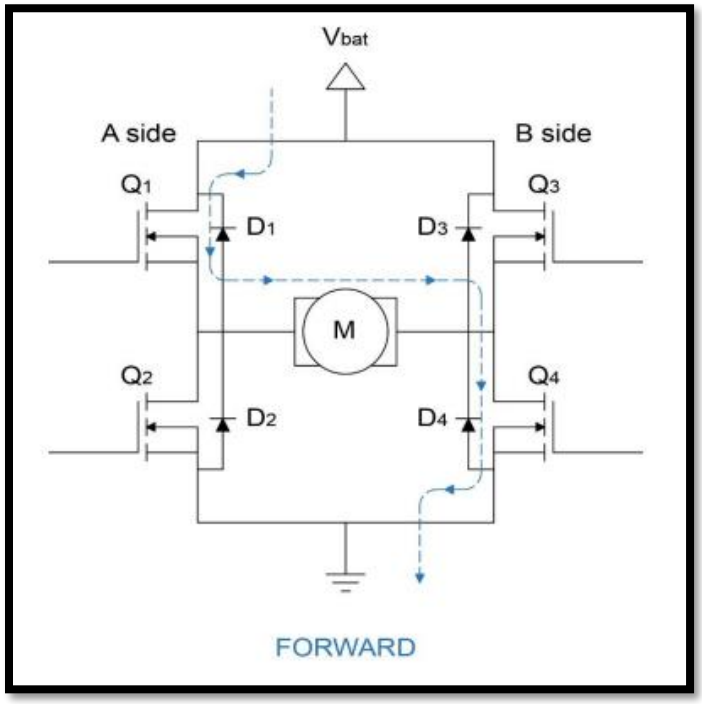
Εικόνα 5.1 : Οδηγός κινητήρων LN298N



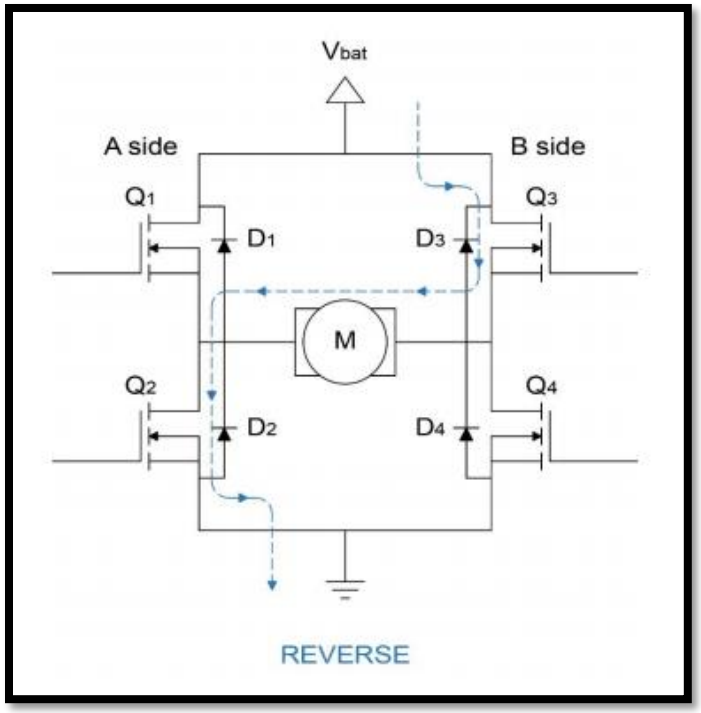
Εικόνα 5.2 : Κυκλωματική Διάταξη H-Bridge

Η λειτουργία της είναι απλή, καθώς όταν τα transistor της εικόνας 5.2 Q1 και Q4 είναι σε ενεργή κατάσταση λειτουργίας, η αριστερή πλευρά του μοτέρ είναι συνδεδεμένη στην τροφοδοσία, ενώ η δεξιά πλευρά στην γείωση.

Αυτό έχει ως αποτέλεσμα το μοτέρ μας να κινείται αριστερόστροφα (μπροστινή κίνηση). Στην αντίθετη περίπτωση όταν τα τρανζίστορ Q2 και Q3 είναι σε λειτουργία, η δεξιά πλευρά του μοτέρ θα είναι συνδεδεμένη στην τροφοδοσία και η αριστερή στην γείωση. Με αποτέλεσμα την κίνηση του μοτέρ δεξιόστροφα (κίνηση προς τα πίσω). Η κίνηση του ρεύματος για τις δυο περιπτώσεις φαίνεται στις εικόνες 5.3, 5.4 .



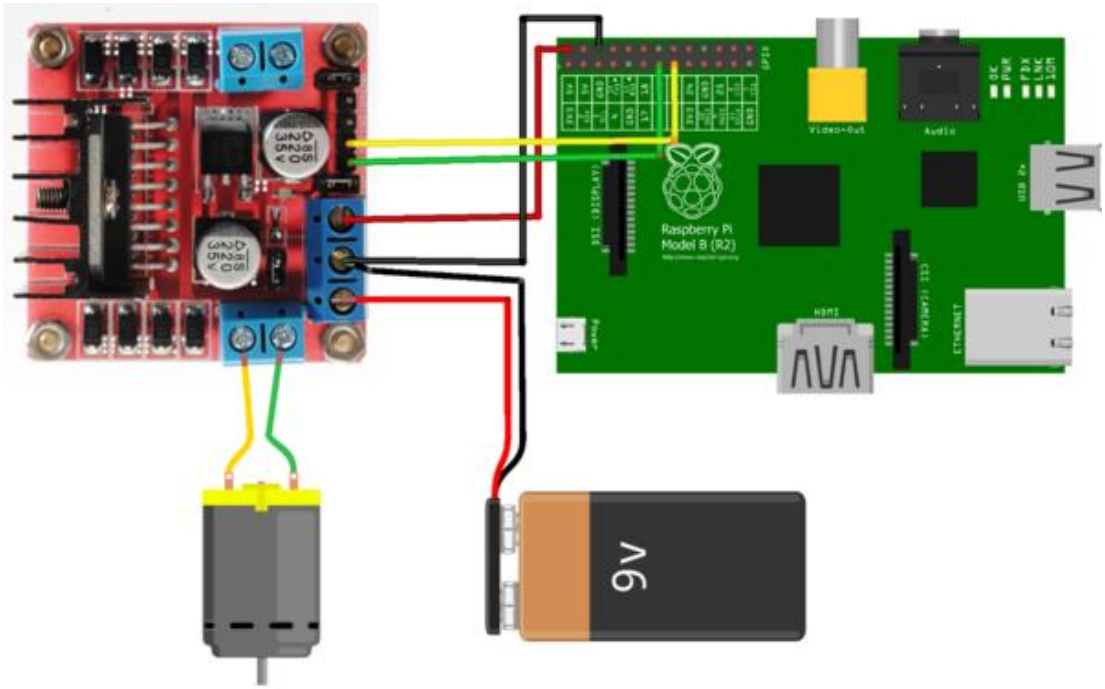
Εικόνα 5.3: Πορεία ρεύματος στην αριστερόστροφη κίνηση



Εικόνα 5.4: Πορεία ρεύματος στην δεξιόστροφη κίνηση

5.1.1 Οδηγός Κινητήρων και Raspberry Pi

Το LN298N συνδέεται με το Raspberry Pi και μπορεί να χρησιμοποιήσει όποια GPIO Pins θέλουμε. Μέσω αυτών των Pin στέλνει PWM παλμούς ώστε να οδηγήσει τα μοτέρ, ρυθμίζοντας την ταχύτητα και την διεύθυνση. Να σημειωθεί ότι το Duty Cycle είναι ανάλογο της ταχύτητας των μοτέρ. Αυξάνοντας το, αυξάνουμε και την ταχύτητα περιστροφής των μοτέρ.



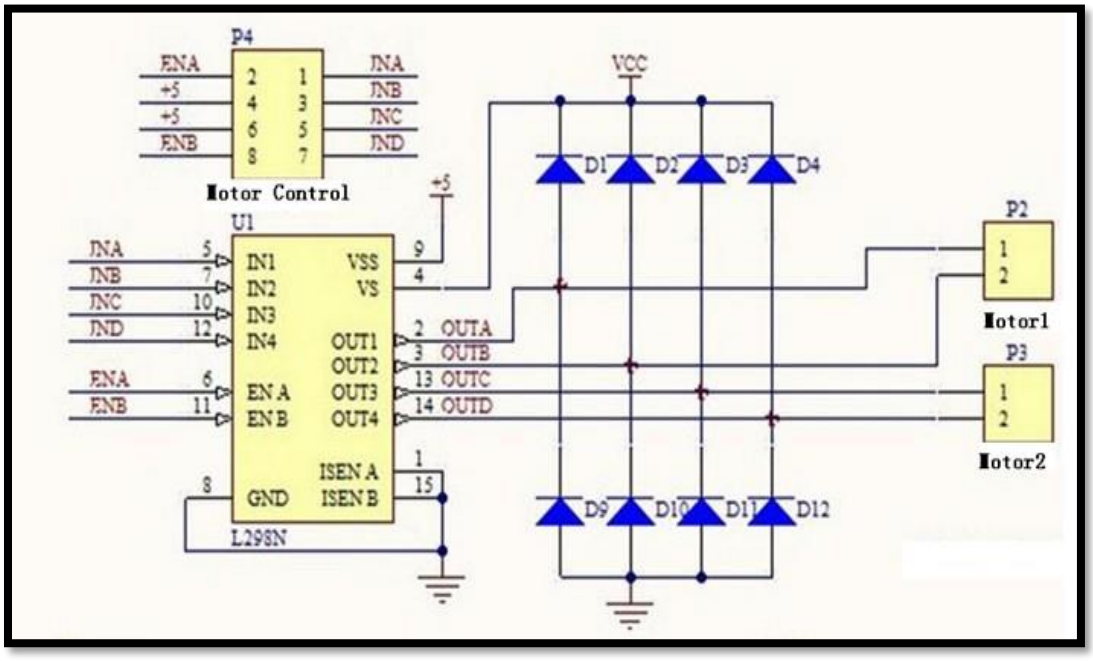
Εικόνα 5.5 : Σχηματική αναπαράσταση σύνδεσης LN298N και Raspberry Pi

Μπορούμε να τροφοδοτήσουμε το LN298N είτε από το Raspberry Pi με 5v είτε από κάποια εξωτερική τροφοδοσία.(5-12V).Στη συγκεκριμένη εργασία χρησιμοποιήσαμε την τροφοδοσία μέσω του Raspberry Pi μιας και δεν είχαμε ως γνώμονα την ταχύτητα στο όχημα μας.

Συνοψίζοντας το LN298N περιέχει:

- Δυο διαφορετικές συνδέσεις τροφοδοσίας.
- Παροχή μέγιστου ρεύματος 2A σε κάθε γέφυρα.
- Pin εξόδου για κάθε κινητήρα, καθώς και Pin (Enable) για τον έλεγχο της κατεύθυνσης των κινητήρων.
- LED που χρησιμοποιούνται ως δείκτες ένδειξης για τα 5V, καθώς και για την ορθή λειτουργία των μοτέρ.
- Κλέμες PCB οπού μας διευκολύνει στην εξωτερική σύνδεση των εξαρτημάτων, στην περίπτωση μας συνδέουμε τα μοτέρ.

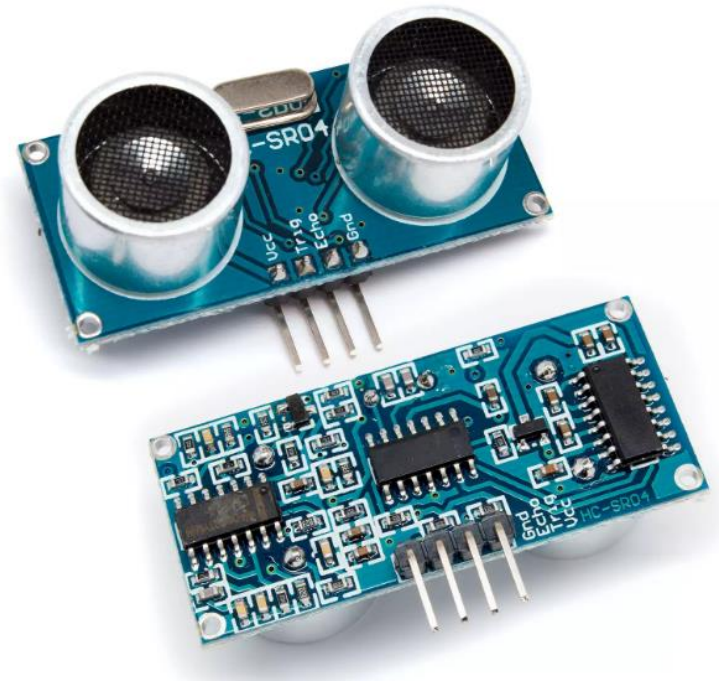
Λόγο της κατασκευής του το LN298N είναι ένα πολύ εύχρηστο εργαλείο οδήγησης επαγωγικών φορτίων. Συνδέοντας δύο μοτέρ σε κάθε H-γέφυρα έχουμε την δυνατότητα να οδηγήσουμε και τα τέσσερα μοτέρ του ρομποτικού συστήματος.



Εικόνα 5.6: Σχηματικό LN298N

5.2 Αισθητήρας Υπερήχων (HC-SR04)

Ως υπέρηχο ορίζουμε το κύμα το οποίο βρίσκεται πάνω από την μέγιστη συχνότητα που μπορεί να ακούσει το ανθρώπινο αυτί. Ο άνθρωπος έχει τη δυνατότητα να αντιλαμβάνεται ήχους με συχνότητες από 20Hz μέχρι 20kHz, ενώ οι υπέρηχοι που αποστέλλει ο συγκεκριμένος αισθητήρας είναι συχνότητας 40kHz.



Εικόνα 5.7 : Αισθητήρας υπέρηχων HC-SR04 με διαστάσεις (45*20*15mm)

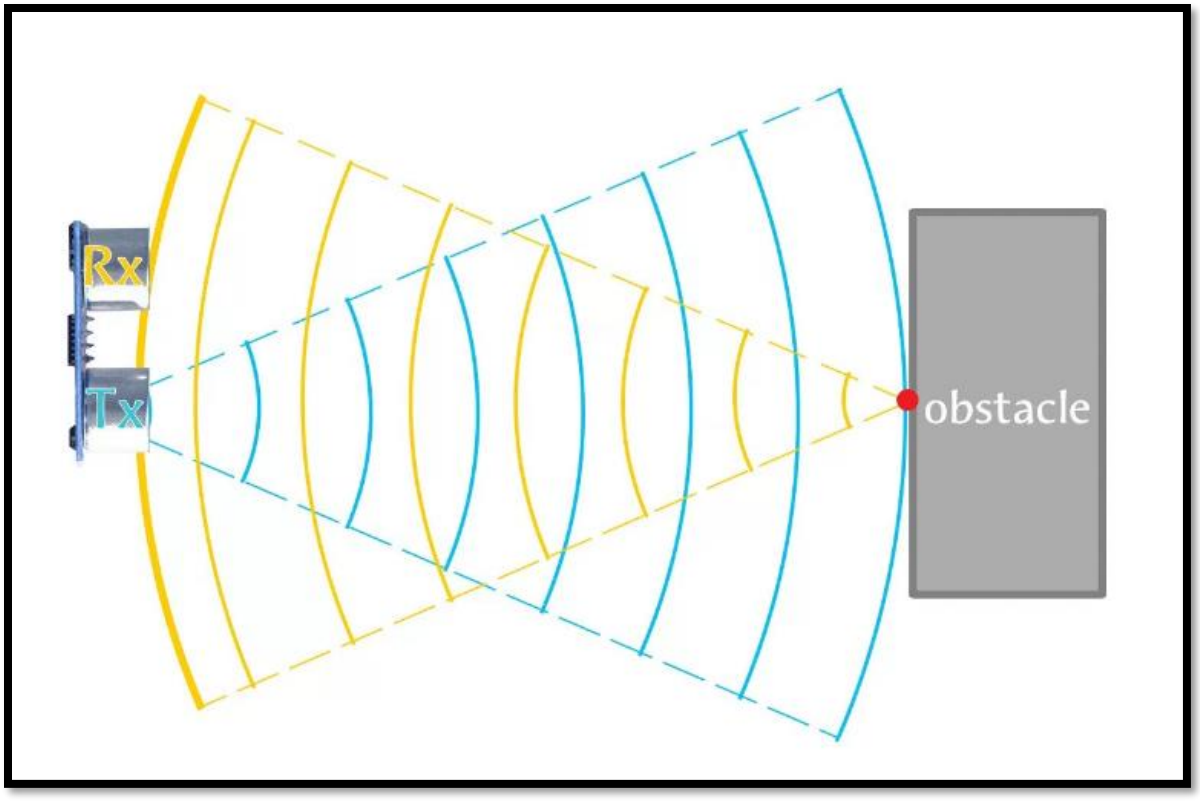
Ο αισθητήρας υπερήχων βασίζεται σε ένα φαινόμενο που φέρει την ονομασία Doppler. Το φαινόμενο Doppler, είναι η παρατηρούμενη αλλαγή στην συχνότητα και το μήκος κύματος

ενός κύματος από κάποιον παρατηρητή που βρίσκεται σε σχετική κίνηση με την πηγή των κυμάτων. Οι συγκεκριμένοι αισθητήρες χρησιμοποιούνται σε εφαρμογές που χρειάζεται να είναι γνωστή η απόσταση του αισθητήρα, και γενικότερα της εφαρμογής από ένα πιθανό εμπόδιο ή αντικείμενο.

Η λειτουργία των αισθητήρων είναι ίδια με αυτή των Sonar και των Radar. Εκτιμούν την απόσταση ενός στόχου λαμβάνοντας υπόψη τους την αντανάκλαση ενός ραδιοκύματος ή ενός ηχητικού σήματος πάνω στο στόχο. Δημιουργούν υψηλής συχνότητας κύματα και χρησιμοποιώντας το επιστρεφόμενο σήμα καθορίζουν την απόσταση ή ακόμα και την ταχύτητα του στόχου. Για να το επιτύχουν αυτό χρησιμοποιούν τον χρόνο που έκανε το σήμα για να καλύψει την απόσταση από τον αισθητήρα στο αντικείμενο και πίσω.

Πιο συγκεκριμένα, από την πηγή του κύματος, γίνεται αποστολή ενός υπερήχου στο χώρο. Όλα τα σώματα έχουν την ιδιότητα να αντανακλούν τους ήχους που προσπίπτουν πάνω τους. Έτσι αν υπάρχει κάποιο αντικείμενο του οποίου η ηχώ του υπερήχου που στάλθηκε, αντανακλάστηκε και επιστέφει προς τον αισθητήρα, γίνετε αισθητή μέσω ενός μικροφώνου. Γνωρίζοντας το πόσο χρόνο διαρκεί το ταξίδι του υπερήχου, από την μετάδοση του έως και την λήψη του από το μικρόφωνο, μπορούμε να υπολογίσουμε πόση απόσταση διέσχισε ο υπέρηχος και διαιρώντας με το δυο μας επιστέφεται η απόσταση του αισθητήρα από το αντικείμενο.

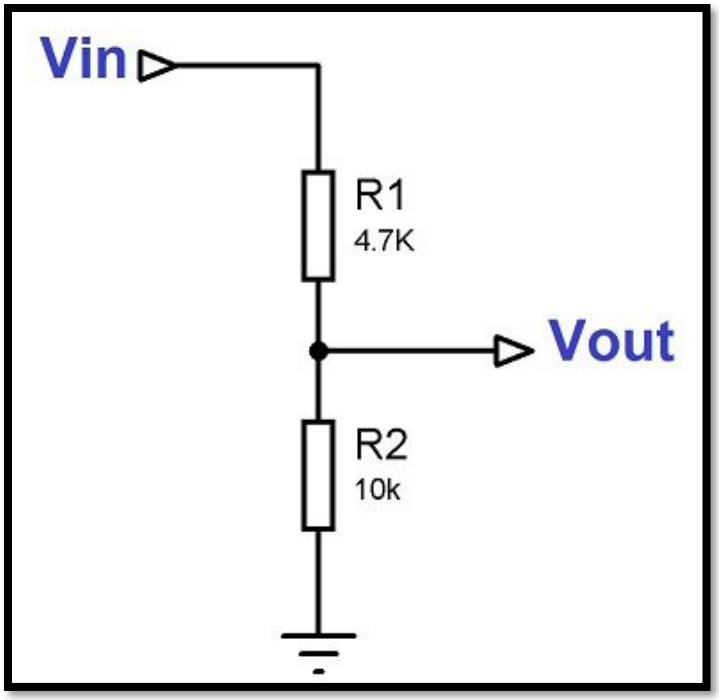
Ο αισθητήρας υπερήχων HC-SR04 παρέχει ένα εύρος μέτρησης από 2cm ως 400cm με ακρίβεια που αγγίζει τα 3mm. Το εξάρτημα αποτελείται από έναν πομπό (Trigger) και έναν δέκτη (Echo). Επίσης έχει ένα Pin για την τάση λειτουργίας του VCC καθώς και ένα για την γείωση, GND.



Εικόνα 5.8 : Παράδειγμα λειτουργίας αισθητήρα υπέρηχων HC-SR04

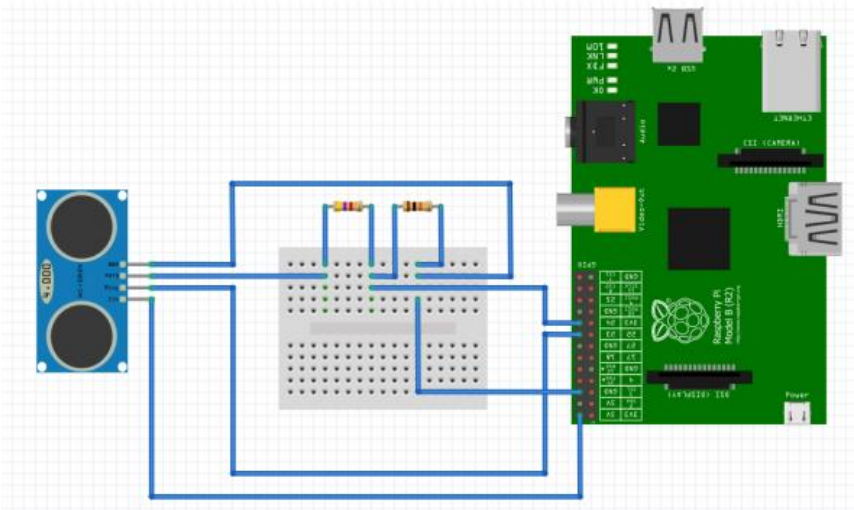
5.2.1 Αισθητήρας Υπερήχων και Raspberry Pi

Η έξοδος του ECHO Pin του αισθητήρα υπερήχων είναι 5v.Τα Input Pins του Raspberry Pi είναι 3.3v. Δεν θα ήταν σωστό να συνδέσουμε τον αισθητήρα απευθείας πάνω στα απροστάτευτα 3.3v Pins του Raspberry Pi διότι θα υπήρχε περίπτωση να προκληθεί ζημιά στην πλακέτα μας. Για αυτό ακριβώς το λόγο μπορούμε χρησιμοποιήσουμε δύο αντιστάσεις ως Διαιρέτη Τάσης.

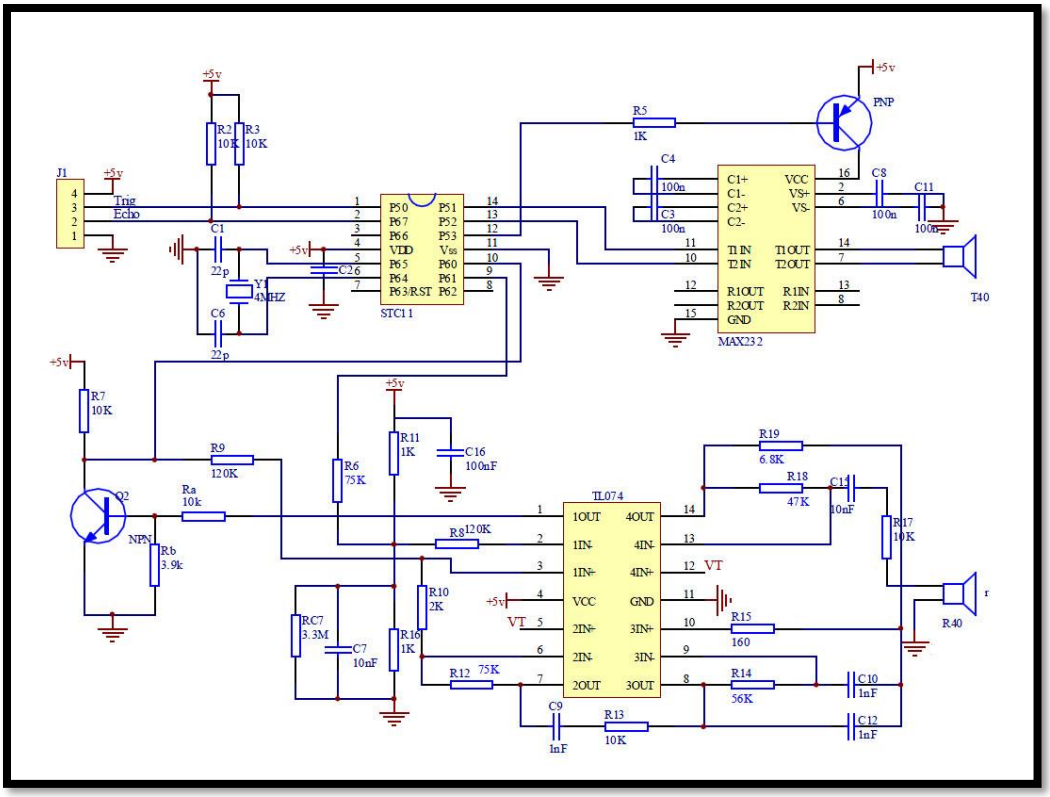


Εικόνα 5.9 : Διαιρέτης Τάσης από 5v σε 3.3v

Πιο συγκεκριμένα, για την σύνδεση του αισθητήρα υπερήχων HC-SR04 με το Raspberry Pi ας δούμε την παρακάτω εικόνα



Εικόνα 5.10: Σχηματική αναπαράσταση σύνδεσης Αισθητήρα Υπερήχων και Raspberry Pi



Εικόνα 5.11 : Σχηματικό HC-SR04

Κώδικας Python για δοκιμή του αισθητήρα υπερήχων (HC-SR04)

```
1.  import RPi.GPIO as GPIO     #Εισαγωγή GPIO library
2.  import time                 #Εισαγωγή time library
3.  GPIO.setmode(GPIO.BCM)      #Ορίζουμε το επιθυμητό GPIO pin Mode
4.
5.  TRIG = 23                   #Θέτουμε το pin 23 στο TRIG Pin
6.  ECHO = 24                   #Θέτουμε το pin 24 στο ECHO Pin
7.
8.  print "Μέτρηση απόστασης σε εξέλιξη"
9.
10. GPIO.setup(TRIG,GPIO.OUT)   #Θέτουμε το TRIG Pin (23) ως έξοδο
11. GPIO.setup(ECHO,GPIO.IN)    #Θέτουμε το ECHO Pin (24) ως είσοδο
12.
```

```
13. while True:
14.
15.   GPIO.output(TRIG, False)    #Ορίζουμε το TRIG Pin ως LOW
16.   print "Περιμένοντας τον αισθητήρα να σταθεροποιηθεί"
17.   time.sleep(2)              #Καθυστέρηση 2 δευτερολέπτων
18.   GPIO.output(TRIG, True)     #Ορίζουμε το TRIG Pin ως HIGH
19.   time.sleep(0.00001)        #Καθυστέρηση 0.00001 δευτερολέπτων
20.   GPIO.output(TRIG, False)    #Ορίζουμε το TRIG Pin ως LOW
21.
22.   while GPIO.input(ECHO)==0:  #Τσεκάρουμε πότε το ECHO Pin είναι           LOW
23.
24.     pulse_start = time.time() #Σώζουμε τον τελευταίο γνωστό χρόνο του παλμού που ήταν LOW
25.
26.   while GPIO.input(ECHO)==1:  #Τσεκάρουμε πότε το ECHO Pin είναι  HIGH
27.
28.     pulse_end = time.time()   #Σώζουμε τον τελευταίο γνωστό χρόνο του παλμού που ήταν HIGH
29.
30.   pulse_duration = pulse_end - pulse_start   #Βρίσκουμε την διάρκεια του παλμού
31.   #αποθηκεύοντας την σε μια μεταβλητή
32.
33.   distance = pulse_duration * 17150     #Πολλαπλασιάζουμε την διάρκεια του παλμού
34.   #με την τιμή 17150 για να βρούμε την απόσταση
35.
36.   distance = round(distance, 2)
37.  #Στρογγυλοποιούμε σε 2 δεκαδικά ψηφία
38.
39.   if distance > 2 and distance < 400:
40.   #Τσεκάρουμε αν η απόσταση είναι μέσα
41.   #στο εύρος του αισθητήρα
42.     print "Απόσταση:",distance - 0.5,"cm"
43.     #Εκτύπωση απόστασης με 0.5 καλιμπράρισμα
44.   else:
45.     print "Εκτός Εύρους Αισθητήρα"
```

ΚΕΦΑΛΑΙΟ 6

ΟΡΙΣΜΟΣ ΠΡΟΒΛΗΜΑΤΟΣ ΚΑΙ ΑΛΓΟΡΙΘΜΟΙ ΕΠΙΛΥΣΗΣ

Αφού το όχημα είχε σχεδιασθεί και υλοποιηθεί, είχε μείνει μόνο το προγραμματιστικό κομμάτι που θα έκανε το όχημα αυτό λειτουργικό. Το πρόβλημα που είχε δημιουργηθεί ήταν ο τρόπος με τον οποίο θα έκανα το Raspberry Pi, το οποίο αντλούσε δεδομένα από την κάμερα που ήταν συνδεδεμένη πάνω του, να μπορεί να αντιλαμβάνεται κάποια συγκεκριμένα σήματα του Κώδικα Οδικής Κυκλοφορίας.

Εικόνα 6.1: Σήμα υποχρεωτικής διακοπής πορείας

Εικόνα 6.2: Σήμα υποχρεωτικής πορείας αριστερά

Εικόνα 6.3: Σήμα υποχρεωτικής πορείας δεξιά

6.1 Αναγνώριση Κατεύθυνσης Βέλους

Αρχικά, ασχολήθηκα με τα σήματα τα οποία είχαν να κάνουν με την αλλαγή της πορείας του οχήματος, δηλαδή τα σήματα των εικόνων 6.2 και 6.3.Αφού έκανα μια έρευνα στο διαδίκτυο και δεν βρήκα κάτι που μου έδινε ένα απόλυτά σωστό αποτέλεσμα, δημιούργησα έναν δικό μου αλγόριθμο ο οποίος δίνοντας του κάποιο από τα σήματα των εικόνων 6.2 και 6.3 σαν είσοδο θα μπορούσε να αντιληφθεί με απόλυτη ακρίβεια αν το βελάκι θα ήταν αριστερό ή δεξί.

Αλγόριθμος αναγνώρισης κατεύθυνσης βέλους:

1. Ανάλυση των καρέ του βίντεο και αναζήτηση για κυκλικά σχήματα.

2. Εύρεση κυκλικού σχήματος.

3. Εύρεση κεντρικού σημείου κυκλικού σχήματος.

4. Αναζήτηση για γωνίες από το κέντρο του κύκλου και αριστερά.

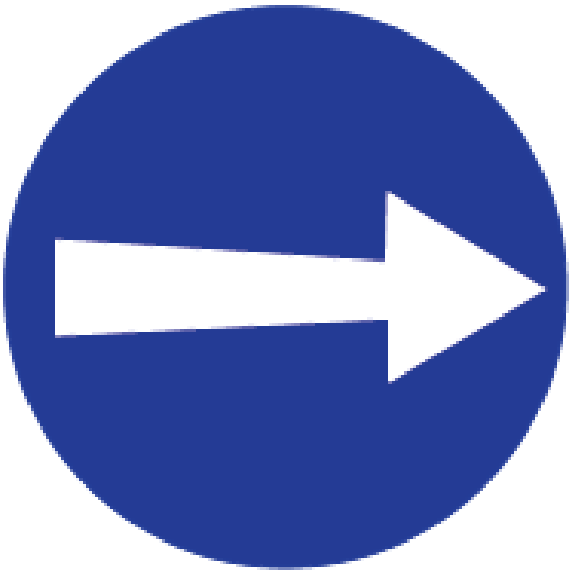
5. Αναζήτηση για γωνίες από το κέντρο του κύκλου και δεξιά.

6. Σύγκριση αριθμού γωνιών των δύο πλευρών.

7. Αποτέλεσμα σύγκρισης και συμπέρασμα.

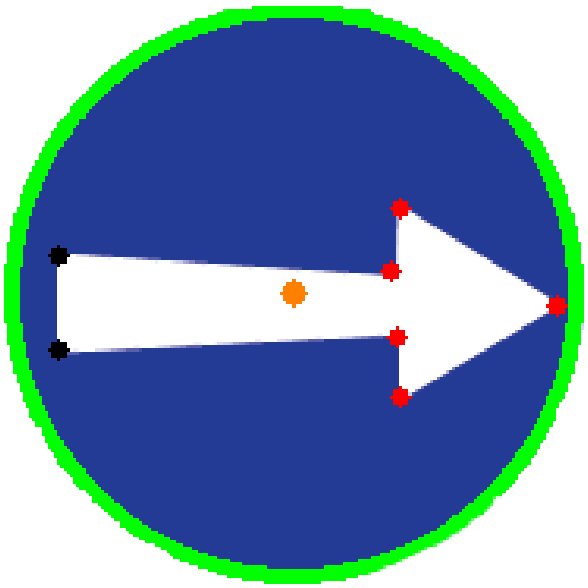
Παρακάτω θα δούμε ένα πραγματικό παράδειγμα με περισσότερες λεπτομέρειες για να δούμε καλύτερα πως λειτουργεί ο αλγόριθμος αυτός.

Ας υποθέσουμε πώς το όχημα μας συναντά μια τέτοια πινακίδα καθώς κινείται:

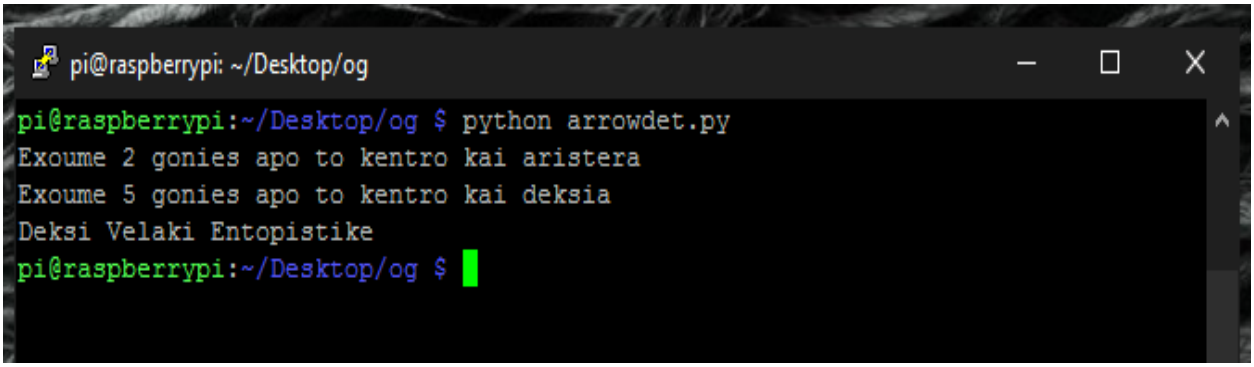


Εικόνα 6.4: Σήμα υποχρεωτικής πορείας δεξιά ως παράδειγμα

Με την εφαρμογή του αλγορίθμου που αναλύσαμε πιο πάνω αυτό θα είναι αυτό που θα «βλέπει» το όχημα μας μετά την επεξεργασία του παραπάνω σήματος.



Εικόνα 6.5: Αποτέλεσμα επεξεργασίας σήματος παραδείγματος 6.4



Εικόνα 6.6: Αποτέλεσμα στο τερματικό του Raspberry Pi

Στην εικόνα 6.5, η πορτοκαλί κουκίδα αναφέρεται στο κεντρικό σημείο του κύκλου, οι μαύρες κουκίδες αναφέρονται στις γωνίες που βρέθηκαν από το κεντρικό σημείου το κύκλου και αριστερά και τέλος, οι κόκκινες κουκίδες αναφέρονται στις γωνίες που βρέθηκαν από το κεντρικό σημείο του κύκλου και δεξιά.

Σημείωση: Τα διαφορετικά χρώματα επιλέχθηκαν για να μπορούμε να παρατηρούμε αν συμπεριφέρεται ορθά ο αλγόριθμος.

Παρακάτω θα δούμε τον κώδικα του αλγορίθμου που αναπτύχθηκε για την αναγνώριση της κατεύθυνσης του βέλους.

Python Function:

```
1.  def findcorners(img):
2.
3.      img = cv2.imread(img)          # Εικόνα προς είσοδο
4.      gray = cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)
    # Μετατροπή σε αποχρώσεις του Γκρι
5.
6.      corners = cv2.goodFeaturesToTrack(gray,7,0.05,5)
    # Δήλωση για γωνίες
7.      circles = cv2.HoughCircles(gray, cv2.HOUGH_GRADIENT,1.2, 100)
    # Δήλωση για κυκλικά σχήματα
8.      counta = 0
     # Αρχικοποίηση μεταβλητών για γωνίες από το κέντρο και αριστερά
9.      countd = 0
    # Αρχικοποίηση μεταβλητών για γωνίες από το κέντρο και δεξιά
10.     if corners is not None:
    # Σιγουρευόμαστε ότι έστω κάποιες γωνίες βρέθηκαν
11.
12.         corners = np.int0(corners)
    # Μετατροπή των συντεταγμένων των γωνιών σε ακέραιους αριθμούς
13.
```

```
14.
15.         if circles is not None:
    # Σιγουρευόμαστε ότι έστω κάποια κυκλικά σχήματα βρέθηκαν
16.
17.             circles = np.round(circles[0, :]).astype("int")
    # Μετατροπή των (x, y) συντεταγμένων και της ακτίνας του κύκλου σε ακέραιους αριθμούς
18.
19.
20.             for (x, y, r) in circles:
    # Λουπάρουμε μέσα στις (x, y) συντεταγμένες και την ακτίνα των κυκλικών σχημάτων που θ
    α βρούμε
21.
22.
23.                 cv2.circle(img, (x, y), r, (0, 255, 0), 4)
    # Σχεδίαση πράσινου κύκλου γύρω από το κυκλικό σχήμα που εντοπίστηκε
24.
25.
26.                 for i in corners:
     # Λουπάρουμε μέσα στις γωνίες
27.
28.
29.                     cv2.circle(img, (x, y), 4, (0, 128, 255), -
    1)   # Σχεδίαση κεντρικού σημείου το κύκλου με πορτοκαλί χρώμα
30.
31.                     x1,y1 = i.ravel()
    # Δήλωση συντεταγμένων για τις γωνίες
32.
33.
34.                     if x1>x and x1<x+85 and y1>y-85 and y1<y+85:
     # Έλεγχος για το άν βρίσκονται οι γωνίες από το κέντρο και δεξιά
35.
```

```
36.                     cv2.circle(img, (x1, y1), 3, (0, 0, 255), -
    1)  # Σχεδίαση σημείου γωνίας με κόκκινο χρώμα
37.                     countd=countd+1
    # Αύξηση μεταβλητής κατά ένα
38.
39.
40.                 elif x1<x and x1>x-85 and y1>y-85 and y1<y+85:
    # Έλεγχος για το άν βρίσκονται οι γωνίες από το κέντρο και αριστερά
41.
42.                     cv2.circle(img, (x1, y1), 3, (0, 0, 0), -1)
                        # Σχεδίαση σημείου γωνίας με μαύρο χρώμα
43.                     counta=counta+1
                       # Αύξηση μεταβλητής κατά ένα
44.
45.
46.   print "Exoume %d gonies apo to kentro kai aristera" % counta
47.   print "Exoume %d gonies apo to kentro kai deksia" % countd
48.
49.   cv2.imwrite("DetectedCorners.png",img)    # Αποθήκευση της εικόνας με τις γωνίες που β
    ρέθηκαν
50.
51.   if (counta+countd >= 2):              # Έλεγχος αν οι γωνίες είναι >=2
52.
53.       if (counta > countd):
54.
55.           print "Aristero Velaki Entopistike"
56.
57.
58.       elif (counta < countd):
59.
60.           print "Deksi Velaki Entopistike"
```

Η συγκεκριμένη Function όπως βλέπουμε, δέχεται ως είσοδο μια εικόνα η οποία έχει τραβηχτεί από την κάμερα που βρίσκεται στο όχημα μας, σε συγκεκριμένη απόσταση από το σήμα που βρέθηκε μπροστά του και αντιλαμβάνεται την κατεύθυνση του βέλους.

Όπως βλέπουμε στην γραμμή 6 της παραπάνω Function αναγράφεται αυτή η εντολή:

corners = cv2.goodFeaturesToTrack(gray,7,0.05,5)

Το cv2.goodFeaturesToTrack() είναι μια function της βιβλιοθήκης OpenCV και έχει την δυνατότητα να ανιχνεύει σε μια grayscale εικόνα τις γωνίες.

Παρακάτω θα δούμε ποια είναι τα κριτήρια που μπορούμε να επεξεργαστούμε για ένα καλύτερο αποτέλεσμα.

cv2.goodFeaturesToTrack(image, maxCorners, qualityLevel, minDistance[, corners[, mask[, blockSize[, useHarrisDetector[, k]]]]])

- Η παράμετρος **image** χαρακτηρίζει την εικόνα στην οποία επιλέγουμε να γίνει η ανίχνευση για γωνίες.

- Η παράμετρος **maxCorners** ωστόσο χαρακτηρίζει τον ανώτερο αριθμό των γωνιών που μπορούν να βρεθούν σε μια εικόνα

- Η παράμετρος **qualityLevel** χαρακτηρίζει την ελάχιστη αποδεκτή ποιότητα των γωνιών μιας εικόνας . Για παράδειγμα αν η καλύτερη γωνία έχει ως ανάλυση τιμή 1500 και το qualityLevel που έχει δηλωθεί είναι 0,01 τότε όλες οι γωνίες με τιμή μικρότερη από 15 απορρίπτονται.

- Η παράμετρος **minDistance** χαρακτηρίζει την ελάχιστη δυνατή Ευκλείδεια απόσταση μεταξύ των γωνιών.

- Η παράμετρος **corners** χαρακτηρίζει το διάνυσμα των γωνιών που έχουν ανιχνευτεί.

- Η παράμετρος **mask** χαρακτηρίζει την περιοχή ενδιαφέροντος. Αν η εικόνα δεν είναι κενή, καθορίζει την περιοχή που έχουν ανιχνευτεί γωνίες.

- Η παράμετρος **useHarrisDetector** μας δίνει την δυνατότητα να επιλέξουμε αν θέλουμε να χρησιμοποιήσουμε το Harris Detector ή όχι .

- Η παράμετρος **k** είναι μια ελεύθερη παράμετρος του Harris Detector .

Στη γραμμή 7 της παραπάνω Function αναγράφεται αυτή η εντολή:

circles = cv2.HoughCircles(gray, cv2.HOUGH_GRADIENT,1.2, 100)

Το cv2.HoughCircles() είναι μια function της βιβλιοθήκης OpenCV και έχει την δυνατότητα να ανιχνεύει σε μια grayscale εικόνα κυκλικά σχήματα.

cv2.HoughCircles(image, method, dp, minDist[, circles[, param1[, param2[, minRadius[, maxRadius]]]]])

- Η παράμετρος **image** χαρακτηρίζει την εικόνα στην οποία επιλέγουμε να γίνει η ανίχνευση για κυκλικά σχήματα.

- Η παράμετρος **method** χαρακτηρίζει την μέθοδο ανίχνευσης που θα χρησιμοποιήσουμε. Μέχρι τώρα υπάρχει η CV_HOUGH_GRADIENT.

- Η παράμετρος **dp** χαρακτηρίζει την αντίστροφη αναλογία της συσσωρευμένης ανάλυσης με την ανάλυση της εικόνας

- Η παράμετρος **minDist** χαρακτηρίζει την ελάχιστη απόσταση των κεντρικών σημείων των κυκλικών σχημάτων που έχουν ανιχνευτεί.

- Η παράμετρος **circles** χαρακτηρίζει τα διανύσματα εξόδου των κυκλικών σχημάτων που έχουν βρεθεί. Κάθε διάνυσμα έχει 3 στοιχεία.(x , y , Radius)

- Η παράμετρος **minRadius** χαρακτηρίζει την ελάχιστη ακτίνα κύκλου

- Η παράμετρος **maxRadius** χαρακτηρίζει την μέγιστη ακτίνα κύκλου

6.2 Αναγνώριση σήματος STOP

Αφού ο αλγόριθμος αναγνώρισης κατεύθυνσης βέλους είχε φτάσει σε ένα ικανοποιητικό επίπεδο και είχε την δυνατότητα να ανιχνεύει τον προσανατολισμό δίχως λάθη, προχώρησα στο επόμενο βήμα που ήταν η αναγνώριση του σήματος υποχρεωτικής διακοπής πορείας, δηλαδή του σήματος STOP.

Εικόνα 6.7: Σήμα υποχρεωτικής διακοπής πορείας

Υπάρχουν πάρα πολλοί τρόποι για να εντοπίσουμε μια πινακίδα STOP. Για παράδειγμα θα μπορούσαμε να την εντοπίσουμε χρησιμοποιώντας το σχήμα του και το χρώμα του, δηλαδή να χρησιμοποιούσαμε τις 8 πλευρές του και να ψάχναμε για ένα κόκκινο σχήμα με 8 πλευρές. Ένας άλλος τρόπος, θα ήταν να γινόταν οπτική αναγνώριση χαρακτήρων η οποία θα αναγνώριζε τα 4 γράμματα S – T – O – P. Άλλος τρόπος θα ήταν το Template Matching (Ταίριασμα με το πρότυπο) στον οποίο θα μπορούσαμε να δώσουμε ως πρότυπο την εικόνα 5.6 για να γίνει ο εντοπισμός του συγκεκριμένου σήματος.

Προσωπικά, προτίμησα να χρησιμοποιήσω ένα εργαλείο που παρέχει η OpenCV βιβλιοθήκη με την εγκατάσταση της. Το συγκεκριμένο εργαλείο χρησιμοποιείται για την εκπαίδευση ταξινομητών με ένα δεδομένο σύνολο θετικών και αρνητικών δειγμάτων, που αφορούν ένα συγκεκριμένο αντικείμενο. Το εργαλείο αυτό παράγει ένα xml αρχείο που περιέχει τα δεδομένα που αφορούν τον εν λόγω ταξινομητή και μπορεί να χρησιμοποιηθεί στη συνέχεια για την ανίχνευση του συγκεκριμένου αντικειμένου.

Αφού εκπαιδευτεί ο ταξινομητής, μπορεί να εφαρμοστεί σε μία περιοχή ενδιαφέροντος μιας εικόνας εισόδου. Ο ταξινομητής αποκρίνεται με "1" αν η περιοχή περιέχει το αναζητούμενο αντικείμενο και με "0" σε αντίθετη περίπτωση. Για να αναζητηθεί το αντικείμενο σε ολόκληρη την εικόνα, το παράθυρο αναζήτησης μπορεί να μμετακινηθεί σε διαφορετικές θέσεις και να ελεγχθεί η απόκριση του ταξινομητή. Ο ταξινομητής έχει σχεδιαστεί έτσι ώστε το παράθυρο αναζήτησης να μπορεί να τροποποιηθεί ως προς τις διαστάσεις του εύκολα, για να ανευρεθούν αντικείμενα διαφορετικών διαστάσεων, αντί να απαιτείται αντίστοιχη μεταβολή των διαστάσεων της ίδιας της εικόνας. Έτσι, για να βρεθεί ένα αντικείμενο άγνωστων διαστάσεων σε μία εικόνα, η διαδικασία αναζήτησης μπορεί να γίνει όσες φορές χρειάζεται σε διαφορετικές κλίμακες.

Ο τελικός ταξινομητής αποτελείται από επίπεδα απλούστερων ταξινομητών σε διάταξη καταρράκτη (cascade). Κάθε επίπεδο του καταρράκτη αποτελείται από ένα ισχυρό ταξινομητή (strong classifier) που κατασκευάζεται από απλούστερους αδύναμους ταξινομητές (weak classifiers) με την τεχνική της ενίσχυσης (boosting). Οι τεχνικές ενίσχυσης που υποστηρίζονται είναι η Discrete AdaBoost, η Real AdaBoost, η Gentle AdaBoost και η LogitBoost. Οι αδύναμοι ταξινομητές που χρησιμοποιούνται είναι δένδρα απόφασης (decision trees) με 1 έως 4 κόμβους. Οι αδύναμοι ταξινομητές παίρνουν ως είσοδο ορθογώνια χαρακτηριστικά (features) τύπου Haar, που καθορίζονται από την θέση, τις διαστάσεις, τον προσανατολισμό και την μορφή τους και παίρνουν τιμές μία σταθμισμένη διαφορά των εντάσεων των εικονοστοιχείων στις περιοχές που ορίζουν. Τα αθροίσματα των εντάσεων υπολογίζονται με τη χρήση της εικόνας ολοκλήρωμα (integral image) που έχει προϋπολογιστεί.

Έτσι μια χαμηλού επιπέδου ρουτίνα, για παράδειγμα, παρέχει τη δυνατότητα στον χρήστη να ελέγξει μία συγκεκριμένη περιοχή της εικόνας για το αν περιέχει ή όχι πρόσωπο. Βοηθητικές ρουτίνες υπολογίζουν τις εικόνες ολοκληρώματα, και ρυθμίζουν την κλίμακα αναζήτησης του ταξινομητή για πρόσωπα διαφορετικών μεγεθών κλπ.

Αντίθετα, μία υψηλού επιπέδου συνάρτηση, όπως η cvHaarDetectObjects κάνει όλα τα προηγούμενα, χρησιμοποιώντας τον ταξινομητή που είναι αποθηκευμένος σε xml αρχείο, παρέχοντας αυτόματα τα αποτελέσματα ανίχνευσης στο χρήστη, πράγμα που είναι τις περισσότερες φορές αρκετό.

6.2.1 Εκπαίδευση Ταξινομητή (Haar Classifier Training)

Προκειμένου να εκπαιδεύσουμε τον δικό μας ταξινομητή χρειαζόμαστε δείγματα, πράγμα που σημαίνει ότι χρειαζόμαστε πολλές εικόνες που δείχνουν το αντικείμενο που θέλουμε να ανιχνεύσουμε (Θετικά Δείγματα) και ακόμα περισσότερες εικόνες που **δεν** περιέχουν μέσα το αντικείμενο αυτό. (Αρνητικά Δείγματα)

Ο αριθμός τον εικόνων που χρειαζόμαστε για να καταφέρουμε να εκπαιδεύσουμε έναν δικό μας ταξινομητή εξαρτάται από μία ποικιλία παραγόντων, συμπεριλαμβανομένης της ποιότητας των εικόνων, το αντικείμενο που θέλουμε να αναγνωρίσουμε, την μέθοδο για την παραγωγή των δειγμάτων και την δύναμη της CPU που διαθέτουμε.

Η εκπαίδευση ενός εξαιρετικά ακριβή ταξινομητή παίρνει πάρα πολύ χρόνο και χρειάζεται έναν τεράστιο αριθμό από δείγματα.

POSITIVE IMAGES (ΘΕΤΙΚΑ ΔΕΙΓΜΑΤΑ)

Για να έχουμε σωστά θετικά δείγματα χρειάζεται να τραβήξουμε φωτογραφίες από το αντικείμενο που θέλουμε να ανιχνεύσουμε ή να βρούμε εικόνες από το διαδίκτυο(σε καλή ποιότητα κατά προτίμηση). Μπορούμε ακόμα και να εξάγουμε εικόνες από διάφορα βίντεο που απεικονίζονται. Απλά χρειαζόμαστε από 40 ως 100 από αυτά. Όσο μεγαλύτερος ο αριθμός των θετικών δειγμάτων σε διαφορετικούς φωτισμούς και φόντο τόσο μεγαλύτερη θα είναι η επιτυχία και ακρίβεια του ταξινομητή μας.

Αφού έχουμε συλλέξει τις φωτογραφίες πρέπει να τις περικόψουμε μία – μία ώστε μόνο το επιθυμητό αντικείμενο να είναι ορατό. Θα πρέπει να προσέξουμε πως οι αναλογίες των περικομμένων δεν θα πρέπει να διαφέρουν κατά πολύ μεταξύ τους. Τα καλύτερα αποτελέσματα προέρχονται κυρίως από θετικά δείγματα που απεικονίζουν ακριβώς το επιθυμητό αντικείμενο.

Εικόνα 6.8: Παράδειγμα συλλογής θετικών δειγμάτων

Ενώ έχει γίνει η συλλογή των θετικών δειγμάτων, τις τοποθετούμε στον φάκελο ./positive_images που βρίσκεται στον φάκελο εγκατάστασης της OpenCV βιβλιοθήκης στο λειτουργικό σύστημά μας.

Με την παρακάτω εντολή δημιουργείτε ένα text file με την ονομασία **positives.txt** το οποίο περιέχει μια λίστα με τις διαδρομές από όλα τα θετικά δείγματα που συλλέξαμε.

find ./positive_images -iname "*.jpg" > positives.txt

NEGATIVE IMAGES (ΑΡΝΗΤΙΚΑ ΔΕΙΓΜΑΤΑ)

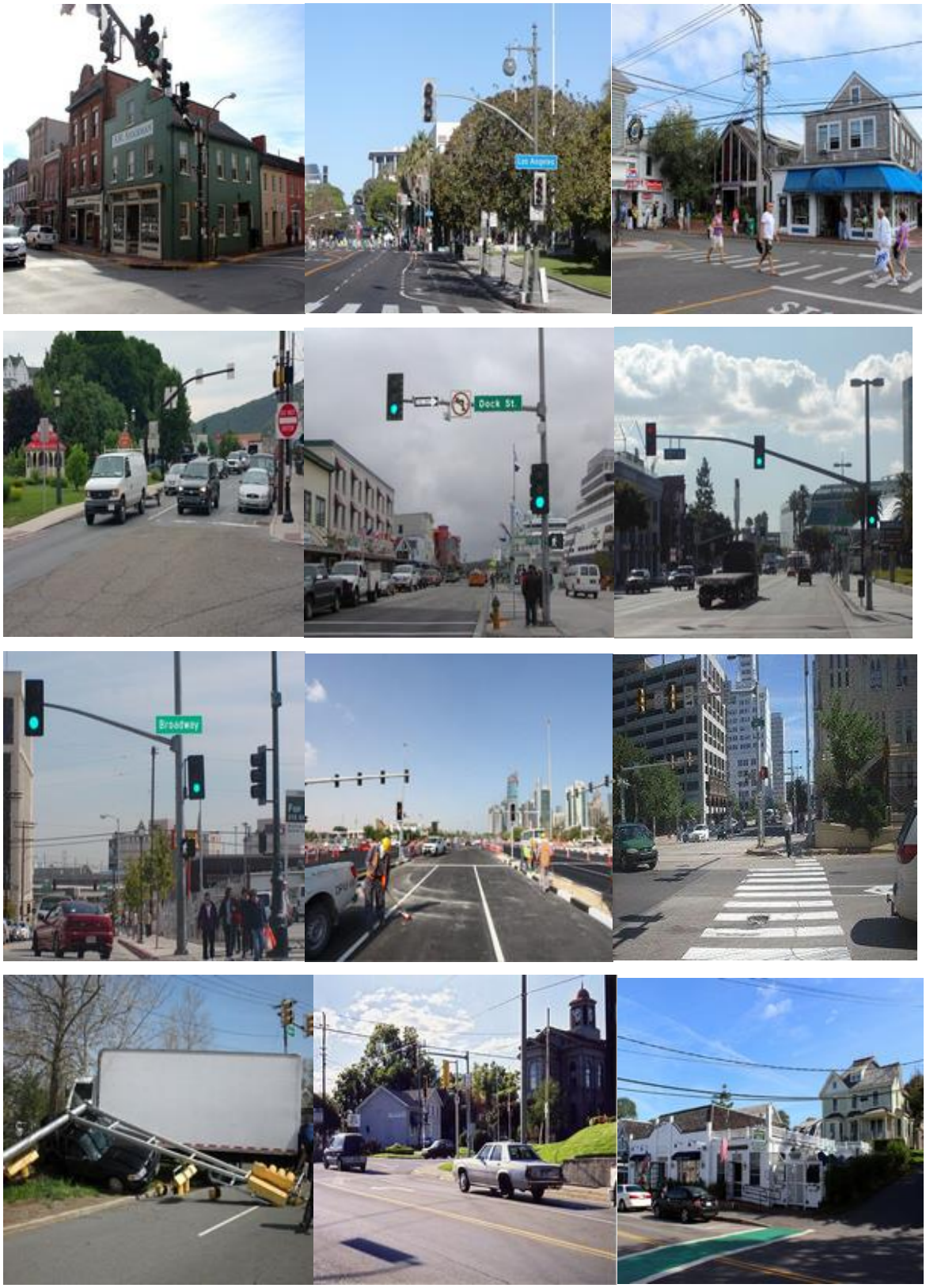
Τώρα χρειαζόμαστε και τις αρνητικές εικόνες, τις εικόνες αυτές που δεν περιέχουν κάποιον σηματοδότη STOP. Για την καλύτερη ακρίβεια του ταξινομητή μας αυτό που θα μπορούσαμε να κάνουμε θα ήταν να συλλέγαμε αρνητικές εικόνες οι οποίες να είναι σχεδόν ίδιες με τις θετικές απλά χωρίς τον σηματοδότη STOP που θέλουμε να αναγνωρίσουμε.

Χρειαζόμαστε τουλάχιστον 600 από αυτές. Είναι μια πολύ χρονοβόρα διαδικασία, αλλά για το καλύτερο δυνατό αποτέλεσμα θα πρέπει να συλλέξουμε όσο το δυνατό περισσότερες από αυτές.

Ενώ έχει γίνει η συλλογή των αρνητικών δειγμάτων, τις τοποθετούμε στον φάκελο ./negative_images που βρίσκεται στον φάκελο εγκατάστασης της OpenCV βιβλιοθήκης στο λειτουργικό σύστημά μας.

Με την παρακάτω εντολή δημιουργείτε ένα text file με την ονομασία **negatives.txt** το οποίο περιέχει μια λίστα με τις διαδρομές από όλα τα αρνητικά δείγματα που συλλέξαμε.

find ./negative_images -iname "*.jpg" > negatives.txt



Εικόνα 6.9: Παράδειγμα συλλογής αρνητικών δειγμάτων

ΔΗΜΙΟΥΡΓΩΝΤΑΣ ΔΕΙΓΜΑΤΑ

Έχοντας συλλέξει τα θετικά και τα αρνητικά δείγματα για τον ταξινομητή μας, είμαστε έτοιμοι για να ξεκινήσουμε με την παραγωγή των δειγμάτων από αυτά.

Είμαστε έτοιμη να χρησιμοποιήσουμε ένα εργαλείο που μας παρέχει επίσης η βιβλιοθήκη OpenCV το **opencv_createsamples.** Το εργαλείο αυτό μας προσφέρει διάφορες επιλογές όπως, πως θα γίνει η παραγωγή των δειγμάτων δημιουργώντας ένα ***.vec** αρχείο το οποίο θα χρησιμοποιήσουμε αργότερα για την εκπαίδευση του ταξινομητή μας.

Το **opencv_createsamples** παράγει έναν τεράστιο αριθμό από θετικά δείγματα από τις θετικές εικόνες που συλλέξαμε χρησιμοποιώντας μετασχηματισμούς και στρεβλώσεις στις ήδη υπάρχουσες εικόνες.

Ο **Naotoshi Seo** δημιούργησε κάποια χρήσιμα scripts τα οποία βοηθούν κατά πολύ στην διαδικασία παραγωγής θετικών δειγμάτων. Αρχικά χρησιμοποιήσαμε το **createsamples.pl** ένα μικρό script γραμμένο στην γλώσσα προγραμματισμού Perl, το οποίο παρήγαγε 1500 θετικά δείγματα συνδυάζοντας κάθε θετικό δείγμα με ένα τυχαίο αρνητικό. Έπειτα, τρέξαμε όλα αυτά τα δείγματα μέσα στο **opencv_createsamples.**

Αυτό γίνεται με την χρήση της παρακάτω εντολής :

```
perl bin/createsamples.pl positives.txt negatives.txt samples 1500\
  "opencv_createsamples -bgcolor 0 -bgthresh 0 -maxxangle 1.1\
  -maxyangle 1.1 maxzangle 0.5 -maxidev 40 -w 80 -h 40"
```

Το επόμενο πράγμα που έχουμε να κάνουμε είναι να συγχωνεύσουμε τα *.vec αρχεία που βρίσκονται στον φάκελο **samples**. Για να γίνει αυτό πρέπει να αρχικά να φτιάξουμε μια λίστα

από αυτά και έπειτα να χρησιμοποιήσουμε ένα ακόμη εργαλείο με την ονομασία mergevec.cpp το οποίο συνδυάζει όλα τα *.vec αρχεία σε ένα.

Για να χρησιμοποιήσουμε το συγκεκριμένο εργαλείο θα πρέπει αρχικά να το αντιγράψουμε μέσα στον φάκελο του OpenCV και να το κάνουμε compile μαζί με τα υπόλοιπα αρχεία της βιβλιοθήκης.

Αυτό γίνεται χρησιμοποιώντας τις παρακάτω εντολές.

```
cp src/mergevec.cpp ~/opencv-2.4.5/apps/haartraining
cd ~/opencv-2.4.5/apps/haartraining
g++ `pkg-config --libs --cflags opencv` -I. -o mergevec mergevec.cpp\
  cvboost.cpp cvcommon.cpp cvsamples.cpp cvhaarclassifier.cpp\
  cvhaartraining.cpp\
  -lopencv_core -lopencv_calib3d -lopencv_imgproc -lopencv_highgui -lopencv_objdetect
```

Έπειτα πάμε πίσω στο Repository και εκτελούμε το mergevec.

```
find ./samples -name '*.vec' > samples.txt
./mergevec samples.txt samples.vec
```

Αφού έχουν εκτελεστεί οι παραπάνω εντολές με επιτυχία, μπορούμε να χρησιμοποιήσουμε το samples.vec αρχείο για την εκπαίδευση του ταξινομητή μας.

Παρατήρηση: Τα Perl files δεν περιλαμβάνονται στην εγκατάσταση της OpenCV βιβλιοθήκης. Μπορούμε να τα βρούμε στον παρακάτω σύνδεσμο

https://github.com/mrnugget/opencv-haar-classifier-training

ΕΚΠΑΙΔΕΥΣΗ ΤΟΥ ΤΑΞΙΝΟΜΗΤΗ

Η OpenCV βιβλιοθήκη μας προσφέρει δύο διαφορετικές εφαρμογές για την εκπαίδευση ενός ταξινομητή, την **opencv_traincascade** και την **opencv_haartraining**. Εμείς χρησιμοποιήσαμε την **opencv_traincascade** αφού μας δίνει την δυνατότητα να γίνει η εκπαίδευση του ταξινομητή χρησιμοποιώντας πολλαπλούς πυρήνες του συστήματός μας, το οποίο θα μας γλίτωνε πολύ χρόνο. Επίσης μας δίνει την δυνατότητα να μπορούμε να χρησιμοποιήσουμε τον ταξινομητή μας σε διάφορες εκδόσεις της OpenCV δίχως κανένα πρόβλημα.

Με την παρακάτω εντολή δίνουμε σαν είσοδο στην εφαρμογή **opencv_traincascade** τα θετικά δείγματα (samples.vec) και τα αρνητικά (negatives.txt) και κάποιες άλλες ρυθμίσεις όπως, τι ποσοστό της μνήμης να χρησιμοποιήσει το σύστημα κατά τη διάρκεια της εκπαίδευσης του ταξινομητή και διάφορα άλλα.

```
opencv_traincascade -data classifier -vec samples.vec -bg negatives.txt\
  -numStages 20 -minHitRate 0.999 -maxFalseAlarmRate 0.5 -numPos 1000\
  -numNeg 600 -w 80 -h 40 -mode ALL -precalcValBufSize 1024\
  -precalcIdxBufSize 1024
```

Αφού εκτελέστηκε η εντολή, θα χρειαστεί πάρα πολύ ώρα για να φτάσει στο τέλος της. Επίσης η μνήμη και η CPU του υπολογιστή θα είναι απασχολημένες για όλη αυτή την διάρκεια. Πιο κάτω θα δούμε κάποιες εικόνες με την έξοδο του training program που μας προσφέρει η OpenCV.

```
===== TRAINING 0-stage =====
<BEGIN
POS count : consumed   1000 : 1000
NEG count : acceptanceRatio    600 : 1
Precalculation time: 11
+----+---------+---------+
|  N |    HR   |    FA   |
+----+---------+---------+
|   1|        1|        1|
+----+---------+---------+
|   2|        1|        1|
+----+---------+---------+
|   3|        1|        1|
+----+---------+---------+
|   4|        1|        1|
+----+---------+---------+
|   5|        1|        1|
+----+---------+---------+
|   6|        1|        1|
+----+---------+---------+
|   7|        1| 0.711667|
+----+---------+---------+
|   8|        1|     0.54|
+----+---------+---------+
|   9|        1|    0.305|
+----+---------+---------+
END>
Training until now has taken 0 days 3 hours 19 minutes 16 seconds.
```

Εικόνα 6.10: Έξοδος προγράμματος εκπαίδευσης ταξινομητή.

Κάθε γραμμή αντιπροσωπεύει ένα χαρακτηριστικό που εκπαιδεύεται και περιέχει κάποιες πληροφορίες για το HitRatio και το FalseAlarm ratio. Αν κάποιο επίπεδο εκπαίδευσης περιέχει λίγα χαρακτηριστικά τότε κάτι πηγαίνει λάθος με τα δεδομένα που χρησιμοποιούμε για την εκπαίδευση.

Αν όλα έχουν πάει καλά μέχρι το τέλος, ο ταξινομητής θα είναι αποθηκευμένος με την ονομασία **classifier.xml** στον φάκελο **classifier** και είναι έτοιμος για να τον χρησιμοποιήσουμε και να ανιχνεύσουμε STOP σηματοδότες.

Python Code:

```
1.  import cv2  #Δήλωση εισαγωγής βιβλιοθήκης OpenCV
2.  import numpy as np  #Δήλωση εισαγωγής βιβλιοθήκης Numpy
3.
4.
5.  stop_cascade = cv2.CascadeClassifier('xml/stop_sign.xml')
6.  #Δήλωση ταξινομητή,τοποθεσία xml file
7.
8.  img = cv2.imread('stop.jpg') φόρτωμα εικόνας
9.  gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY) #Μετατροπή σε Grayscale
10. stops = stop_cascade.detectMultiScale(gray, 1.1)
11. #Κλήση συνάρτησης detectMultiScale
12.
13. for (x,y,w,h) in stops:
14.     text_color = (0,255,0)  #Χρώμα γραμμάτων
15.     cv2.putText(img, "STOP SIGN",(x,y-10),
        cv2.FONT_HERSHEY_PLAIN, 1.0, text_color, thickness=1)
16.     #Κλήση συνάρτηστης putText για την ένδειξη "STOP"
17.
18.     cv2.rectangle(img,(x,y),(x+w,y+h),(0,255,0),2)
19.     #Κλήση συνάρτηστης rectangle για την ένδειξη "STOP"
20.
21. print "Found {0} STOP sign(s)!".format(len(stops))
22.
23. for (x, y, w, h) in stops:
24.     print "STOP sign at %d , %d" % (x + (w / 2), y + (h / 2))
25.     #Εύρεση συντεταγμένων ανιχνευμένων STOP
26.
```

```
27.    cv2.imwrite('detected_stop.jpg',img)
28.    #Αποθήκευση εικόνας με ανιχνευμένα STOP
```

Στη γραμμή 10 του παραπάνω κώδικα αναγράφεται αυτή η εντολή:

stops = stop_cascade.detectMultiScale(gray, 1.1)

Η **detectMultiScale** είναι μια Function της βιβλιοθήκης OpenCV η οποία μας δίνει την δυνατότητα να εντοπίζουμε αντικείμενα διαφόρων μεγεθών σε μια εικόνα. Τα ανιχνευμένα αντικείμενα επιστρέφουν σε μάς ως μια λίστα από ορθογώνια.

cv2.CascadeClassifier.detectMultiScale(image[, scaleFactor[, minNeighbors[, flags[, minSize[, maxSize]]]]])

- Η παράμετρος **image** χαρακτηρίζει την εικόνα στην οποία επιλέγουμε να γίνει η ανίχνευση για αντικείμενα.
- Η παράμετρος **scaleFactor** χαρακτηρίζει κατά πόσο πρέπει να ελαττωθεί το μέγεθος της εικόνας σε κάθε scale.
- Η παράμετρος **minNeighbors** χαρακτηρίζει πόσοι «γείτονες» θα πρέπει να υπάρχουν για να διατηρήσουν τον εντοπισμό του ορθογωνίου.
- Η παράμετρος **flags** είναι μια παράμετρος που έχει το ίδιο νόημα με μια παλαιότερη Function την cvHaarObjects. Δεν μπορεί να χρησιμοποιηθεί για καινούρια Cascades.

- Η παράμετρος **minSize** χαρακτηρίζει το ελάχιστο μέγεθος του αντικειμένου. Τα αντικείμενα μικρότερού μεγέθους από αυτό, αγνοούνται.

- Η παράμετρος **maxSize** χαρακτηρίζει το μέγιστο μέγεθος του αντικειμένου. Τα αντικείμενα μεγαλύτερου μεγέθους από αυτό, αγνοούνται.

Στη γραμμή 15 του παραπάνω κώδικα αναγράφεται αυτή η εντολή:

cv2.putText(img,"STOP SIGN",(x,y-10),cv2.FONT_HERSHEY_PLAIN, 1.0, text_color, thickness=1)

Η **cv2.putText** είναι μια Function της βιβλιοθήκης OpenCV η οποία μας δίνει την δυνατότητα να τοποθετούμε ένα string από λέξεις μέσα στην εικόνα

Python: cv2.putText(img, text, org, fontFace, fontScale, color[, thickness[, lineType[, bottomLeftOrigin]]])

- Η παράμετρος **img** χαρακτηρίζει την εικόνα στην οποία επιλέγουμε να γίνει τοποθέτηση του Text String.

- Η παράμετρος **text** χαρακτηρίζει τι θα περιέχει το Text String.

- Η παράμετρος **org** χαρακτηρίζει την αριστερή κάτω γωνία του Text String.

- Η παράμετρος **fontFace** χαρακτηρίζει τον τύπο της γραμματοσειράς.

- Η παράμετρος **fontScale** χαρακτηρίζει τον συντελεστή κλίμακας που πολλαπλασιάζεται με το μέγεθος της βάσης της γραμματοσειράς.

- Η παράμετρος **color** χαρακτηρίζει το χρώμα της γραμματοσειράς.

- Η παράμετρος **thickness** χαρακτηρίζει το πάχος της γραμματοσειράς.

- Η παράμετρος **lineType** χαρακτηρίζει τον τύπο των γραμμών.

- Η παράμετρος **bottomLeftOrigin** όταν είναι αληθής, η προέλευση των δεδομένων της εικόνας είναι από την κάτω αριστερά γωνία, διαφορετικά από την πάνω αριστερά γωνία.

Στη γραμμή 18 του παραπάνω κώδικα αναγράφεται αυτή η εντολή:

cv2.rectangle(img,(x,y),(x+w,y+h),(0,255,0),2)

Η **cv2.rectangle** είναι μια Function της βιβλιοθήκης OpenCV η οποία μας δίνει την δυνατότητα να σχεδιάζουμε ένα ορθογώνιο μέσα στην εικόνα.

Python: cv2.rectangle(img, pt1, pt2, color[, thickness[, lineType[, shift]]])

- Η παράμετρος **img** χαρακτηρίζει την εικόνα στην οποία επιλέγουμε να γίνει τοποθέτηση του ορθογωνίου.

- Η παράμετρος **pt1** χαρακτηρίζει την μια κορυφή του ορθογωνίου.

- Η παράμετρος **pt2** χαρακτηρίζει την άλλη κορυφή του ορθογωνίου.

- Η παράμετρος **color** χαρακτηρίζει το χρώμα του ορθογωνίου.

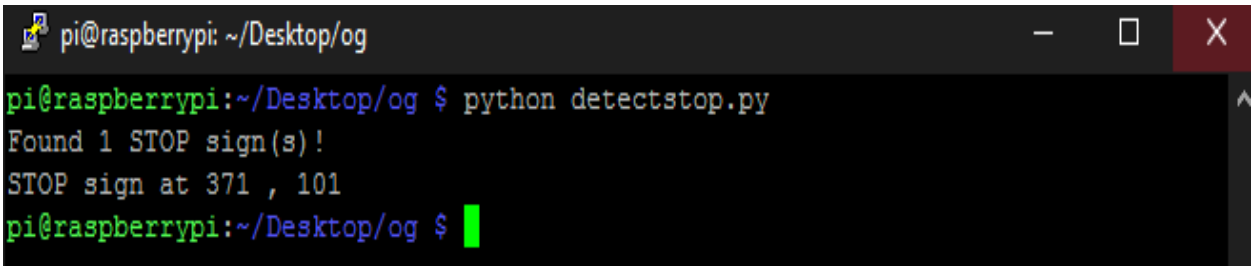
- Η παράμετρος **thickness** χαρακτηρίζει το πάχος των γραμμών του ορθογωνίου.

- Η παράμετρος **lineType** χαρακτηρίζει τον τύπο των γραμμών του ορθογωνίου.

- Η παράμετρος **shift** χαρακτηρίζει τον κλασματικό αριθμό των bits στο σημείο των συντεταγμένων.

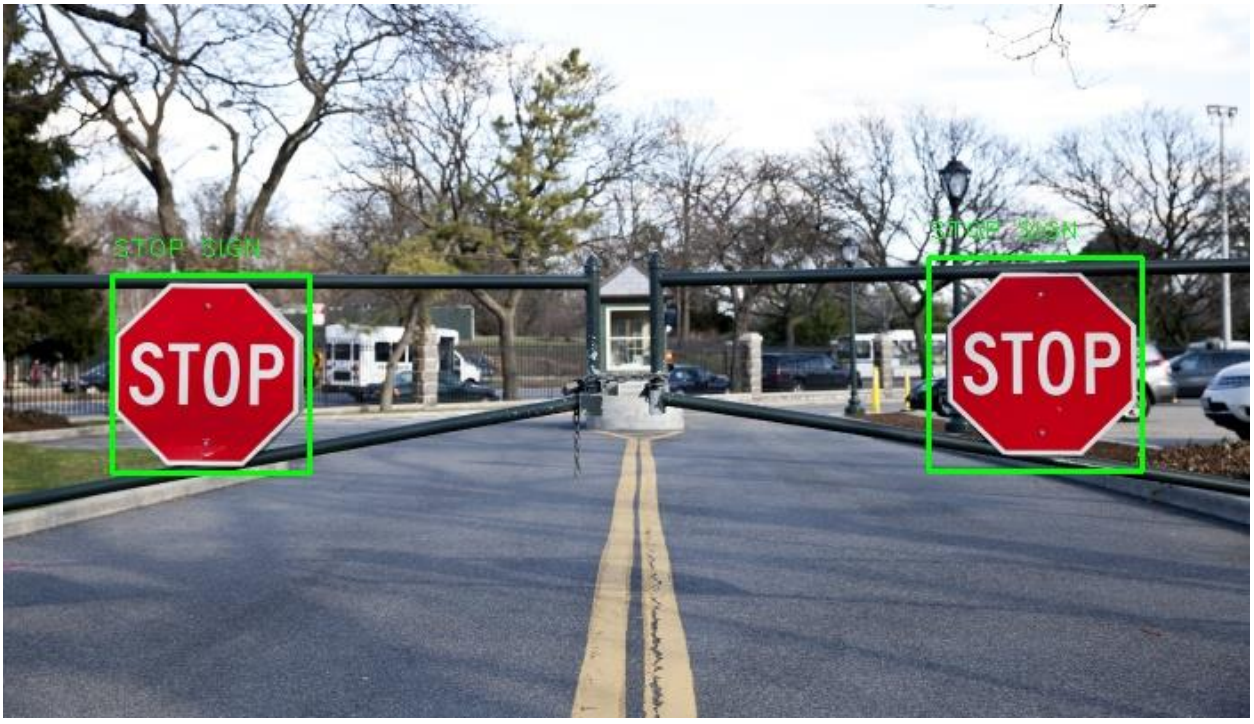
Εικόνα 6.11: Εικόνα εισόδου για τον παραπάνω κώδικα (Παράδειγμα 1ο)

Εικόνα 6.12: Εικόνα εξόδου για τον παραπάνω κώδικα (Παράδειγμα 1ο)

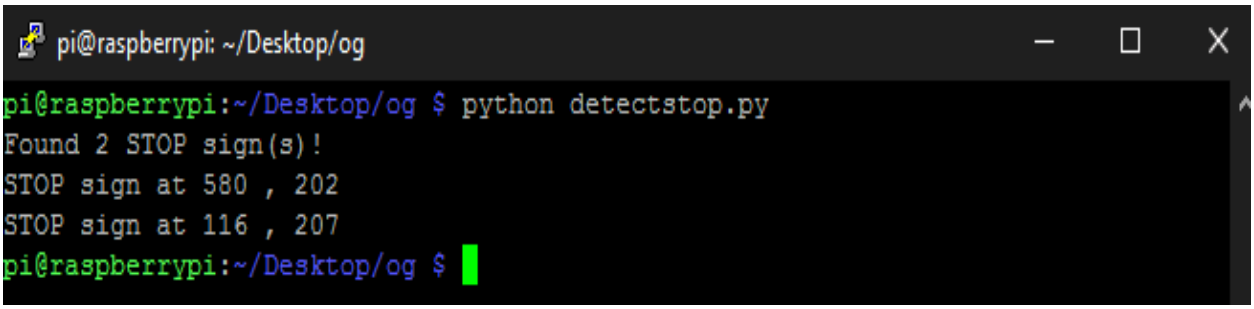


Εικόνα 6.13: Έξοδος κώδικα στο τερματικό (Παράδειγμα 1ο)

Εικόνα 6.14: Εικόνα εισόδου με πολλαπλή σήματα STOP (Παράδειγμα 2ο)



Εικόνα 6.15: Εικόνα εισόδου με πολλαπλά σήματα STOP (Παράδειγμα 2ο)



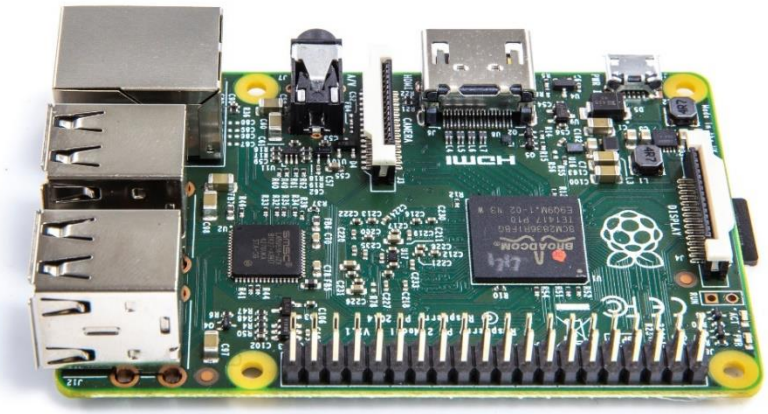
Εικόνα 6.16: Έξοδος κώδικα στο τερματικό (Παράδειγμα 2ο)

ΚΕΦΑΛΑΙΟ 7

ΛΕΠΤΟΜΕΡΕΙΕΣ ΕΓΚΑΤΑΣΤΑΣΗΣ

7.1 Λίστα Υλικών

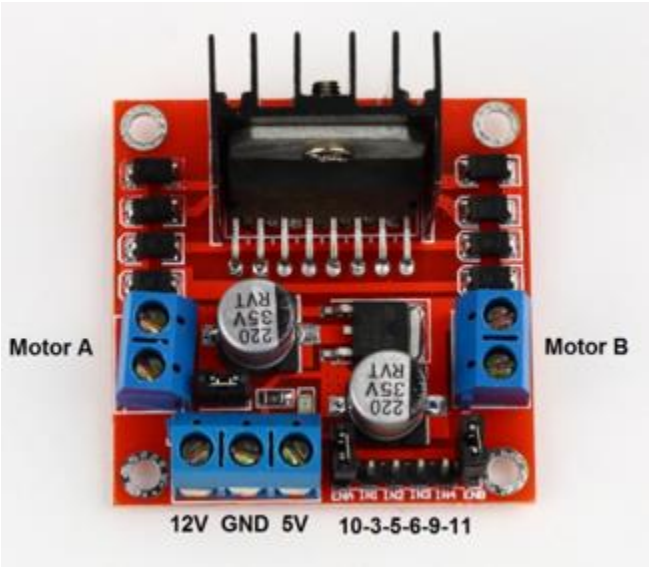
- Raspberry Pi 2 (Model B)



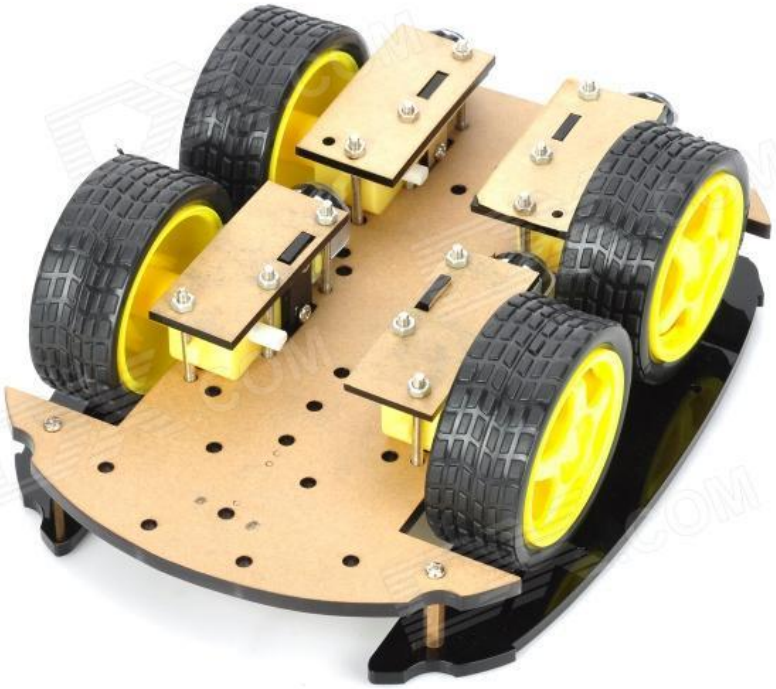
- Power Bank(5V 2.1A)

- Camera Module - Logitech C625

- Motor Shield (LN298N)



- 4WD Robot Smart Car Chassis Kit



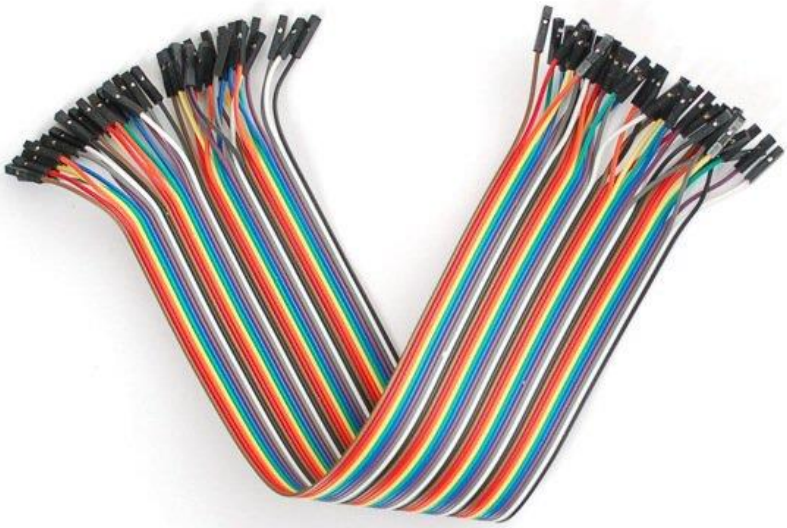
- 2x Αισθητήρες Υπερήχων (HC-SR04)

- 4x Πορτοκαλί LED

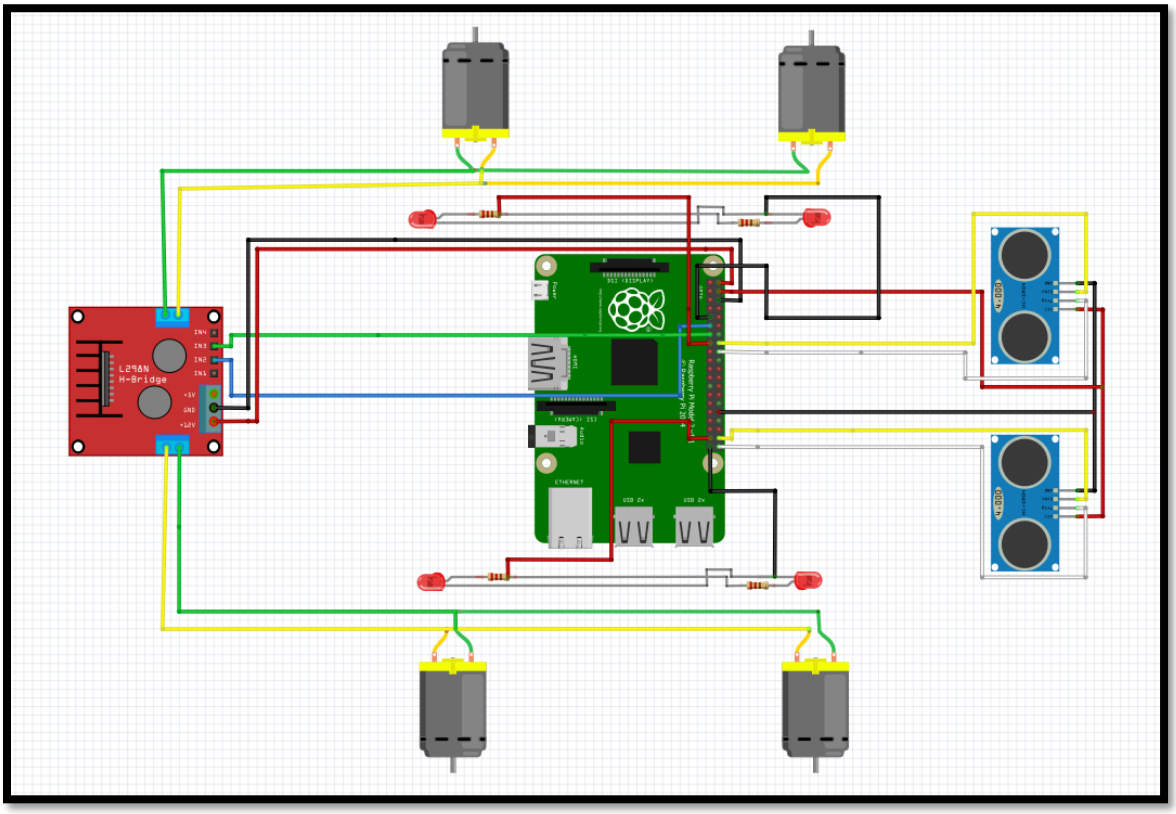
- Διάφορες αντιστάσεις

- Καλώδια Male to Female

- Καλώδια Female to Female



7.2 Σχηματική Αναπαράσταση Κυκλώματος



Εικόνα 7.1: Σχηματική αναπαράσταση κυκλώματος μέσω εφαρμογής Fritzing

Στην παραπάνω εικόνα παριστάνεται η σχηματική αναπαράσταση του κυκλώματος της παρούσας πτυχιακής εργασίας. Το μόνο που λείπει για την ολοκλήρωση του κυκλώματος είναι η κάμερα που χρησιμοποιήθηκε. Η κάμερα που χρησιμοποιήθηκε όπως αναφέραμε και πιο πάνω είναι η Logitech C625 και επιλέχτηκε για την καλύτερη δυνατή απόδοση. Η πτυχιακή εργασία είχε αρχικά υλοποιηθεί με την Raspberry Pi Camera Module Rev 1.3 αλλά δεν έδινε τα επιθυμητά αποτελέσματα.

7.3 Κώδικας Πτυχιακής Εργασίας

```
1.  import numpy as np
2.  import cv2
3.  from imutils.video import WebcamVideoStream
4.  import imutils
5.  import RPi.GPIO as GPIO
6.  import time
7.
8.  LEFT_LED = 22 # Αριθμός Pin LED αριστερής πλευράς
9.  RIGHT_LED = 26 # Αριθμός Pin LED δεξιάς πλευράς
10.
11.
12. def init(): # Δηλώσεις Pin
13.     GPIO.setwarnings(False)
14.     GPIO.setmode(GPIO.BCM)
15.     GPIO.setup(17,GPIO.OUT)
16.     GPIO.setup(27,GPIO.OUT)
17.
18.     GPIO.setup(LEFT_LED,GPIO.OUT)
19.     GPIO.setup(RIGHT_LED,GPIO.OUT)
20.     GPIO.output(LEFT_LED, False)
21.     GPIO.output(RIGHT_LED, False)
22.
23. init()
24.
25. pwm = GPIO.PWM(17, 100)  # Δήλωση τροχών
26. pwm2 = GPIO.PWM(27, 100)
27.
28.
```

```
29. def leftled(): # Προσωμοίωση Αριστερού Φλας
30.     GPIO.output(LEFT_LED, True)
31.     time.sleep(0.5)
32.     GPIO.output(LEFT_LED, False)
33.     time.sleep(0.5)
34.     GPIO.output(LEFT_LED, True)
35.     time.sleep(0.5)
36.     GPIO.output(LEFT_LED, False)
37.
38.
39. def righttled(): # Προσωμοίωση Δεξιού Φλας
40.     GPIO.output(RIGHT_LED, True)
41.     time.sleep(0.5)
42.     GPIO.output(RIGHT_LED, False)
43.     time.sleep(0.5)
44.     GPIO.output(RIGHT_LED, True)
45.     time.sleep(0.5)
46.     GPIO.output(RIGHT_LED, False)
47.
48. def onoff(): # Προσωμοίωση αλαρμ
49.     GPIO.output(LEFT_LED, True)
50.     GPIO.output(RIGHT_LED, True)
51.     time.sleep(1)
52.     GPIO.output(LEFT_LED, False)
53.     GPIO.output(RIGHT_LED, False)
54.
55. def alarm(): # Προσωμοίωση αλαρμ
56.     onoff()
57.     time.sleep(0.4)
58.     onoff()
59.     time.sleep(0.4)
```

```
60.
61.
62. def start(): # Εκκίνηση τροχών
63.     GPIO.output(17,GPIO.HIGH)
64.     GPIO.output(27,GPIO.HIGH)
65.
66. def stop(): # Σταμάτημα τροχών
67.     GPIO.output(17,GPIO.LOW)
68.     GPIO.output(27,GPIO.LOW)
69.
70. def left(): # Αριστερή στροφή
71.     GPIO.output(17,GPIO.HIGH)
72.     time.sleep(1.5)
73.     GPIO.output(17,GPIO.LOW)
74.
75. def epitopou(): # Επιτόπου
76.     GPIO.output(17,GPIO.HIGH)
77.     time.sleep(2.8)
78.     GPIO.output(17,GPIO.LOW)
79.
80. def right(): #Δεξιά Στροφή
81.     GPIO.output(27,GPIO.HIGH)
82.     time.sleep(1.5)
83.     GPIO.output(27,GPIO.LOW)
84.
85. def findcorners(img):
86.     img = cv2.imread(img) # Εικόνα προς είσοδο
87.     gray = cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)
88.     # Μετατροπή σε αποχρώσεις του Γκρι
89.
90.     corners = cv2.goodFeaturesToTrack(gray,15,0.05,5) # Δήλωση για γωνίες
```

```
91.  circles = cv2.HoughCircles(gray, cv2.HOUGH_GRADIENT,1.2, 100)
     # Δήλωση για κυκλικά σχήματα
92.  counta = 0 # Αρχικοποίηση μεταβλητών για γωνίες από το κέντρο και αριστερά
93.  countd = 0 # Αρχικοποίηση μεταβλητών για γωνίες από το κέντρο και δεξιά
94.
95.
96.  if corners is not None:  # Σιγουρευόμαστε ότι έστω κάποιες γωνίες βρέθηκαν
97.
98.      corners = np.int0(corners)
         # Μετατροπή των συντεταγμένων των γωνιών σε ακέραιους αριθμούς
99.
100.
101.         if circles is not None:
             # Σιγουρευόμαστε ότι έστω κάποια κυκλικά σχήματα βρέθηκαν
102.
103.             circles = np.round(circles[0, :]).astype("int")
104.             # Μετατροπή των (x, y) συντεταγμένων και της ακτίνας του
                 κύκλου σε ακέραιους αριθμούς
105.
106.
107.             for (x, y, r) in circles:
108.             # Λουπάρουμε μέσα στις (x, y) συντεταγμένες και την ακτίνα
                 των κυκλικών σχημάτων που θα βρούμε
109.
110.                 #cv2.circle(img, (x, y), r, (0, 255, 0), 4)
111.         # Σχεδίαση πράσινου κύκλου γύρω από το κυκλικό σχήμα που εντοπίστηκε
112.
113.             for i in corners: # Λουπάρουμε μέσα στις γωνίες
114.
115.                 #cv2.circle(img, (x, y), 4, (0, 128, 255), -1)
116.             # Σχεδίαση κεντρικού σημείου το κύκλου με πορτοκαλί χρώμα
```

```
117.
118.                     x1,y1 = i.ravel()
119.                     # Δήλωση συντεταγμένων για τις γωνίες
120.
121.                     if x1>x and x1<x+50 and y1>y-50 and y1<y+50:
122.               # Έλεγχος για το άν βρίσκονται οι γωνίες από το κέντρο και δεξιά
123.
124.                       #cv2.circle(img, (x1, y1), 3, (0, 0, 255), -1)
125.                       # Σχεδίαση σημείου γωνίας με κόκκινο χρώμα
126.                         countd=countd+1 # Αύξηση μεταβλητής κατά ένα
127.
128.                     elif x1<x and x1>x-50 and y1>y-50 and y1<y+50:
129.             # Έλεγχος για το άν βρίσκονται οι γωνίες από το κέντρο και αριστερά
130.
131.                       #cv2.circle(img, (x1, y1), 3, (0, 0, 0), -1)
132.                       # Σχεδίαση σημείου γωνίας με μαύρο χρώμα
133.                       counta=counta+1 # Αύξηση μεταβλητής κατά ένα
134.
135.
136.           print "Exoume %d gonies apo to kentro kai aristera" % counta
137.           print "Exoume %d gonies apo to kentro kai deksia" % countd
138.
139.           if (counta+countd >= 2): # Έλεγχος αν οι γωνίες είναι >=2
140.              if (counta > countd):
141.
142.                 print "LEFT ARROW FOUND"
143.                 leftled() # Άναψε αριστερό Φλας
144.                 pwm.stop() # Σταμάτημα τροχών
145.                 pwm2.stop()
146.                 time.sleep(0.5)
147.                 left()  # Αριστερή στροφή
```

```
148.                 pwm.start(70)
149.                 pwm2.start(70) # Εκκίνηση τροχών
150.
151.             elif (counta < countd):
152.
153.                 print "RIGHT ARROW FOUND"
154.                 righttled() # Άναψε δεξί Φλας
155.                 pwm.stop() #Σταμάτημα τροχών
156.                 pwm2.stop()
157.                 time.sleep(0.5)
158.                 right() # Δεξιά Στροφή
159.                 pwm.start(70) # Εκκίνηση τροχών
160.                 pwm2.start(70)
161.
162.
163.     pwm.start(70)  # Εκκίνιση τροχών
164.     pwm2.start(70)
165.     stop_cascade = cv2.CascadeClassifier('xml/stop_sign.xml')
        # Δήλωση ταξινομητή
166.     vs = WebcamVideoStream(src=0).start() # Εκκίνηση Video Streaming
167.
168.
169.     while(True):
170.         # Σύλληψη Frames
171.         frame = vs.read()
172.         frame = imutils.resize(frame, width=400)
173.
174.         # Μετατροπή κάθε Frame σε Grayscale Image
175.         gray = cv2.cvtColor(frame,cv2.COLOR_BGR2GRAY)
176.
177.         # Δήλωση - Αρχικοποίηση κύκλων
```

```
178.        circles = cv2.HoughCircles(gray, cv2.HOUGH_GRADIENT, 1.2, 350)
179.        # Κλήση συνάρτησης detectMultiScale
180.        stops = stop_cascade.detectMultiScale(gray, 1.1)
181.
182.
183.        if circles is not None:
184.        # Σιγουρευόμαστε ότι έστω κάποια κυκλικά σχήματα βρέθηκαν
185.
186.            circles = np.round(circles[0, :]).astype("int")
187.       # Μετατροπή των (x, y) συντεταγμένων και της ακτίνας του
          κύκλου σε ακέραιους αριθμούς
188.
189.            # Λουπάρουμε μέσα στις (x, y) συντεταγμένες και την ακτίνα
          των κυκλικών σχημάτων που θα βρούμε
190.            for (x, y, r) in circles:
191.                # Σχεδίαση πράσινου κύκλου γύρω από το κυκλικό σχήμα
                που εντοπίστηκε
192.                cv2.circle(gray, (x, y), r, (0, 255, 0), 4)
193.
194.                if format(len(circles)) > 0: # Αν τα κυκλικά που
                                εντοπίστηκαν είναι έστω 1
195.
196.                    #print "X =",x   # Συντεταγμένες x
197.                    #print "Y =",y  # Συντεταγμένες y
198.                    #print "R =",r  # Ακτίνα Κύκλου
199.                    sign = True
200.
201.
202.
203.                    if r > 35 and y > 50 and y < 140:
204.                    # Απόσταση από το σήμα και συγκεκριμένη θέση
```

```
205.
206.                     pwm.stop()
207.                     pwm2.stop()
208.                     cv2.imwrite('image1.jpg',frame)
                    # Συλληψη Φωτογραφίας
209.                     findcorners('image1.jpg')
                    # Εντοπισμός Κατεύθυνσης Βέλους
210.                     sign = False
211.
212.
213.                 if x < 180 and sign == True:
214.                 # Αν το σήμα είναι πιο δεξιά απο το όχημα να
                πάει προς εκείνη την κατεύθυνση
215.
216.                     pwm2.start(10)
217.                     pwm.start(60)
218.
219.                 elif x > 220  and sign == True:
220.                 # Αν το σήμα είναι πιο αριστερά απο το όχημα να
                πάει προς εκείνη την κατεύθυνση
221.
222.                     pwm.start(10)
223.                     pwm2.start(60)
224.
225.
226.                 elif x > 180 and x < 220  and sign == True:
227.                 # Αν το σήμα είναι στην ίδια ευθεία με το όχημα
                να συνεχίσει όπως είναι
228.
229.                     pwm.start(70)
230.                     pwm2.start(70)
```

```
231.
232.         for (x,y,w,h) in stops:
233.             #text_color = (255,0,0)
234.             #cv2.putText(frame, "STOP",(x+50,y-10),
    cv2.FONT_HERSHEY_PLAIN, 1.0, text_color, thickness=1)
235.             #cv2.rectangle(frame,(x,y),(x+w,y+h),(255,0,0),2)
236.
237.             #print "X = ", x        # Συντεταγμένες εντοπισμένου STOP
238.             #print "Y = ", y
239.             #print "W = ", w
240.             #print "H = ", h
241.
242.         if len(stops) == 1:  # Αν βρέθηκε σήμα STOP
243.
244.             if w >= 105:   # Επιθυμητή απόσταση απο σηματοδέκτη STOP
245.                 print("STOP SIGN FOUND")
246.                 pwm.stop()  # Σταμάτημα τροχών
247.                 pwm2.stop()
248.                 alarm()  # Προσωμοίωση Αλαρμ
249.                 time.sleep(1)
250.                 epitopou() # Επιτόπου για να γυρίσει το όχημα πίσω σε εμάς
251.                 pwm.start(70) # Εκκίνηση τροχών
252.                 pwm2.start(70)
253.
254.             elif x < 100:
255.             # Αν το STOP είναι πιο δεξιά απο το όχημα να πάει
                προς εκείνη την κατεύθυνση
256.
257.                 pwm2.start(10)
258.                 pwm.start(60)
259.
```

```
260.            elif x > 200:
261.            # Αν το STOP είναι πιο αριστερά απο το όχημα
                να πάει προς εκείνη την κατεύθυνση
262.
263.                pwm.start(10)
264.                pwm2.start(60)
265.
266.
267.            elif x > 100 and x < 200:
268.            # Αν το STOP είναι στην ίδια ευθεία με το όχημα να
                συνεχίσει όπως είναι
269.
270.                pwm.start(70)
271.                pwm2.start(70)
272.
273.
274.        # Frame Εξόδου
275.        #cv2.imshow('frame',gray)
276.        if cv2.waitKey(1) & 0xFF == ord('q'):  # Πιεση πλήκτρου q έξοδο
277.            break
278.
279.    cap.release() # Όταν τερματίσει το πρόγραμμα, να τερματίσει την σύλληψη
280.    cv2.destroyAllWindows() # Κλείσιμο όλων των παραθύρων
```

ΚΕΦΑΛΑΙΟ 8

ΣΥΜΠΕΡΑΣΜΑΤΑ ΚΑΙ ΜΕΛΛΟΝΤΙΚΗ ΕΞΕΛΙΞΗ

8.1 Συμπεράσματα

Στην παρούσα Πτυχιακή Εργασία προτάθηκε ένα αυτόνομο όχημα κινούμενο με την αναγνώριση ορισμένων σημάτων κυκλοφορίας. Η υλοποίηση του οχήματος έγινε με τη χρήση της πλατφόρμας Raspberry Pi 2 και τη γλώσσα προγραμματισμού Python.

Η κατασκευή του οχήματος ήταν μια αρκετά διασκεδαστική διαδικασία· έχοντας τις βασικές γνώσεις από διάφορες κατασκευές που ασχολήθηκα σε προηγούμενα εξάμηνα,δεν με δυσκόλευσε ιδιαίτερα.Η υλοποίηση όμως του προγραμματιστικού κομματιού απαίτησε να αφιερώσω αρκετό χρόνο για την εκμάθηση της γλώσσας προγραμματισμού Python η οποία είναι μια γλώσσα που δεν διδάχθηκα ποτέ σε ολόκληρο τον κύκλο μαθημάτων μου.

Κατά την εκμάθηση της γλώσσας προγραμματισμοού Python είχα αρχικά αρκετές δυσκολίες τις οποίες έπρεπε να ξεπεράσω για να φθάσω σε ένα ικανοποιητικό επίπεδο και να καταφέρω να προγραμματίσω το Raspberry Pi να αναγνωρίζει τα σήματα κυκλοφορίας.

Αφιερώνοντας ώρες σε δωρεάν διαδικτυακά μαθήματα εκμάθησης της γλώσσας προγραμματισμού Python και της βιβλιοθήκης Open CV, κατάφερα να φέρω εις πέρας την εργασία που μου ανατέθηκε αλλά και να αποκτήσω πολλά εφόδια τα οποία αδιαμφισβήτητα θα αξιοποιήσω και στο μέλλον.

Η συγκεκριμένη εργασία, πέρα από Πτυχιακή Εργασία, για εμένα, ήταν και ένα μεγάλο μάθημα καθώς δημιούργησα κάτι με το οποίο δεν είχα ασχοληθεί ξανά στο παρελθόν.

8.2 Μελλοντική Εξέλιξη

Ως μελλοντικές πιθανές εξελίξεις του παρόντος συστήματος που αναπτύχθηκε για την συγκεκριμένη Πτυχιακή Εργασία θα μπορούσαμε να σημειώσουμε:

1) Την δημιουργία ενός Manual Steering Mode, το οποίο θα δίνει την δυνατότητα στον χρήστη να χρησιμοποιήσει το αυτοκίνητο σαν ένα απλό τηλεκατευθυνόμενο όχημα, θέτωντας το όμως αδύνατο να συγκρουστεί με κάποιο εμπόδιο χρησιμοποιώντας τους δύο αισθητήρες υπερήχων (HC-SR04) που είναι ήδη τοποθετημένοι.

   Για την υλοποίηση του παραπάνω Mode θα χρειαστεί ένας IR Sensor (Infrared Radiation), και ένας IR Blaster(Οποιοδήποτε τηλεχειριστήριο τηλεόρασης,air-condition του σπιτιού).

2) Την δημιουργία ενώς Web Server μέσω του Raspberry Pi, από τον οποίο θα μας δίνεται η δυνατότητα απομακρυσμένης χρήσης των τροχών και της κάμερας που βρίσκονται στο όχημα.

   Η συκγκριμένη εξέλιξη απαιτεί ένα Wi-Fi Adapter για την υλοποίηση του καθώς η συκγκεκριμένη έκδοση του Raspberry Pi που χρησιμοποίησα για την παρούσα εργασία δεν έρχεται με κάποιο προεγκατεστημένο Wi-Fi Module.

ΒΙΒΛΙΟΓΡΑΦΙΑ

[1]: Εγκατάσταση OpenCV σε Raspberry Pi
http://www.pyimagesearch.com/2015/02/23/install-opencv-and-python-on-your-raspberry-pi-2-and-b/

[2]: Εκπαίδευση Ταξινομητή (Haar Classifier Training)

http://coding-robin.de/2013/07/22/train-your-own-opencv-haar-classifier.html

[3]: Ηλεκτροκινητήρες

http://physiclessons.blogspot.com/2013/03/blog-post_3505.html

http://www.robolab.tuc.gr/ASSETS/PAPERS_PDF/ROBOTICS/LAB/4_STEPPER&SERVOS_LAB.pdf

https://en.wikipedia.org/wiki/Electric_motor

[4]: Αισθητήρας Υπερήχων (HC-SR04)

https://apothesis.lib.teicrete.gr/bitstream/handle/11713/7880/ChionidisPavlos_TzavolakisEmmanouil2016.pdf?sequence=1

[5]: OpenCV Θεωρία

http://graphics.di.uoa.gr/people/docs/Perakis_MSc_Thesis.pdf

http://nemertes.lis.upatras.gr/jspui/bitstream/10889/4394/1/Protonotarios.Ioannis.Thesis.pdf

https://en.wikipedia.org/wiki/OpenCV

https://www.pyimagesearch.com/

[6]: Μηχανική Όραση - Θεωρία

https://apothesis.lib.teicrete.gr/bitstream/handle/11713/7656/KritsotakisEvangelos2016.pdf

[7]:Το Βίντεο ως Σήμα - Θεωρία

http://rtsimage.di.uoa.gr/publications/rtdvp.pdf

[8]:Πρωτόκολλο FTP

https://el.wikipedia.org/wiki/File_Transfer_Protocol

[9]:Πρωτόκολλο SSH

https://el.wikipedia.org/wiki/SSH