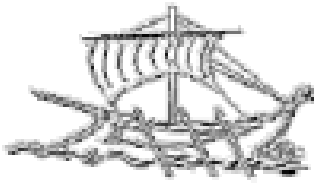


ΑΕΙ ΠΕΙΡΑΙΑ Τ.Τ.
ΣΧΟΛΗ ΤΕΧΝΟΛΟΓΙΚΩΝ ΕΦΑΡΜΟΓΩΝ
ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΗΛΕΚΤΡΟΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΙΚΩΝ ΣΥΣΤΗΜΑΤΩΝ Τ.Ε.

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Πλοήγηση Αυτόνομου Οχήματος

Γεώργιος Μουρτζούνης

Ευάγγελος Καβαλιέρος

Εισηγητής: Ιωάννης Ν. Έλληνας, Καθηγητής

ΑΘΗΝΑ
ΝΟΕΜΒΡΙΟΣ 2017

(Κενό φύλλο)

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Πλοήγηση Αυτόνομου Οχήματος

Γεώργιος Μουρτζούνης
Α.Μ.: 43884

Ευάγγελος Καβαλιέρος
Α.Μ.: 43837

Εισηγητής:

Ιωάννης Ν. Έλληνας, Καθηγητής

Εξεταστική Επιτροπή:

Ημερομηνία εξέτασης --/--/2017

(Κενό φύλλο)

ΕΥΧΑΡΙΣΤΙΕΣ

Θα θέλαμε να εκφράσουμε τις θερμές μας ευχαριστίες στον επιβλέποντα καθηγητή Ιωάννη Ν. Έλληνα, του τμήματος Μηχανικών Ηλεκτρονικών Υπολογιστικών Συστημάτων, για την ακλόνητη εμπιστοσύνη του και την αμέριστη και συνεχή καθοδήγησή του τόσο κατά τη διάρκεια εκπόνησης της πτυχιακής εργασίας όσο και κατά το διάστημα συγγραφής της.

Επίσης θα θέλαμε να ευχαριστήσουμε αφενός τις οικογένειές μας, που υποστήριξαν ποικιλοτρόπως τις σπουδές μας, κι αφετέρου τους φίλους μας για τη ψυχολογική υποστήριξη και παρακίνηση που παρείχαν καθ' όλη τη διάρκεια της φοίτησής μας.

(Κενό φύλλο)

ΠΕΡΙΛΗΨΗ

Η παρούσα πτυχιακή εργασία πραγματεύεται την πλοήγηση ενός αυτόνομου οχήματος και στόχος αυτής είναι η περιήγηση κι η παραμονή του εντός της διαμορφωμένης πίστας. Τα παραπάνω επιτυγχάνονται μέσω του υπολογιστή Raspberry Pi 3 Model B, μιας Web Camera και ενός οχήματος με 2 DC κινητήρες. Η κάμερα αναγνωρίζει τη λωρίδα και το αμαξίδιο κινείται ευθεία, ενώ παράλληλα προσαρμόζει τις μπροστινές ρόδες ούτως ώστε να παραμείνει εντός αυτής. Η ανάπτυξη λογισμικού πραγματοποιείται μέσω της προγραμματιστικής γλώσσας Python και κατά κύριο λόγο μέσω της βιβλιοθήκης OpenCV.

ABSTRACT

This diploma thesis deals with the navigation of an autonomous vehicle. The aim of this is to navigate and stay within the configured track and identify a traffic signal. The above are achieved through the Raspberry Pi 3 Model B computer, a Web Camera and a vehicle with two DC motors. The camera identifies the lane and the car moves forward, while adjusting the front wheels so that it stays within it. The project is developed through the Python programming language and primarily the OpenCV library.

ΕΠΙΣΤΗΜΟΝΙΚΗ ΠΕΡΙΟΧΗ: Επεξεργασία Εικόνας σε Πραγματικό Χρόνο
ΛΕΞΕΙΣ ΚΛΕΙΔΙΑ: Επεξεργασία Εικόνας, Computer Vision, Πλοήγηση, Python, Raspberry

ΠΕΡΙΕΧΟΜΕΝΑ

ΚΑΤΑΛΟΓΟΣ ΕΙΚΟΝΩΝ

ΚΑΤΑΛΟΓΟΣ ΠΙΝΑΚΩΝ

ΣΥΝΤΟΜΟΓΡΑΦΙΕΣ

A	Ampere
CGI	Computer Generated Imagery
CLI	Command-Line Interface
DC	Direct Current
FTP	File Transfer Protocol
GB	Gigabyte
GPIO	General Purpose Input Output
GPU	Graphics Processing Unit
GUI	Graphical User Interface
HDMI	High Definition Multimedia Interface
HTML	Hypertext Markup Language
I^2C	Inter Integrated Circuit
IoT	Internet of Things
LAN	Local Area Network
MB	Megabyte
PWM	Pulse Width Modulation
RAM	Random Access Memory
RPC	Remote Procedure Call
W	Watt
WAV	Waveform Audio File Format
Wi-Fi	Wireless Fidelity
XML – RPC	Extensible Markup Language

ΕΙΣΑΓΩΓΗ

Σε αυτό το κεφάλαιο αναλύεται το αντικείμενο της πτυχιακής εργασίας, παρουσιάζεται ο βασικός σκοπός της, καθώς επίσης και μια σύντομη ιστορική αναδρομή.

1.1 Περιγραφή του αντικειμένου της πτυχιακής εργασίας

Η παρούσα πτυχιακή εργασία εστιάζει στην επεξεργασία εικόνας σε συνδυασμό με τον έλεγχο DC κινητήρων, οι οποίοι κι απαρτίζουν το αυτόνομο όχημα. Συγκεκριμένα χρησιμοποιείται το Raspberry Pi 3 Model B, ένας αρκετά ανταγωνιστικός για το μέγεθός του υπολογιστής. Σε αυτό συνδέεται η Web Camera μέσω τροφοδοτούμενης θύρας USB, οι δυο DC κινητήρες με τον ελεγκτή L298n κι αυτός με τις επιθυμητές ακίδες GPIO. Ο βασικός σκοπός της πτυχιακή εργασίας είναι η αυτόνομη κίνηση, η ομαλή πλοήγηση και η παραμονή του αμαξιδίου εντός της προσαρμοσμένης πίστας. Αυτό επιτυγχάνεται από την ανάλυση των frames σε πραγματικό χρόνο μέσω της κάμερας, την αναγνώριση της λωρίδας, στην οποία βρίσκεται το όχημα, και τον έλεγχο των κινητήρων , οι οποίοι είναι υπεύθυνοι για την κίνηση των τροχών.

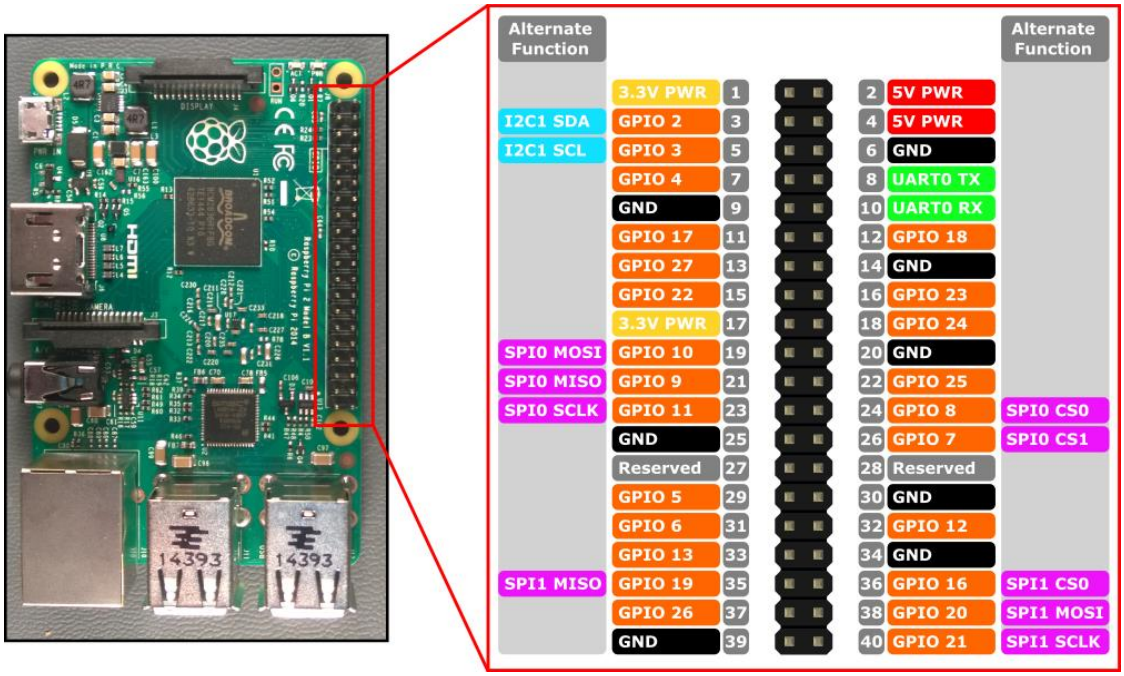
1.2 Περιγραφή και Χαρακτηριστικά Raspberry Pi

Πρόκειται για ένα υπολογιστή χαμηλού κόστους και μικρού μεγέθους, ο οποίος είναι δομημένος σε ανοιχτού τύπου λογισμικό και βασίζεται στο Linux. Η ζήτηση του Raspberry Pi αυξήθηκε δραματικά, καθώς έχει πολλαπλές εφαρμογές που ικανοποιούν τις απαιτήσεις του κοινωνικού συνόλου τόσο σε προσωπικό, όσο και σε επαγγελματικό επίπεδο. Μπορεί να χρησιμοποιηθεί ως κονσόλα για Arcade παιχνίδια, Media Centre για την τηλεόραση, προσωπικό Cloud Server, για εκμάθηση Προγραμματιστικών Γλωσσών, bitcoin mining, ως συναγερμός κ.α. Τα τεχνικά χαρακτηριστικά του Raspberry Pi 3 model B, με το οποίο υλοποιήθηκε η πτυχιακή εργασία, αναφέρονται παρακάτω:

Πίνακας 1.1

Χαρακτηριστικά	
Επεξεργαστής	○ Broadcom BCM2387 (SoC) ○ 1.2 GHz Quad-Core ARM Cortex-A53 ○ 802.11 b/g/n Wireless LAN ○ Bluetooth 4.1 (Κλασσικό και Χαμηλής κατανάλωσης)
GPU	○ Dual Core VideoCore IV Multimedia Co-Processor
Μνήμη	○ 1 GB RAM LPDDR2
Λειτουργικό Σύστημα	○ Εκδόσεις Linux ○ Windows 10 IoT ○ (κάνει boot από micro SD κάρτα)
Διαστάσεις	○ 85 x 56 x 17mm
Συνδέσεις	
Ethernet	○ 10/100 Mbps
Έξοδος Βίντεο	○ HDMI (Composite)
Έξοδος Ήχου	○ 3.5mm jack ○ HDMI ○ USB 4 x USB 2.0
Ακίδες GPIO	○ 40 Ακίδες
Θύρα Κάμερας	○ 15 Ακίδες Camera Serial Interface (CSI-2)
Σύνδεση Οθόνης	○ Display Serial Interface (DSI)
Τροφοδοσία	○ Micro USB 5V , 2.5A

1.3 Ακίδες GPIO



Εικόνα 1.1: Raspberry Pi 3 Pinout

Ένα από τα πιο σημαντικά χαρακτηριστικά του Raspberry Pi είναι οι σειρές ακίδων GPIO. Αποτελούνται από 40 pins εκ των οποίων τα 26 είναι γενικής χρήσης εισόδου/εξόδου, ενώ τα υπόλοιπα έχουν την ιδιότητα τροφοδοσίας, γείωσης ή ID EEPROM. Μπορούν να τραβήξουν μέχρι 50mA ρεύματος με ασφάλεια, ενώ κάθε ακίδα ξεχωριστά μπορεί μόνο 16mA.

Πίνακας 1.2

Ακίδα	Ιδιότητα
1, 17	Τροφοδοσία 3.3V (έως 50 mA)
2, 4	Τροφοδοσία 5V
6, 9, 14, 20, 25, 30, 34, 39	Γείωση
27, 28	I^2C επικοινωνία με EEPROM
3, 5, 7, 8, 10, 11, 12, 13, 15, 16, 18, 19, 21, 22, 23, 24, 26, 29, 31, 32, 33, 35, 36, 37, 38, 40	Γενικής Χρήσης (General Purpose Input/Output)

1.4 Υποστήριξη Λειτουργικών Συστημάτων

Αν ληφθεί υπόψη πως το Λειτουργικό Σύστημα του Raspberry Pi βρίσκεται μέσα σε μια Micro SD κάρτα , από την οποία γίνεται το boot, αντιλαμβάνεται κανείς τις δυνατότητές του. Υπάρχουν δεκάδες Λειτουργικά Συστήματα από τα οποία ο χρήστης μπορεί να επιλέξει αναλόγως των απαιτήσεών του. Κάποια από αυτά είναι:

Πίνακας 1.3

Λογισμικά	
○ Raspbian	○ Arch Linux ARM
○ Ubuntu MATE	○ Gentoo Linux
○ Snappy Ubuntu	○ FreeBSD
○ Pidora	○ Kali Linux
○ Linutop	○ RISC OS Pi
○ SARPi	○ Noobs

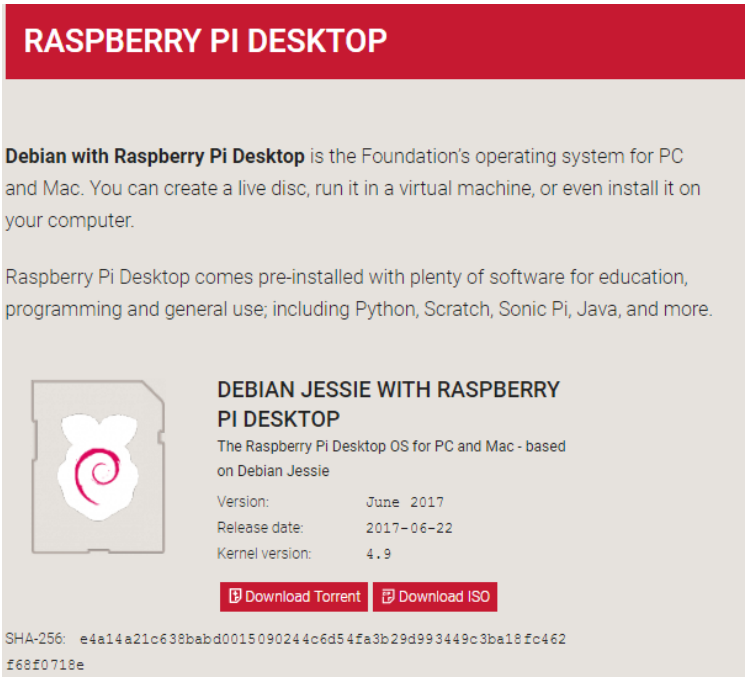
Στην προκειμένη πτυχιακή εργασία έχει επιλεχθεί το Λειτουργικό Σύστημα Raspbian και πιο συγκεκριμένα η έκδοση Jessie. Πρόκειται για ένα δωρεάν λειτουργικό σύστημα, που είναι βασισμένο στο Debian, και είναι βελτιστοποιημένο για τις hardware λειτουργίες του Raspberry Pi. Οι παροχές που διαθέτει ξεπερνά κάθε προηγούμενο, καθώς συμπεριλαμβάνονται εργαλεία ανάπτυξης λογισμικού, εκμάθησης προγραμματισμού κ.α. Πλέον η προκαθορισμένη συμπεριφορά του είναι να κάνει boot κατευθείαν στο γραφικό περιβάλλον του Desktop , εν αντιθέσει με το τερματικό Linux σε προηγούμενες εκδόσεις, καθιστώντας το ένα πλήρως λειτουργικό κι ανταγωνιστικό υπολογιστή. Ειδικότερα, περιέχει επεξεργαστή κειμένου, υπολογιστικά φύλλα, παρουσίαση γραφικών, σχέδια διανυσμάτων, εργαλεία προγραμματισμού για γλώσσες όπως Java και Python και νέα παράθυρα διαμόρφωσης των ρυθμίσεων του Raspberry. Το νέο αυτό περιβάλλον είναι φιλικότερο προς το χρήστη και του παρέχει τις δυνατότητες ενεργοποίησης/απενεργοποίησης διεπαφών(interfaces), αλλαγή ζώνης ώρας και

γλώσσας πληκτρολογίου. Τέλος παρέχονται περισσότεροι έλεγχοι στις επιλογές εκκίνησης(boot) από ότι στο παρελθόν, δίνοντας την επιλογή στο χρήστη να συνδεθεί αυτόματα ως χρήστης «pi» κατά τη διάρκεια του boot, τόσο στο CLI όσο και στον υπολογιστή.

1.5 Εγκατάσταση Λειτουργικού Συστήματος

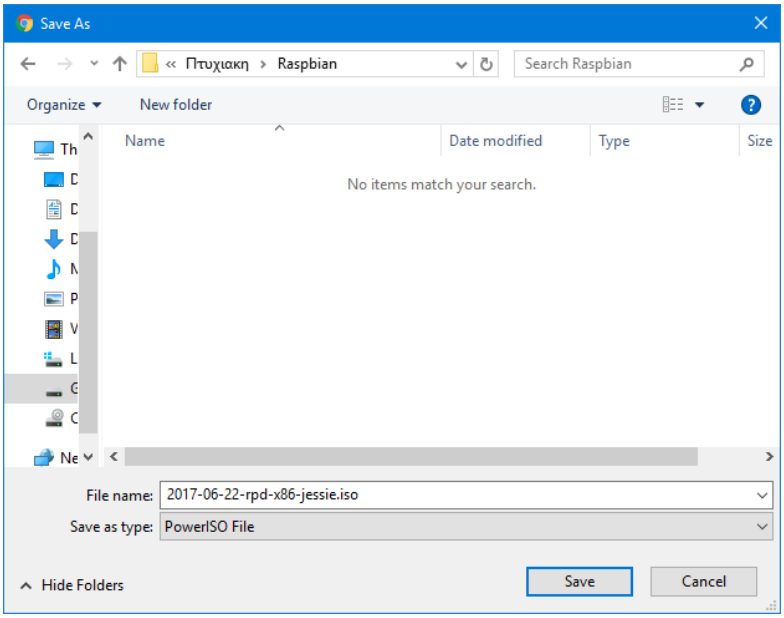
Όπως έχει προαναφερθεί, απαιτείται μια micro SD κάρτα, στην οποία θα εμπεριέχεται η εικόνα του Λειτουργικού Συστήματος. Έπειτα ο χρήστης εκτελεί τα παρακάτω βήματα:

1. Πλοηγείται στην ιστοσελίδα, όπου βρίσκονται τα αρχεία λήψης Raspbian, και κατεβάζει την εικόνα είτε μέσω Torrent είτε απευθείας.



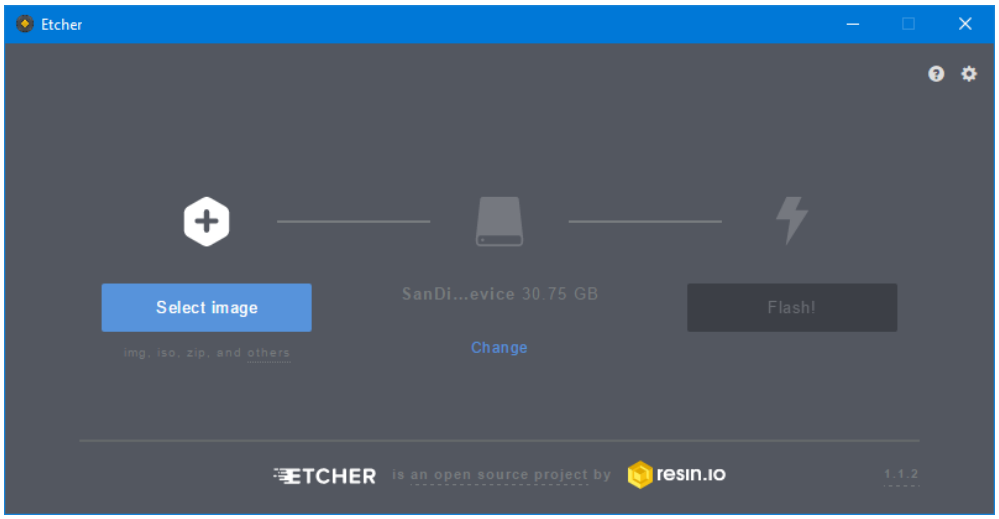
Εικόνα 1.2: Λήψη Αρχείου Raspbian Jessie

2. Αποθηκεύει την εικόνα στο επιθυμητό μονοπάτι (path) .



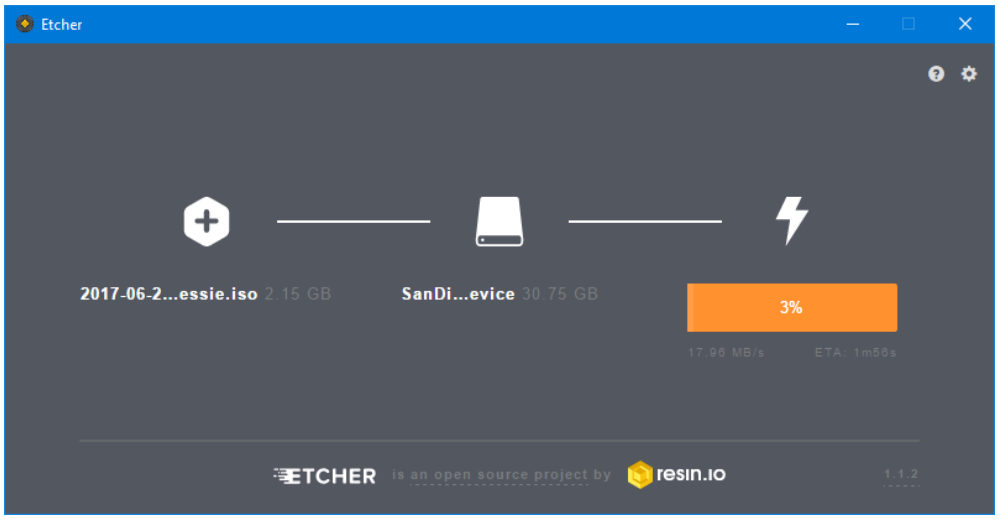
Εικόνα 1.3: Αποθήκευση Εικόνας

3. Κατεβάζει το εργαλείο γραφής Etcher και συνδέει την SD κάρτα , μέσω ενός card reader, στον υπολογιστή.
4. Εφόσον εγκαταστήσει το πρόγραμμα, το ανοίγει και επιλέγει την εικόνα του λειτουργικού συστήματος.



Εικόνα 1.4: Επιλογή Εικόνας

5. Πατάει «Flash» και περιμένει να ολοκληρωθεί η διαδικασία εγγραφής στην SD κάρτα.



Εικόνα 1.5: Flash Raspbian Jessie

6. Μόλις ολοκληρωθούν τα παραπάνω βήματα, αφαιρεί την SD κάρτα από το card reader και τη τοποθετεί στο Raspberry Pi.

1.6 Ιστορική Αναδρομή

Το Raspberry Pi εφευρέθηκε το Φεβρουάριο του 2012 από το Raspberry Pi Foundation και αρχικά προοριζόταν για να παρακινήσει τους μαθητές προς τα υπολογιστικά συστήματα. Εκείνη την περίοδο κυκλοφορούσαν οι πρώτες δύο συσκευές, το Model A και το Model B. Σύντομα σημείωσε μεγάλη απήχηση και σε τομείς εκτός σχολείων, όπως στη ρομποτική, λόγω των πολλών εφαρμογών του αλλά και του χαμηλού κόστους του. Έτσι το 2014 κυκλοφόρησαν τα Model A+ και Model B+ , τα οποία είχαν περισσότερες ακίδες GPIO με λιγότερη κατανάλωση ισχύος. Ενώ το 2015, ανακοινώθηκε η άφιξη του Raspberry Pi 2 αφενός με μεγαλύτερη επεξεργαστική ισχύ κι αφετέρου με διπλάσια RAM. Παρακάτω παρουσιάζεται ένας πίνακας με τα χαρακτηριστικά των προηγούμενων μοντέλων:

Πίνακας 1.4

RP1	Model A	Model A+	Model B	Model B+
Επεξεργαστής	Μονοπύρηνος @ 700MHz	Μονοπύρηνος @ 700MHz	Μονοπύρηνος @ 700MHz	Μονοπύρηνος @ 700MHz
Μνήμη	256mb	256mb	512mb	512mb
Θύρες USB	1	1	2	4
Έξοδος Βίντεο	Ανάλυση: 640x350-1920x1200	Ανάλυση: 640x350-1920x1200	Ανάλυση: 640x350-1920x1200	Ανάλυση: 640x350-1920x1200
GPIO	8	17	8	17
Διαστάσεις	85.6mm x 56.5mm	65mm x 65.5mm	85.6mm x 56.5mm	85.6mm x 56.5mm

HEADLESS PI

Στο κεφάλαιο αυτό θα δοθεί ο ορισμός του Headless Pi , ποιος είναι ο απώτερος σκοπός του και ποια τα εργαλεία που θα χρησιμοποιηθούν για την επίτευξη αυτού.

2.1 Ορισμός και σκοπός του Headless Pi

Η βασική ιδέα, όπως συνιστά και το όνομα «Headless», είναι το Raspberry Pi να μην έχει «κεφάλι», δηλαδή να μην είναι συνδεδεμένο σε κάποια οθόνη. Ως επί το πλείστον ο σκοπός του είναι να αποκατασταθεί πλήρως από συσκευές όπως πληκτρολόγιο, ποντίκι κι οθόνη και να λειτουργεί ασύρματα μέσω κάποιου υπολογιστή , είτε με σύνδεση τοπικού δικτύου είτε μέσω server.

2.2 Εγκατάσταση VNC και σύνδεση με το Raspberry Pi

Το VNC Connect αποτελείται από το VNC Server και το VNC Viewer. Το VNC Server επιτρέπει τη σύνδεση από τον υπολογιστή ή το κινητό τηλέφωνο στο Raspberry Pi ελέγχοντάς το σε πραγματικό χρόνο. Ενώ το VNC Viewer επιτρέπει τη σύνδεση και τον έλεγχο ενός υπολογιστή ή ενός άλλου Raspberry. Παρακάτω παρουσιάζονται τα βήματα που χρειάζεται να ακολουθήσει ο χρήστης ούτως ώστε να υπάρχει επικοινωνία μεταξύ υπολογιστή και Raspberry.

1. Ο χρήστης εγκαθιστά το λογισμικό του VNC Viewer από τον σύνδεσμο στον υπολογιστή, που θέλει να ελέγχει το Raspberry Pi.

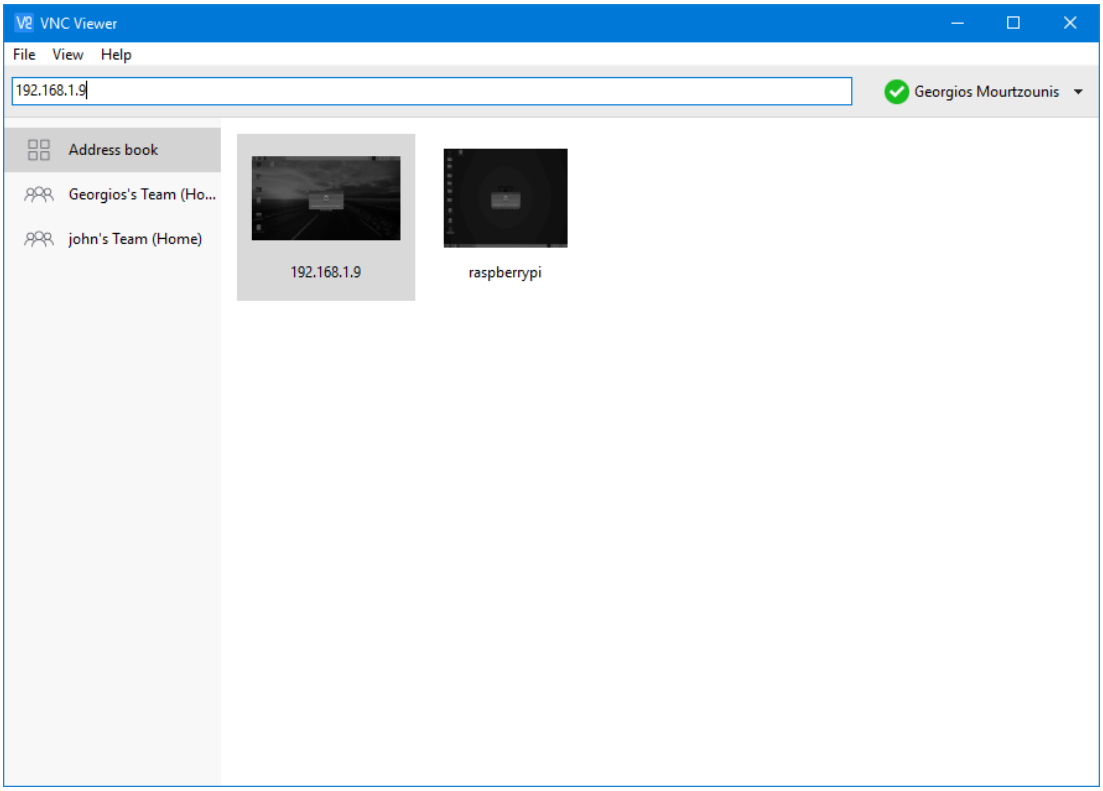
2. Στο Raspberry Pi εκτελεί τις ακόλουθες εντολές στο τερματικό του για να βεβαιωθεί πως έχει τη νεότερη έκδοση του λογισμικού:

```
$sudo apt-get update
$sudo apt-get install realvnc-vnc-server
$sudo apt-get install realvnc-vnc-viewer
```

3. Πλοηγείται στο Menu > Preferences > Raspberry Pi Configuration > Interfaces κι επιλέγει το VNC να είναι Enabled.

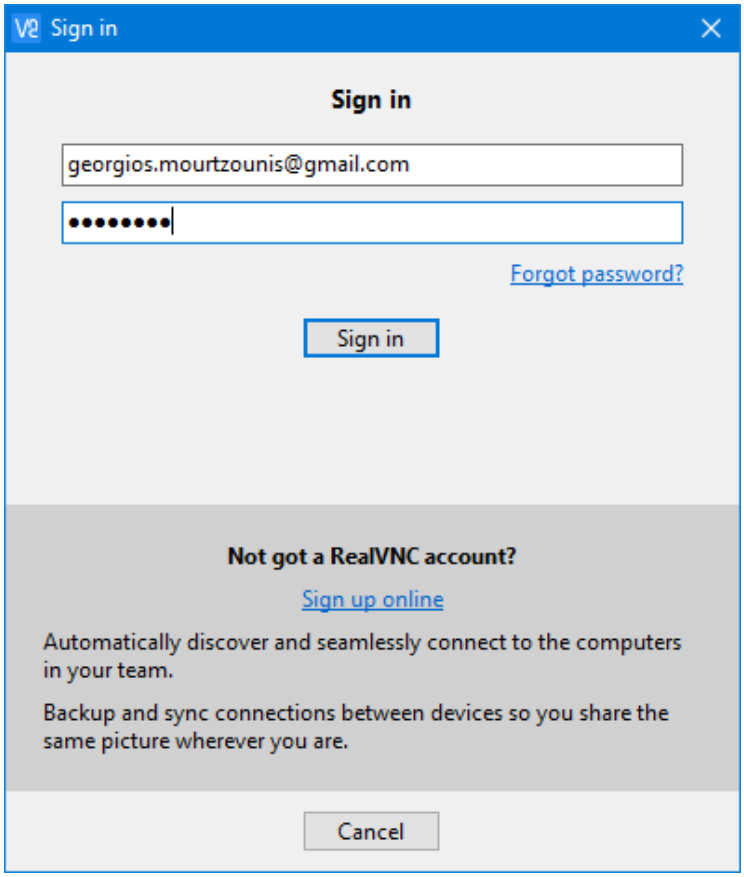
   Όπως προαναφέρθηκε η σύνδεση μεταξύ Raspberry και υπολογιστή, μπορεί να επιτευχθεί με δύο τρόπους. Πρώτα θα αναλυθεί αυτός του τοπικού δικτύου κι ύστερα αυτός του server.

4. (Τρόπος Τοπικού Δικτύου). Ο χρήστης ανοίγει το τερματικό από το Raspberry Pi και εκτελεί την εντολή «hostname -I» για να μάθει την IP διεύθυνσή του, δεδομένου ότι βρίσκονται στο ίδιο δίκτυο. Στη συνέχεια ανοίγει το VNC Viewer, που εγκατέστησε προηγουμένως στον υπολογιστή του, και στη μπάρα αναζήτησης συμπληρώνει την IP διεύθυνση του Raspberry.



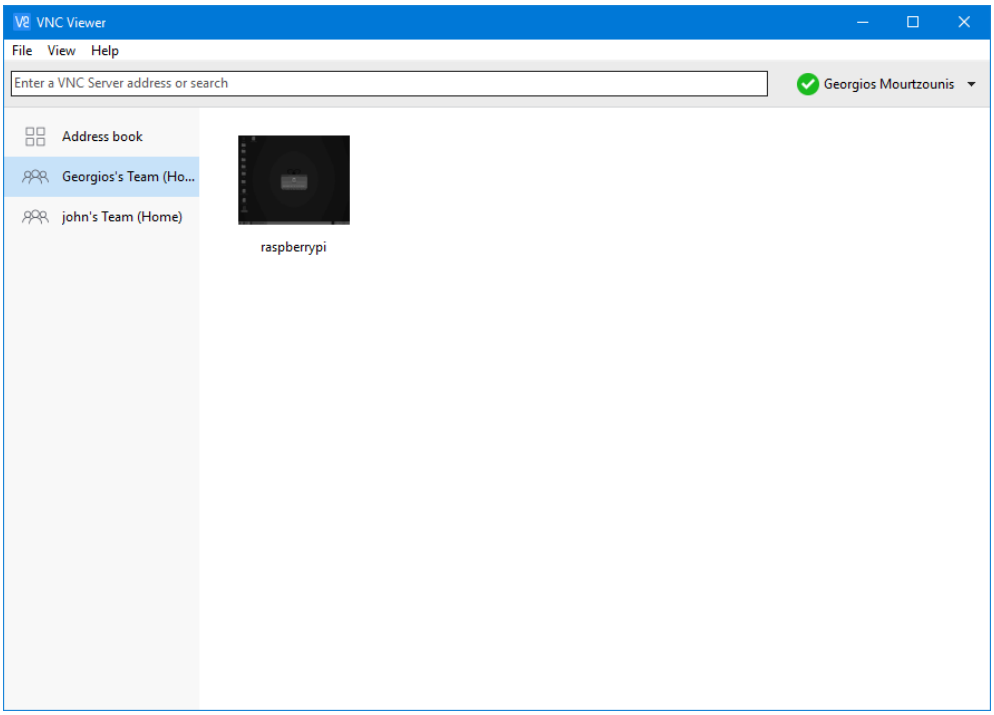
Εικόνα 2.1: Αναζήτηση Raspberry μέσω IP

5. (Τρόπος Cloud). Ο χρήστης χρειάζεται μονάχα να δημιουργήσει ένα λογαριασμό στον ακόλουθο σύνδεσμο. Έπειτα κάνει σύνδεση με τα στοιχεία του, τόσο στον υπολογιστή όσο και στο Raspberry.



Εικόνα 2.2: Σύνδεση στο Cloud

Στο VNC Viewer υπάρχει πλέον ένα κουτάκι που υποδεικνύει τη σύνδεση με το Raspberry κι ο χρήστης χρειάζεται μόνο να πατήσει διπλό κλικ πάνω του.



Εικόνα 2.3: Αυτόματη Σύνδεση

Στη συνέχεια θα ερωτηθεί για τα στοιχεία εισόδου username και password, τα οποία είναι pi και raspberry αντίστοιχα. Ωστόσο, συνιστάται η αλλαγή των στοιχείων αυτών για περεταίρω ασφάλεια και προστασία της ιδιωτικότητας (privacy) του χρήστη.

Παρόλο που η απευθείας σύνδεση μέσω τοπικού δικτύου (π.χ. Wi-Fi) είναι μια αρκετά γρήγορη κι εύκολη μέθοδος, ενδείκνυται αυτή του σύννεφου (Cloud) ειδικά όταν πρόκειται για συνδέσεις μέσω του διαδικτύου. Ο λόγος για τον οποίο προτιμάται το Cloud είναι η κρυπτογράφηση που παρέχει από άκρο σε άκρο (end-to-end) όπως επίσης διότι δεν είναι αναγκαίο να γνωρίζει κανείς την IP διεύθυνση του Raspberry ή να παρέχει μια στατική.

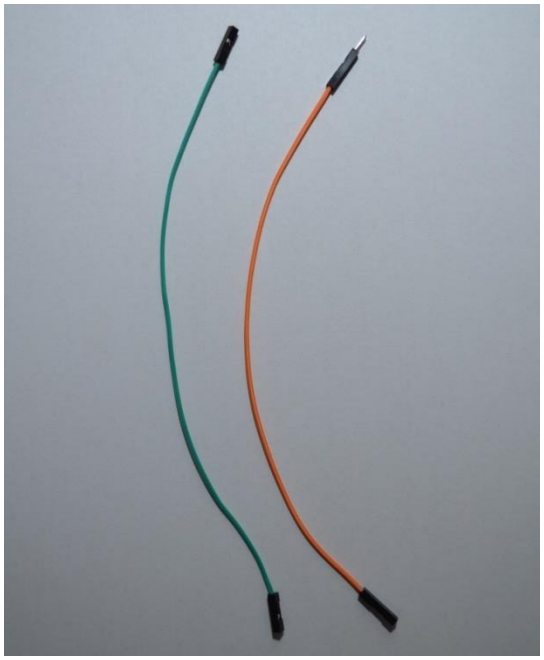
Συνοψίζοντας, υπό τη μορφή «Headless», το Raspberry Pi είναι πιο λειτουργικό κι εργονομικό καθώς μπορεί να χρησιμοποιηθεί σε μια πληθώρα εφαρμογών, χωρίς να δεσμεύει περιττό χώρο, ο οποίος σε πολλές περιπτώσεις φαντάζει πολυτέλεια.

ΕΞΑΡΤΗΜΑΤΑ ΑΥΤΟΝΟΜΟΥ ΟΧΗΜΑΤΟΣ

Στο τρίτο κατά σειρά κεφάλαιο, θα αναλυθούν τα εξαρτήματα, που ήταν απαραίτητα για την περάτωση της πτυχιακής εργασίας, όπως επίσης τα τεχνικά χαρακτηριστικά και το κόστους τους.

3.1 Εξαρτήματα και Προδιαγραφές

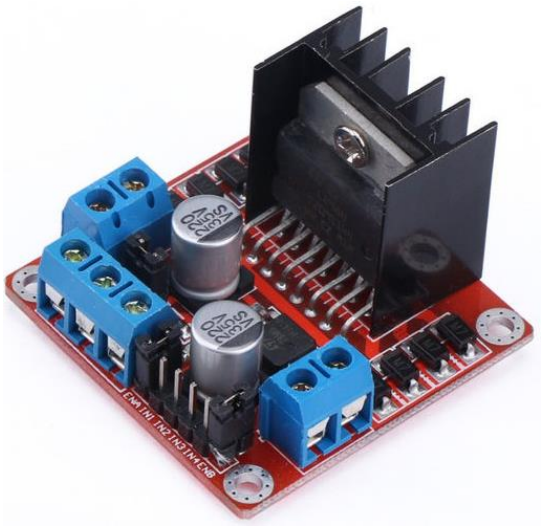
1. Καλώδια. Ένας καταλυτικός παράγοντας για την υλοποίηση της πτυχιακής εργασίας, καθώς φέρνουν σε «επικοινωνία» το Raspberry Pi, μέσω των ακίδων GPIO, με τον ελεγκτή L298n και τους DC κινητήρες. Χρησιμοποιήθηκαν δύο είδη καλωδίων, τα «female to female» και τα «male to female».
   Κόστος 0.5€



Εικόνα 3.1: Καλώδια Εργασίας

2. Ελεγκτής Κινητήρα L298n Dual H-Bridge. Η γέφυρα-Η είναι ένα κύκλωμα που μπορεί να ελέγχει το ρεύμα σε οποιαδήποτε πολικότητα και να ελέγχεται από το PWM,το οποίο είναι ένας τρόπος ελέγχου της διάρκειας ενός ηλεκτρονικού παλμού. Ο ελεγκτής λειτουργεί με τάση 5V, έχοντας παράλληλα τάση οδήγησης (drive voltage) 5V-35V, ρεύμα στα 0-36mA και ρεύμα οδήγησης

(drive current) 2A. Η μέγιστη ισχύς του αγγίζει τα 25W, ενώ οι διαστάσεις του είναι 43 x 43 x 26 χιλιοστά και ζυγίζει μόλις 26 γραμμάρια.
Κόστος: 5.90€



Εικόνα 3.2: Ελεγκτής L298n

3. Ενώ αρχικά είχε επιλεχθεί η αυθεντική Raspberry Pi Camera Module V2, κατά τη διάρκεια εκπόνησης της πτυχιακής εργασίας η επίδοσή της ήταν απογοητευτική, συνεπώς επιλέχθηκε μια απλή Web Camera, η οποία αποδείχθηκε αποδοτικότερη. Το μοντέλο που χρησιμοποιήθηκε είναι η Logitech C905.
Κόστος: 35€

Εικόνα 3.3: Web Camera

4. Για την αποφυγή σκόνης, βραχυκυκλωμάτων και γενικότερα την προστασία του Raspberry Pi χρησιμοποιήθηκε θήκη.
   Κόστος: 9.40€

Εικόνα 3.4: Θήκη Raspberry Pi

5. Power Bank. Επιλέχθηκε με την προοπτική να μην περιορίζεται η απόσταση που θα διανύσει το αμαξίδιο από το μάκρος του καλωδίου τροφοδοσίας. Το power bank είναι της μάρκας TP-LINK, έχει χωρητικότητα 10.400mAh και παρέχει ρεύμα της τάξεως του ενός και των δύο Ampere. Στην προκειμένη περίπτωση χρησιμοποιείται η έξοδος με τα δύο Ampere, καθώς είναι η ελάχιστη τροφοδοσία που μπορεί να δεχτεί το Raspberry Pi.
   Κόστος: 21€

Εικόνα 3.5: Φορητό Power Bank

6. USB Hub με τροφοδοσία. Για την ομαλή λειτουργία της Web κάμερας, απαιτείται ρεύμα της τάξης των 250mA.
   Κόστος: 5€

Εικόνα 3.6: Usb Hub με τροφοδοσία

7. DC κινητήρες. Χρησιμοποιούνται δύο , εξ αυτών ο πρώτος είναι υπεύθυνος για την κατεύθυνση κι ο δεύτερος για τη διεύθυνση του οχήματος.
   Κόστος: 0€ (αφαιρέθηκαν από παλιό τηλεκατευθυνόμενο)

Εικόνα 3.7: Τύπος κινητήρα εργασίας

8. Τροφοδοτικό. Χρησιμοποιείται κατά κύριο λόγο όσο το όχημα δε βρίσκεται εν κινήσει. Η τάση τροφοδοσίας του είναι στα 5V και το ρεύμα του φτάνει έως και 2.5A.
   Κόστος: 10€

Εικόνα 3.8: Αυθεντικό Τροφοδοτικό Raspberry Pi

9. SD Card. Όπως έχει προαναφερθεί, εδώ βρίσκεται το λειτουργικό σύστημα του Raspberry Pi και λόγω αυτού έχει επιλεχθεί η Sandisk Ultra microSDHC, η οποία είναι κλάσης 10, έχει χωρητικότητα 32GB,ταχύτητα ανάγνωσης 80MB/s και είναι γνωστή για την αξιοπιστία της στην ποιότητα κατασκευής.
   Κόστος: 18€

Εικόνα 3.9: Micro SD Κάρτα

10. Μπαταρίες. Χρησιμοποιήθηκαν έξι μπαταρίες για τη λειτουργία των δύο κινητήρων. Η τάση έκαστης μπαταρίας είναι 1.5V.
    Κόστος: 5€

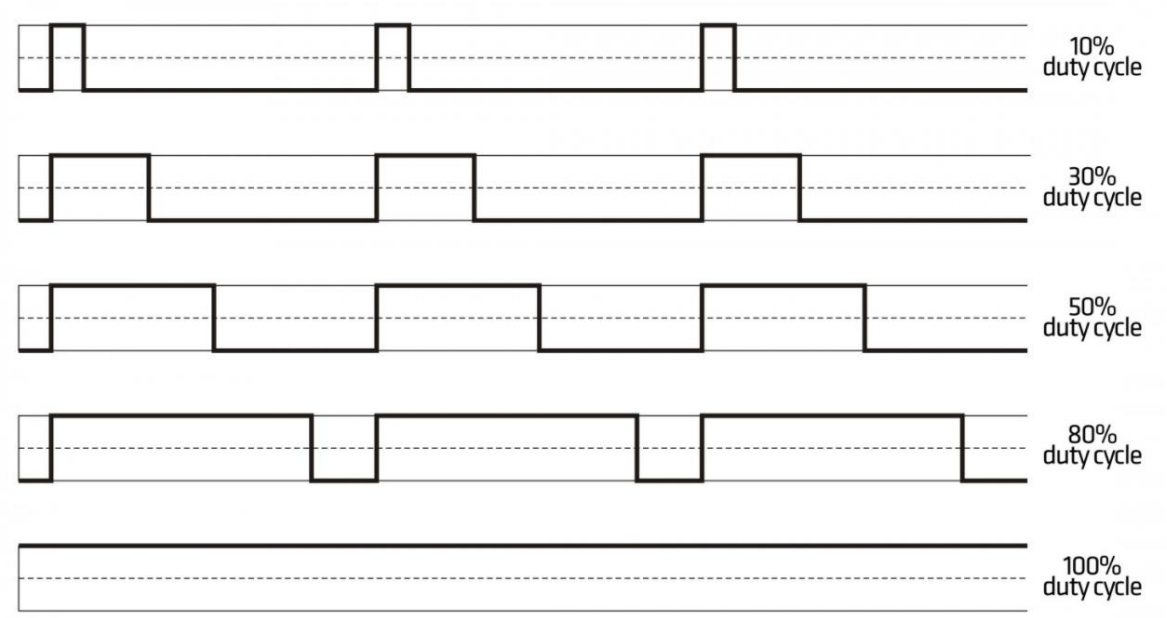
Εικόνα 3.10: Μπαταρίες

3.2 PWM

Η τεχνολογία P.W.M ή αλλιώς Pulse Width Modulation ανήκει στις τεχνολογίες αντιστροφέων (inverters), οι οποίοι χρησιμοποιούνται για την μετατροπή συνεχούς ρεύματος σε εναλλασσόμενο. Παρόλο που αυτή η τεχνική διαμόρφωσης μπορεί να χρησιμοποιηθεί για την κωδικοποίηση πληροφοριών για μετάδοση, η κύρια χρήση της είναι να επιτρέψει τον έλεγχο της ισχύος που παρέχεται στις ηλεκτρικές συσκευές, όπως οι κινητήρες. Η τεχνολογία P.W.M είναι μία συχνά χρησιμοποιούμενη τεχνική κατά την οποία η μέση τιμή της τάσης ή της έντασης του ρεύματος, που τροφοδοτεί ένα φορτίο, ελέγχεται μέσω ηλεκτρονικών διακοπτών μεταξύ της τροφοδοσίας και της κατανάλωσης ισχύος. Συγκεκριμένα, όσο περισσότερο είναι ενεργοποιημένος ο διακόπτης σε σχέση με τις περιόδους διακοπής, τόσο μεγαλύτερη είναι η συνολική ισχύς που παρέχεται στο φορτίο. Η συχνότητα μεταγωγής PWM πρέπει να είναι πολύ υψηλότερη από ό, τι θα επηρέαζε το φορτίο (τη συσκευή που χρησιμοποιεί την ισχύ), που σημαίνει ότι η προκύπτουσα κυματομορφή που αντιλαμβάνεται το φορτίο πρέπει να είναι όσο το δυνατόν πιο ομαλή. Ο ρυθμός (ή η συχνότητα) στον οποίο πρέπει να αλλάξει η τροφοδοσία μπορεί να ποικίλει σημαντικά ανάλογα με το φορτίο και την εφαρμογή.

Ο όρος duty cycle (κύκλος εργασίας) περιγράφει το ποσοστό του χρόνου ενεργοποίησης στην περίοδο του χρόνου. Ένας κύκλος χαμηλής κατανάλωσης αντιστοιχεί σε χαμηλή ισχύ, επειδή η ισχύς είναι απενεργοποιημένη για το μεγαλύτερο μέρος του χρόνου. Ο κύκλος εργασίας εκφράζεται σε ποσοστό τις

εκατό, ενώ στο 100% είναι πλήρως ενεργοποιημένο. Το κύριο πλεονέκτημα του PWM είναι ότι η απώλεια ισχύος στις συσκευές μεταγωγής είναι πολύ χαμηλή. Όταν ένας διακόπτης είναι απενεργοποιημένος, δεν υπάρχει σχεδόν κανένα ρεύμα και όταν είναι ενεργοποιημένος και η ισχύς μεταφέρεται στο φορτίο, δεν υπάρχει σχεδόν καμία πτώση τάσης στον διακόπτη. Η απώλεια ισχύος, που είναι το προϊόν της τάσης και του ρεύματος, είναι και στις δύο περιπτώσεις κοντά στο μηδέν.



Εικόνα 3.11: PWM Duty Cycle

(Κενό φύλλο)

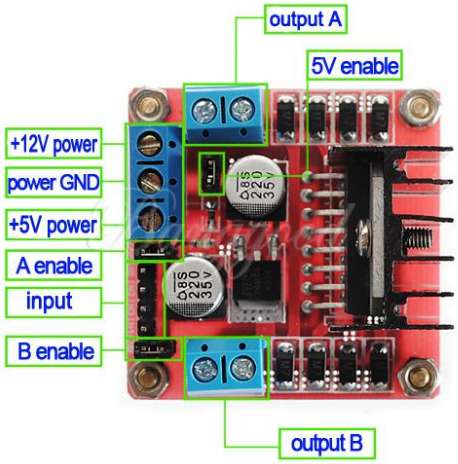
ΣΥΝΔΕΣΜΟΛΟΓΙΑ ΚΑΛΩΔΙΩΝ

Σε αυτό το κεφάλαιο αναλύεται η συνδεσμολογία του αυτόνομου οχήματος τόσο από άποψη κυκλώματος, όσο κι από εξαρτήματα και συσκευές που το απαρτίζουν.

4.1 L298N Dual H Bridge Motor Driver Pinout

Όπως έχει προαναφερθεί, ο συγκεκριμένος ελεγκτής στην παρούσα πτυχιακή εργασία χρησιμοποιείται για τον έλεγχο της ταχύτητας και της κατεύθυνσης των κινητήρων. Προτού δειχθεί η καλωδίωση, που αφορά το συγκεκριμένο εξάρτημα, θα αναλυθούν οι ακίδες του ούτως ώστε να κατανοηθεί πλήρως ο σκοπός κάθε καλωδίου.

- Output A1: Έξοδος κινητήρα A (1)
- Output A2: Έξοδος κινητήρα A (2)
- Output B1: Έξοδος κινητήρα B (1)
- Output B2: Έξοδος κινητήρα B (2)
- +12v Power: Τάση Εισόδου 12 V
- Power GND: Γείωση
- +5v Power: Τάση Εισόδου 5 Volt
- A enable: Σήμα PWM στον κινητήρα A
- Input 1: Είσοδος κινητήρα A (1)
- Input 2: Είσοδος κινητήρα A (2)
- Input 3: Είσοδος κινητήρα B (1)
- Input 4: Είσοδος κινητήρα B (2)
- B enable: Σήμα PWM στον κινητήρα B

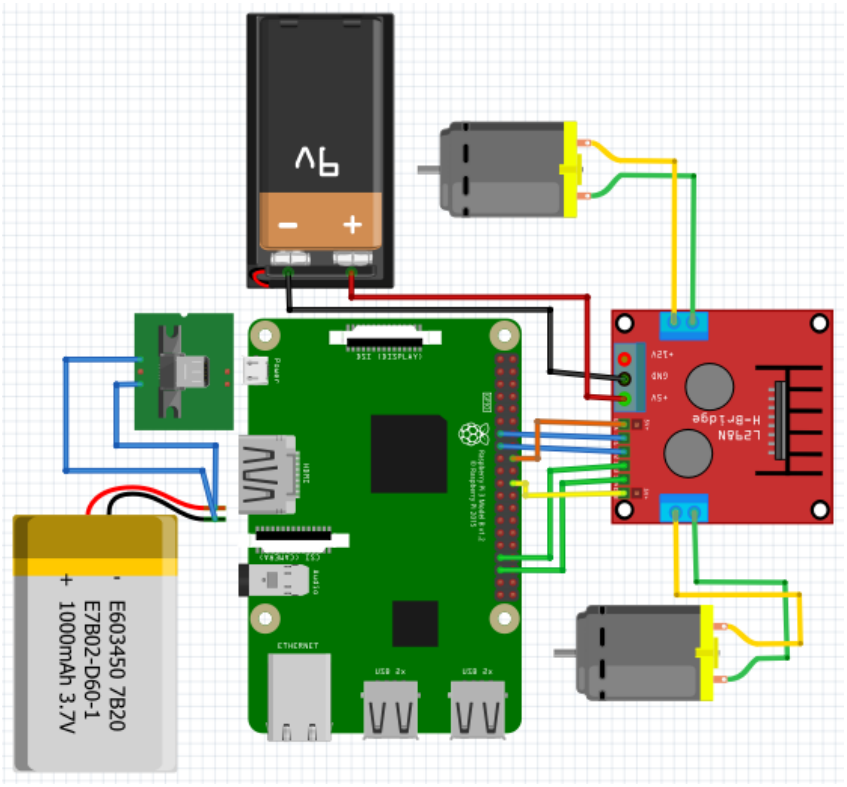


Εικόνα 4.1: L298n pinout

4.2 Συνδεσμολογία οχήματος

Η τροφοδοσία επιτυγχάνεται από τις επαφές των 12V και της γείωσης, στις οποίες είναι συνδεδεμένα τα δύο καλώδια της ενσωματωμένης βάσης μπαταριών του αμαξιδίου. Τα enable A και B, συνδέονται με τις ακίδες GPIO 25 και 24 αντίστοιχα του Raspberry και είναι υπεύθυνα για το σήμα PWM προς τους κινητήρες. Τα Input 1,2 αφορούν τον πρώτο κινητήρα και συνδέονται με τις ακίδες GPIO 22 και 27, ενώ τα Input 3,4 το δεύτερο, ο οποίος συνδέεται με τις ακίδες GPIO 13 και 19. Στα Output A και B είναι συνδεδεμένα τα καλώδια των δύο κινητήρων, οι οποίοι πραγματοποιούν την έξοδο, δηλαδή λειτουργούν βάσει του ελέγχου που έχουν δεχθεί από τον κώδικα.

Το αυτόνομο όχημα είναι εξοπλισμένο με power bank, το οποίο έχει δύο θύρες φόρτισης. Στη θύρα των 2A βρίσκεται ένα USB, το οποίο καταλήγει σε micro και συνδέεται στην κατάλληλη θύρα του Raspberry τροφοδοτώντας το, ενώ από τη θύρα του 1A τροφοδοτείται το USB Hub. Σε αυτό είναι συνδεδεμένη η Web Camera και επικοινωνεί με το Raspberry μέσω θύρας USB. Παρακάτω παρουσιάζεται η συνδεσμολογία Raspberry, L298n και των κινητήρων:



Εικόνα 4.2: Συνδεσμολογία κυκλώματος

PYTHON ΚΑΙ OPENCV

Στο συγκεκριμένο κεφάλαιο αναλύονται αφενός τα χαρακτηριστικά της γλώσσας προγραμματισμού που χρησιμοποιήθηκε, κι αφετέρου της βιβλιοθήκης OpenCV, όπως επίσης και τα πλεονεκτήματά της. Τέλος παρουσιάζονται οι λόγοι για τους οποίους γίνεται ο συνδυασμός των παραπάνω στην προκειμένη πτυχιακή εργασία και ο τρόπος εγκατάστασης της συγκεκριμένης βιβλιοθήκης.

5.1 Χαρακτηριστικά της γλώσσας Python

Η Python είναι μια γλώσσα προγραμματισμού υψηλού επιπέδου, ανοιχτού κώδικα κι έχει απλή, αλλά αποτελεσματική προσέγγιση στον αντικειμενοστραφή προγραμματισμό. Δημιουργήθηκε από τον Ολλανδό Guido van Rossum το 1990 και η φιλοσοφία της είναι η αναγνωσιμότητα του κώδικά της, η ευκολία χρήσης της και το κομψό συντακτικό της μαζί με τη λειτουργία της ως διερμηνευόμενης (αντί μεταγλωττιζόμενης) γλώσσας, τα οποία την καθιστούν ιδανική για δημιουργία σεναρίων εντολών, αλλά και για την ταχεία ανάπτυξη εφαρμογών στις περισσότερες πλατφόρμες. Επιτρέπει στους προγραμματιστές να εκφράσουν έννοιες σε λιγότερες γραμμές κώδικα απ' ότι θα ήταν δυνατόν σε γλώσσες όπως η C++ ή η Java, λόγω του μεγάλου πλήθους βιβλιοθηκών που διευκολύνουν ιδιαίτερα αρκετές συνηθισμένες εργασίες.

Εικόνα 5.1: Λογότυπο Python

Η νέα έκδοση της γλώσσας είναι η Python 3.0 γνωστή και ως Python 3000 ή Py3K. Ο κύριος λόγος για μια μεγάλη νέα έκδοση της Python είναι να απομακρυνθούν όλα τα μικροπροβλήματα και ελαττώματα που συσσωρεύτηκαν με τα χρόνια, και να γίνει η γλώσσα ακόμα πιο καθαρή. Παρακάτω περιγράφονται τα χαρακτηριστικά της γλώσσας:

- **Εύκολη Εκμάθηση**

- **Αναγνωσιμότητα (καθαρό, αναγνώσιμο συντακτικό)**

- **Συντήρηση**

- **Ανοικτού Κώδικα**

- **Απλή, Γρήγορη Ανάπτυξη Εφαρμογών**

- **Διερμηνευόμενη και Φορητή**

Ένα πρόγραμμα που γράφεται σε μια μεταγλωττιζόμενη γλώσσα όπως η C ή η C++ μετατρέπεται από την πηγαία γλώσσα, σε γλώσσα που “μιλάει” ο υπολογιστής χρησιμοποιώντας ένα μεταγλωττιστή με διάφορες σημαίες και επιλογές. Όταν εκτελείται το πρόγραμμα, ο συνδέτης αντιγράφει το πρόγραμμα στη μνήμη και αρχίζει να το τρέχει. Η Python, από την άλλη, δε χρειάζεται μεταγλώττιση σε δυαδικό αρχείο διότι εκτελείται το πρόγραμμα απ' ευθείας από τον πηγαίο κώδικα. Εσωτερικά, η Python μετατρέπει τον πηγαίο κώδικα σε μια ενδιάμεση μορφή, που ονομάζεται bytecode, το μεταφράζει σε γλώσσα μηχανής κι έπειτα το τρέχει. Όλο αυτό, στην πραγματικότητα κάνει τη χρήση της Python πολύ πιο εύκολη όπως επίσης κάνει τα προγράμματα της εξαιρετικά φορητά.

- **Πολύ υψηλού επιπέδου δομές δεδομένων**

- **Επεκτάσιμη**

Αν χρειαστεί ένα κρίσιμο κομμάτι κώδικα να τρέχει πολύ γρήγορα ή αν πρέπει να συμπεριληφθεί στο πρόγραμμα ένα κομμάτι ενός αλγόριθμου, που να μην είναι ανοικτό, τότε μπορεί ο προγραμματιστής να προγραμματίσει εκείνο το κομμάτι σε C ή C++ και μετά να το χρησιμοποιήσει στο κυρίως Python πρόγραμμα

- **Ενσωματώσιμη**

Ένα πρόγραμμα σε Python μπορεί να ενσωματωθεί μέσα στα προγράμματα σε C/C++ παρέχοντας δυνατότητες 'scripting' στους χρήστες.

- **Εκτεταμένες βιβλιοθήκες**

- Η Πρότυπη βιβλιοθήκη της Python είναι πραγματικά τεράστια. Η χρήση της μπορεί να βοηθήσει σε διάφορες περιπτώσεις, σχετικά με κανονικές εκφράσεις, δημιουργία τεκμηρίωσης, δοκιμές μονάδων, νημάτωση, βάσεις δεδομένων, περιηγητές ιστού, CGI, FTP, email, XML, XML-RPC, HTML, αρχεία WAV, κρυπτογράφηση, γραφικές διεπαφές χρήστη (GUI), Tk, κι άλλα πράγματα, που εξαρτούνται από το σύστημα. Πρόκειται, λοιπόν, για τη φιλοσοφία της Python, «Batteries Included», της οποίας οι λειτουργίες είναι διαθέσιμες, εφόσον η γλώσσα είναι εγκατεστημένη. Επιπλεόν από την πρότυπη βιβλιοθήκη, υπάρχουν διάφορες άλλες βιβλιοθήκες υψηλής ποιότητας όπως η wxPython , η Twisted, η Python Imaging Library και πολλές άλλες.

- **Δεν παρουσιάζει πλέον segmentation faults**

- **Αυτόματη διαχείριση μνήμης**

Από την αρχή της ανάπτυξής της, η Python αποσκοπεί στην απλούστερη μορφή συγγραφής κώδικα. Αυτό συμβαίνει τόσο για την εκμάθηση της γλώσσας, όπου στόχος είναι ο ενστερνισμός της, όσο και για την αναγνωσιμότητα του παραγόμενου κώδικα. Απότοκος των παραπάνω είναι η ευκολία συντήρησης του κώδικα και η επέκτασή του. Στην ανάπτυξη λογισμικού κυριαρχεί η ιδεολογία πως «*Η αποσφαλμάτωση απαιτεί τη διπλάσια ευφυΐα από αυτή που γράφτηκε ο κώδικας*». Οπότε η περεταίρω πολυπλοκότητα του κώδικα, συνεπάγεται με ακόμη πιο δύσκολη αποσφαλμάτωση. Όλα τα παραπάνω συνηγορούν στην δυνατότητα της Python να επιτρέπει την ταχύτατη ανάπτυξη εφαρμογών ειδικά σε σχέση με άλλες γλώσσες χαμηλότερου επιπέδου (π.χ. C, C++) ενώ λέγεται ότι συνήθως τα προγράμματα της είναι τρεις έως πέντε φορές μικρότερα σε σχέση με τα αντίστοιχα σε Java.

5.2 Εισαγωγή στην OpenCV

Η OpenCv είναι μια βιβλιοθήκη μηχανικής όρασης (computer vision) και μηχανικής εκμάθησης (machine learning) ανοικτού λογισμικού, που αναπτύχθηκε στην Intel το 1999 από τον Gary Bradsky. Είναι ελεύθερη για εμπορική και ερευνητική χρήση υπό την άδεια ανοικτού λογισμικού BSD και είναι ανεξάρτητη πλατφόρμας. Χρησιμοποιεί περισσότερες από 500 συναρτήσεις γραμμένες σε C και C++ και έχει σχεδιαστεί με στόχο την αποτελεσματική εκμετάλλευση των υπολογιστικών πόρων, με έμφαση στις εφαρμογές πραγματικού χρόνου.

Εικόνα 5.2: Λογότυπο OpenCV

Η βιβλιοθήκη περιέχει περισσότερους από 2500 βελτιστοποιημένους αλγορίθμους, οι οποίοι περιλαμβάνουν ένα ολοκληρωμένο σύνολο αλγορίθμων τεχνητής οράσεως και μηχανικής μάθησης. Αυτοί οι αλγόριθμοι μπορούν να χρησιμοποιηθούν για την ταχεία ανίχνευση και αναγνώριση προσώπων, τον εντοπισμό αντικειμένων, την ταξινόμηση των ανθρώπινων ενεργειών σε βίντεο, την παρακολούθηση κινήσεων κάμερας, την παρακολούθηση κινούμενων αντικειμένων και σε πάρα πολλές ακόμη περιπτώσεις επεξεργασίας και ανάλυσης εικόνων βίντεο.

5.3 Πλεονεκτήματα της OpenCV

Το κυριότερο πλεονέκτημα της OpenCv είναι η ταχύτητα. Αυτός είναι κι ο βασικός λόγος, που δημιουργήθηκε εξ αρχής, καθώς η Intel ήθελε να δείξει πόσο προχωρημένοι ήταν οι επεξεργαστές τις που μπορούσαν να επεξεργάζονται βίντεο σε πραγματικό χρόνο.

Το δεύτερο είναι ότι είναι ελεύθερο λογισμικό κι υποστηρίζεται από εταιρίες και μεγάλο πλήθος προγραμματιστών συμβάλλοντας διαρκώς στην ανάπτυξή της.

Ένα άλλο πλεονέκτημα είναι ότι δεν απαιτεί περίπλοκα μηχανήματα και εξοπλισμούς, αφού κι η πιο απλή webcam είναι αρκετή για είσοδο εικόνας και ο πιο απλός προσωπικός υπολογιστής είναι ικανός για την επεξεργασία της.

5.4 OpenCV - Python

Εικόνα 5.3: Λογότυπα OpenCV, Python

Όπως προαναφέρθηκε, η Python είναι μια γενική γλώσσα προγραμματισμού η οποία έγινε πολύ δημοφιλής σε σύντομο χρονικό διάστημα κυρίως λόγω της απλότητας και της αναγνωσιμότητας του κώδικά της. Επιτρέπει στον προγραμματιστή να εκφράζει τις ιδέες του σε λιγότερες γραμμές κώδικα, χωρίς να μειώνει την αναγνωσιμότητα. Σε σύγκριση με άλλες γλώσσες όπως C / C ++, η Python είναι πιο αργή αλλά μπορεί να επεκταθεί εύκολα με αυτές. Κατ' αυτό τον τρόπο υπάρχει η δυνατότητα συγγραφής κώδικα σε C / C ++ και να χρησιμοποιηθεί ως module της Python. Αυτό συνεπάγεται με δύο πλεονεκτήματα: αφενός ο κώδικας είναι τόσο γρήγορος, όσο κι ο αρχικός της C / C ++ (δεδομένου ότι είναι ο πραγματικός κώδικας C ++ που λειτουργεί στο παρασκήνιο) κι αφετέρου είναι πολύ εύκολο να κωδικοποιηθεί στην Python. Με άλλα λόγια το OpenCV-Python, είναι ένα περιτύλιγμα Python γύρω από την αρχική εφαρμογή C ++. Παράλληλα η υποστήριξη του Numpy καθιστά την εργασία πιο εύκολη, καθώς είναι μια ιδιαίτερα βελτιστοποιημένη βιβλιοθήκη για αριθμητικές λειτουργίες κι η σύνταξή της είναι τύπου MATLAB. Όλες οι δομές του πίνακα OpenCV μετατρέπονται σε-και-από τους αριθμητικούς πίνακες, έτσι, ανεξάρτητα από τις λειτουργίες, που μπορούν να γίνουν με την Numpy, μπορεί να συνδυαστεί με το OpenCV. Τέλος, αρκετές άλλες βιβλιοθήκες όπως το SciPy, το Matplotlib, που υποστηρίζει την Numpy, μπορούν να χρησιμοποιηθούν με αυτό καθιστώντας το OpenCV-Python ένα κατάλληλο εργαλείο για την ταχεία δημιουργία πρωτοτύπων των προβλημάτων ηλεκτρονικής όρασης.

5.5 Εγκατάσταση OpenCV

1. Το πρώτο βήμα είναι να ενημερωθούν και να αναβαθμιστούν τυχόν υπάρχοντα πακέτα:

```
$ sudo apt-get update
```

```
$ sudo apt-get upgrade
```

2. Έπειτα χρειάζεται να εγκατασταθούν κάποια εργαλεία όπως το cmake τα οποία μας βοηθούν στην εγκατάσταση του opencv.

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

3. Με την εγκατάσταση των παρακάτω βιβλιοθηκών δίνεται η δυνατότητα φόρτωσης από τον δίσκο αρχείων εικόνας και βίντεο διαφόρων τύπων.

```
$ sudo apt-get install libjpeg-dev libtiff5-dev libjasper-dev libpng12-dev
$ sudo apt-get install libavcodec-dev libavformat-dev libswscale-dev libv4l-dev
$ sudo apt-get install libxvidcore-dev libx264-dev
```

4. Προκειμένου να καταρτίσουμε το module highgui, (η χρήση του είναι για την εμφάνιση εικόνων στην οθόνη και για την ανάπτυξη βασικών gui), πρέπει να εγκαταστήσουμε την βιβλιοθήκη ανάπτυξης GTK:

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

5. Κάποιες επιπλέον προσθήκες για τη βελτίωση των πράξεων με πίνακες.

```
$ sudo apt-get install libatlas-base-dev gfortran
```

6. Εγκατάσταση της γλώσσας python2.7 και python3 (ή της μίας μόνο έκδοσης).

```
$ sudo apt-get install python2.7-dev python3-dev
```

7. Κατεβάζουμε την έκδοση του opencv που χρειαζόμαστε και την αποσυμπιέζουμε.

```
$ wget -O opencv.zip https://github.com/Itseez/opencv/archive/3.1.0.zip
$ unzip opencv.zip
```

```
$ wget -O opencv_contrib.zip
https://github.com/Itseez/opencv_contrib/archive/3.1.0.zip
$ unzip opencv_contrib.zip
```

8. Εγκατάσταση του pip διαχειριστή πακέτων.

```
$ wget https://bootstrap.pypa.io/get-pip.py
$ sudo python get-pip.py
$ sudo pip install virtualenv virtualenvwrapper
$ sudo rm -rf ~/.cache/pip
```

9. Εγκατάσταση ενός ψηφιακού περιβάλλοντος όπου εκεί θα δουλεύουμε.

```
# virtualenv and virtualenvwrapper
export WORKON_HOME=$HOME/.virtualenvs
source /usr/local/bin/virtualenvwrapper.sh
$ echo -e "\n# virtualenv and virtualenvwrapper" >> ~/.profile
$ echo "export WORKON_HOME=$HOME/.virtualenvs" >> ~/.profile
$ echo "source /usr/local/bin/virtualenvwrapper.sh" >> ~/.profile
```

10. Φορτώνουμε ξανά το ~/.profile για να δούμε ότι όλα λειτουργούν σωστά.

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

11. Δημιουργία του ψηφιακού περιβάλλοντος cv στην έκδοση 3 της python όπου θα χρησιμοποιηθεί για την ανάπτυξη της εργασίας με τεχνητή όραση.

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

12. Έλεγχος ότι βρισκόμαστε στο περιβάλλον cv (θα πρέπει να βλέπουμε τη λέξη cv αριστερά στο τερματικό).

```
$ source ~/.profile
$ workon cv
```

13. Εγκατάσταση του numpy, ενός πακέτου για αριθμητικές πράξεις.

```
$ pip install numpy
```

14. Εγκατάσταση της opencv με τη χρήση του cmake.

```
$ cd ~/opencv-3.1.0/
$ mkdir build
$ cd build
$ cmake -D CMAKE_BUILD_TYPE=RELEASE \
    -D CMAKE_INSTALL_PREFIX=/usr/local \
    -D INSTALL_PYTHON_EXAMPLES=ON \
    -D OPENCV_EXTRA_MODULES_PATH=~/opencv_contrib-
3.1.0/modules \
    -D BUILD_EXAMPLES=ON ..
```

15. Κάνοντας compile την opencv με τη χρήση 4 πυρήνων.

```
$ make -j4
```

16. Τελικά βήματα για την ολοκλήρωση της εγκατάστασης του opencv

```
$ sudo make install
$ sudo ldconfig
$ ls -l /usr/local/lib/python3.4/site-packages/
total 1852
-rw-r--r-- 1 root staff 1895932 Mar 20 21:51 cv2.cpython-34m.so
$ cd /usr/local/lib/python3.4/site-packages/
$ sudo mv cv2.cpython-34m.so cv2.so
$ cd ~/.virtualenvs/cv/lib/python3.4/site-packages/
$ ln -s /usr/local/lib/python3.4/site-packages/cv2.so cv2.so
```

ΑΝΑΛΥΣΗ ΚΩΔΙΚΑ

Στο συγκεκριμένο κεφάλαιο πρόκειται να γίνει ανάλυση των συναρτήσεων και αλγορίθμων, που χρησιμοποιήθηκαν για την υλοποίηση της πτυχιακής εργασίας όσον αφορά το προγραμματιστικό κομμάτι.

6.1 Παρουσίαση βιβλιοθηκών του κώδικα

Οι βιβλιοθήκες είναι ένα από τα πιο καίρια «εργαλεία» στην ανάπτυξη λογισμικού, καθώς περιλαμβάνουν τεκμηριώσεις (documentation), δεδομένα διαμόρφωσης (configuration data) και προγραμμένο κώδικα, υπορουτίνες και κλάσεις. Στη συνέχεια θα αναλυθούν αυτές που χρησιμοποιήθηκαν στο πρόγραμμα της προκειμένης πτυχιακής εργασίας.

```
import cv2   #Εισαγωγή της OpenCV
import time  #Εισαγωγή της βιβλιοθήκης time-sleep
from time import sleep
import RPi.GPIO as GPIO #Χρήση των GPIO
import numpy as np     #Εισαγωγή της Numpy για αριθμητικές πράξεις
import warnings       #Απόκρυψη των warnings
warnings.filterwarnings('ignore')
import tkinter as tk   #Εισαγωγή του πακέτου γραφικών tkinter
from tkinter import *
import scipy           #Εισαγωγή του scipy-find_peaks_cwt για αριθμητικές πράξεις
from scipy.signal import find_peaks_cwt
import imutils   #Εισαγωγή του imutils-WebcamVideoStream για τις βασικές λειτουργίες
from imutils.video import WebcamVideoStream #στο διάβασμα και εμφάνιση των frames
```

Εικόνα 6.1: Δήλωση βιβλιοθηκών

6.2 Ανάλυση κλάσεων και συναρτήσεων

Class Motor

def __init__

Σε αυτή την συνάρτηση ορίζονται το PWM και οι ακίδες GPIO, στις οποίες είναι συνδεδεμένοι οι δύο κινητήρες. Αρχικά, έχουν ορισθεί οι κατευθύνσεις βάσει της πολικότητάς τους κι έχουν τεθεί σε κατάσταση ακινησίας.

```
class Motor:

    def __init__(self,PinForward,PinBackward,PinRight,PinLeft,PinControlForward,PinControlSteering):

        self.PinLeft = PinLeft #13
        self.PinRight = PinRight#19
        self.PinForward = PinForward #27
        self.PinBackward = PinBackward #22
        self.PinControlForward = PinControlForward #25
        self.PinControlSteering = PinControlSteering #24

        GPIO.setmode(GPIO.BCM)
        GPIO.setup(self.PinLeft, GPIO.OUT)
        GPIO.setup(self.PinRight, GPIO.OUT)
        GPIO.setup(self.PinForward, GPIO.OUT)
        GPIO.setup(self.PinBackward, GPIO.OUT)
        GPIO.setup(self.PinControlForward, GPIO.OUT)
        GPIO.setup(self.PinControlSteering, GPIO.OUT)

        self.pwm_forward = GPIO.PWM(self.PinForward,100)
        self.pwm_forward.start(0)

        self.pwm_backward = GPIO.PWM(self.PinBackward,100)
        self.pwm_backward.start(0)

        self.pwm_left = GPIO.PWM(self.PinLeft,100)
        self.pwm_left.start(0)

        self.pwm_right = GPIO.PWM(self.PinRight,100)
        self.pwm_right.start(0)

        GPIO.output(self.PinControlForward, GPIO.HIGH)
        GPIO.output(self.PinControlSteering, GPIO.HIGH)

    def forward_left(self,speed):
        self.pwm_left.ChangeDutyCycle(100)
        self.pwm_forward.ChangeDutyCycle(speed)
        self.pwm_right.ChangeDutyCycle(0)

    def forward_right(self,speed):
        self.pwm_right.ChangeDutyCycle(100)
        self.pwm_forward.ChangeDutyCycle(speed)
        self.pwm_left.ChangeDutyCycle(0)

    def forward(self,speed):
        self.pwm_forward.ChangeDutyCycle(speed)

    def backward_left(self,speed):
        self.pwm_left.ChangeDutyCycle(100)
        self.pwm_backward.ChangeDutyCycle(speed)
        self.pwm_right.ChangeDutyCycle(0)

    def backward_right(self,speed):
        self.pwm_right.ChangeDutyCycle(100)
        self.pwm_backward.ChangeDutyCycle(speed)
        self.pwm_left.ChangeDutyCycle(0)

    def backward(self,speed):
        self.pwm_backward.ChangeDutyCycle(speed)

    def stop(self):
        self.pwm_forward.ChangeDutyCycle(0)
        self.pwm_right.ChangeDutyCycle(0)
        self.pwm_left.ChangeDutyCycle(0)
```

Εικόνα 6.2: Κλάση Motor

def forward_left

Στην ακίδα που ευθύνεται για την αριστερή στροφή των τροχών δίνεται PWM 100, στην ακίδα που κινεί μπροστά το όχημα δίνεται PWM ανάλογο της ταχύτητας που έχει βάσει μεταβλητής που θα αναλυθεί παρακάτω. Τέλος στην ακίδα που στρίβει δεξιά τους τροχούς, το PWM μηδενίζεται.

def forward_right

Στην ακίδα που ευθύνεται για την δεξιά στροφή των τροχών δίνεται PWM 100, στην ακίδα που κινεί μπροστά το όχημα δίνεται PWM ανάλογο της ταχύτητας που έχει βάσει μεταβλητής που θα αναλυθεί παρακάτω. Τέλος στην ακίδα που στρίβει αριστερή τους τροχούς, το PWM μηδενίζεται.

def forward

Όταν κληθεί η συγκεκριμένη συνάρτηση, το PWM της ακίδας, που ευθύνεται για την πρόσω κίνηση του οχήματος, παίρνει την τιμή της μεταβλητής speed.

def backward_left

Στην ακίδα που ευθύνεται για την αριστερή στροφή των τροχών δίνεται PWM 100, στην ακίδα που κινεί πίσω το όχημα δίνεται PWM ανάλογο της ταχύτητας που έχει βάσει μεταβλητής που θα αναλυθεί παρακάτω. Τέλος στην ακίδα που στρίβει δεξιά τους τροχούς, το PWM μηδενίζεται.

def backward_right

Στην ακίδα που ευθύνεται για την δεξιά στροφή των τροχών δίνεται PWM 100, στην ακίδα που κινεί πίσω το όχημα δίνεται PWM ανάλογο της ταχύτητας που έχει βάσει μεταβλητής που θα αναλυθεί παρακάτω. Τέλος στην ακίδα που στρίβει αριστερή τους τροχούς, το PWM μηδενίζεται.

def backward

Όταν κληθεί η συγκεκριμένη συνάρτηση, το PWM της ακίδας, που ευθύνεται για την όπισθεν κίνηση του οχήματος, παίρνει την τιμή της μεταβλητής speed.

def stop

Όταν κληθεί αυτή η συνάρτηση, μηδενίζεται το PWM και των δύο κινητήρων.

Class Threaded_Frame

def __init__(self, src)

Από τη μεταβλητή src, αναγνωρίζεται η υποδοχή της web camera και μέσω αυτής, διαβάζονται τα frames από το WebcamVideoStream της βιβλιοθήκης imutils.

def start(self)

Ξεκινάει η ροή του βίντεο με το νήμα. Το νήμα χρησιμοποιείται για ταχύτερη ανάγνωση των frames από την κάμερα, λόγω της περιορισμένης επεξεργαστικής

ισχύος του Raspberry Pi συναρτήσει των αυξημένων απαιτήσεων του προγράμματος που αναπτύσσεται στην παρούσα πτυχιακή εργασία.

def read(self)

Κατά την κλήση της, διαβάζει τη ροή του βίντεο(stream) από τη συνάρτηση def start(self) και επιστρέφει το τρέχον πλαίσιο(frame).

def stop(self)

Σταματάει το νήμα και τερματίζεται το stream από την κάμερα.

---

def gaussian_blur(frm)

```
def gaussian_blur(frm):
    return cv2.GaussianBlur(frm,(5,5),0)
```

Εικόνα 6.3: Συνάρτηση Gaussian blur

Χρησιμοποιείται για να θολώσει το frame, να απαλείψει το θόρυβο και τις λεπτομέρειες με μάσκα 5 επί 5.

def apply_sobel – def sobel_wrap

```
def apply_sobel(frame_HLS):
    L_channel = frame_HLS[:,:,1]
    L_Sobel = sobel_wrap(L_channel)
    S_channel = frame_HLS[:,:,2]
    S_Sobel = sobel_wrap(S_channel)

    wraped_LS = cv2.bitwise_and(S_Sobel,L_Sobel)
    wraped_LS = np.uint8(wraped_LS)
    wraped_LS = gaussian_blur(wraped_LS)
    return wraped_LS

def sobel_wrap(frm_gs):
    rr,bin_ = cv2.threshold(frm_gs,180,255,cv2.THRESH_BINARY)
    laplacian = cv2.Laplacian(bin_,cv2.CV_64F)
    sobelx = cv2.Sobel(bin_,cv2.CV_64F,1,0,ksize=5)
    sobely = cv2.Sobel(bin_,cv2.CV_64F,0,1,ksize=5)
    return cv2.bitwise_or(sobelx,sobely)
```

Εικόνα 6.4: Συνάρτηση Apply Sobel /Sobel wrap

Εκτός από τη μάσκα χρώματος, εφαρμόζουμε φίλτρο sobel για την ανίχνευση των άκρων. Εφαρμόζουμε φίλτρο sobel στα κανάλια L και S του frame, καθώς αυτά βρέθηκαν να είναι πιο έντονα στο χρώμα και τις παραλλαγές του φωτισμού.

Εικόνα 6.5: Φίλτρο Sobel

def apply_color_mask

```
def apply_color_mask(hsv,frm):
    white_hsv_low  = np.array([   0,    0, white_low.get()])
    white_hsv_high = np.array([ 255,  255, 255])
    mask = cv2.inRange(hsv, white_hsv_low, white_hsv_high)
    res = cv2.bitwise_and(frm,frm, mask = mask)
    binary_res = res[:,:,2]
    rr,binary_res = cv2.threshold(binary_res,180,255,cv2.THRESH_BINARY)
    return binary_res
```

Εικόνα 6.6: Συνάρτηση Apply Color Mask

Η πίστα-δρόμος που χρησιμοποιείται στην παρούσα πτυχιακή, έχει λευκές λωρίδες για την οριοθέτηση του χώρου που κινείται το όχημα. Επομένως με αυτή τη συνάρτηση το μετασχηματισμένο σε hsv frame μέσω μιας μάσκας ελάχιστου και μέγιστου λευκού κατωφλίου(threshold) διαβάζει μόνο το λευκό σε αυτά τα όρια, δηλαδή τις λωρίδες. Στη συνέχεια το αποτέλεσμα μετατρέπεται σε δυαδική εικόνα.

Εικόνα 6.7: Δυαδική Εικόνα

def region_of_interest

```
def region_of_interest(frame, region):
    mask = np.zeros_like(frame)
    cv2.fillPoly(mask, region, 255)
    return cv2.bitwise_and(frame, mask)
```

Εικόνα 6.8: Συνάρτηση Region of Interest

Λαμβάνει το frame σε δυαδική (binary) μορφή και επιστρέφει πίσω μόνο την περιοχή που βρίσκεται το ενδιαφέρον μας, δηλαδή στις λωρίδες. Η περιοχή αυτή έχει οριστεί προηγουμένως δίνοντας τα σημεία του frame στη μεταβλητή region. Το αποτέλεσμα που θα επιστραφεί θα είναι η περιοχή του ενδιαφέροντος μας και το υπόλοιπο frame θα είναι μαύρο.

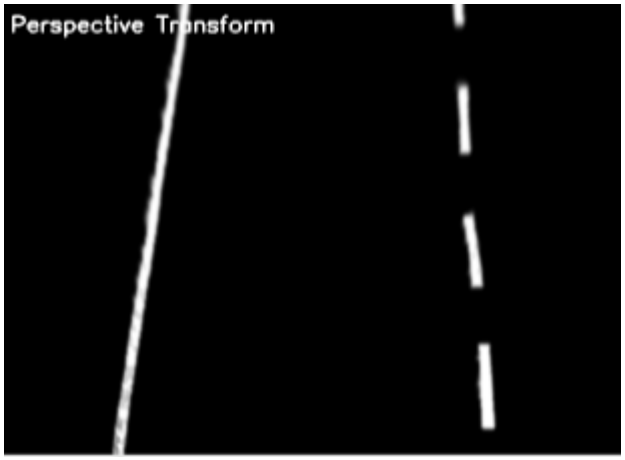
Εικόνα 6.9: Δυαδικό RoI

def four_point_transform

```
def four_point_transform(points):
    maxWidth, maxHeight = frame.shape[1],frame.shape[0]

    dst = np.float32([ [0,0] ,[0,maxHeight] ,[maxWidth, maxHeight] ,[maxWidth, 0] ])

    M = cv2.getPerspectiveTransform(points, dst)
    M_inv = cv2.getPerspectiveTransform(dst,points)
    return M,M_inv
```

Εικόνα 6.10: Συνάρτηση Four Point Transform

Ο μετασχηματισμός της προοπτικής (perspective transform) ορίζεται από τέσσερα σημεία και μας δίνει τη δυνατότητα να δούμε το δρόμο σαν να ήταν ο παρατηρητής ένα πουλί. Πρόκειται για μια μέθοδο, η οποία χρησιμοποιείται συχνά στην κατασκευή σχεδίων, κατόψεων και χαρτών, γεγονός που καθιστά ευκολότερη την περαιτέρω επεξεργασία καθώς κάθε περιττή πληροφορία, σχετικά με το φόντο, αφαιρείται από την εικόνα.



Εικόνα 6.11: Perspective View

def get_lane_base

```
def get_lane_base(frame):
    histogram = np.mean(frame[frame.shape[0]//2:,:], axis=0) + 0.0000001
    indexes = find_peaks_cwt(histogram,[100],max_distances=[800])
    return [(indexes[0], frame.shape[0]), (indexes[-1], frame.shape[0])]
```

Εικόνα 6.12: Συνάρτηση Get Lane Base

Δέχεται για είσοδο το frame σε δυαδική μορφή και μετά από το μετασχηματισμό προοπτικής, επιστρέφει τις συντεταγμένες της βάσης των δύο λευκών λωρίδων.

def get_lane_pixels

```
def get_lane_pixels(frm, lane_base):
    """
      Parameters: frm -- binary, perspective transformed image
                  lane base -- coordinates (x,y) of the base of the lane line
    """

    "Find all pixels in lane_base in a 100px window "
    window_size = 100 * 2
    x_base = lane_base[0]

    if(x_base > window_size):
        window_low = x_base - window_size/2
    else:
        window_low = 0

    window_high = x_base + window_size/2
    # Define a region
    window = frm[:, int(window_low):int(window_high)]
    # Find the coordinates of the white pixels in this region
    x, y = np.where(window == 1)
    # Add window low as an offset
    y += np.uint64(window_low)

    "x,y -- indices of all pixels that belong to that lane line"
    return (x, y)
```

Εικόνα 6.13: Συνάρτηση Get Lane Pixels

Η συγκεκριμένη συνάρτηση έχει ως είσοδο το frame σε δυαδική μορφή και σε μετασχηματισμό προοπτικής όπως επίσης και τις συντεταγμένες της βάσης της λωρίδας. Δημιουργεί ένα παράθυρο 100 εικονοστοιχείων(pixels) και βρίσκει όλα τα pixels στη λωρίδα που έχει δοθεί ανάλογα τη βάση, δηλαδή αριστερά ή δεξιά.

def draw_lane_lines

```
def draw_lane_lines(frame, left_pixels, right_pixels, left_base, right_base):

    frame = np.zeros_like(frame)
    xm_per_pix = 2.3/4800 # meters per pixel in x dimension
    frame_center = (frame.shape[1]/2, frame.shape[0])

    if right_pixels is None:
        line1 = get_curved_lane_line(left_pixels)
        line1_pts = draw_curved_line(frame, line1)
        vehicle_center_distance = 0
    elif left_pixels is None:
        line2 = get_curved_lane_line(right_pixels)
        line2_pts = draw_curved_line(frame, line2)
        vehicle_center_distance = 0
    else:
        line1 = get_curved_lane_line(left_pixels)
        line1_pts = draw_curved_line(frame, line1)
        line2 = get_curved_lane_line(right_pixels)
        line2_pts = draw_curved_line(frame, line2)
        top_points = [line1_pts[-1], line2_pts[-1]]
        base_points = [line1_pts[0], line2_pts[0]]

        vehicle_middle_pixel = int((left_base[0] + right_base[0])/2)
        vehicle_center_distance = float("{0:.3f}".format((vehicle_middle_pixel - frame_center[0]) * xm_per_pix))
        # Fill in the detected lane
        cv2.fillPoly(frame, [np.concatenate((line2_pts, line1_pts, top_points,base_points ))], color=(0,255,0))

    return (frame,vehicle_center_distance)
```

Εικόνα 6.14: Συνάρτηση Draw Lane Lines

Δέχεται για είσοδό της το frame, τα pixels της αριστερής βάσης και λωρίδας και τα αντίστοιχα της δεξιάς πλευράς. Με την κλήση της συνάρτησης draw_curved_line, οι λωρίδες σχεδιάζονται με μπλε χρώμα κι αναλόγως των συντεταγμένων τους, κρίνεται εάν θα χρωματιστούν κι οι δύο ή μόνο μία εξ αυτών. Εφόσον υπάρχουν δύο λωρίδες, η ενδιάμεση περιοχή χρωματίζεται με πράσινο. Στην έξοδό της η συνάρτηση επιστρέφει τις μπλε λωρίδες και την πράσινη περιοχή σε ένα κενό frame, ίδιων διαστάσεων με το αρχικό, όπως επίσης και την απόσταση του οχήματος από το κέντρο της πίστας, δηλαδή το κέντρο της πράσινης περιοχής.

def draw_curved_line

```
def draw_curved_line(frame, line):
    "Parameter:line -- polynomial coefficients representing this line"

    p = np.poly1d(line)
    x = list(range(0, frame.shape[0]))
    y = list(map(int, p(x)))
    pts = np.array([[_y,_x] for _x, _y in zip(x, y)])

    pts = pts.reshape((-1,1,2))

    cv2.polylines(frame, np.int32([pts]), False, color=(255,0,0), thickness=30)
    return pts
```

Εικόνα 6.15: Συνάρτηση Draw Curved Line

Έχει ως είσοδο ένα frame με μηδενικά ίδιων διαστάσεων με το αρχικό, δηλαδή μαύρο. Πάνω σε αυτό θα σχεδιαστεί με μπλε χρώμα μία λωρίδα στο κενό frame έχοντας λάβει υπόψη τη θέση της λευκής. Αυτό επιτυγχάνεται ύστερα από τον υπολογισμό ενός πολυωνύμου δευτέρου βαθμού, εφόσον έχουν δοθεί οι σταθερές του στη μεταβλητή line.

def single_line_drive

```
def single_line_drive(lane_base,left_pixels,right_pixels,line_frame,frame,max_slope):
    if left_pixels[0].any() and lane_base[0][0] < 320:#drive right
        if max_slope < 0:
            right_pixels = None
            curved_lane,dist = draw_lane_lines(line_frame, left_pixels, right_pixels, left_base, right_base)
            curved_lane_frame = np.array(cv2.warpPerspective(curved_lane,M_inv, (frame.shape[1],
        frame.shape[0]),flags=cv2.INTER_LINEAR))
            weighted_frame = cv2.addWeighted(frame,1.0,curved_lane_frame,0.7,0.0)
            cf = 0

            if  max_slope > -0.62:
                motor.backward_left(speed)
                print('back_left')
                sleep(0.6)
                motor.stop()
            motor.forward_right(speed)
            print('bfor_right')
            sleep(0.5)
            motor.stop()
        elif max_slope > 0:
            if max_slope > 0.62:
                motor.backward_right(speed)
                print('back_right')
                sleep(0.6)
                motor.stop()
            motor.forward_left(speed)
            print('bfor_left')
            sleep(0.5)
            motor.stop()
    elif right_pixels[0].any() and lane_base[1][0] > 320:#drive left
        if max_slope > 0:
            left_pixels  = None
            curved_lane,dist = draw_lane_lines(line_frame, left_pixels, right_pixels, left_base, right_base)
            curved_lane_frame = np.array(cv2.warpPerspective(curved_lane,M_inv, (frame.shape[1],
        frame.shape[0]),flags=cv2.INTER_LINEAR))
            weighted_frame = cv2.addWeighted(frame,1.0,curved_lane_frame,0.7,0.0)
            cf = 0
            if max_slope > 0.62:
                motor.backward_right(speed)
                print('back_right')
                sleep(0.6)
                motor.stop()
            motor.forward_left(speed)
            print('bfor_left')
            sleep(0.5)
            motor.stop()
        elif max_slope < 0:
            if  max_slope > -0.62:
                motor.backward_left(speed)
                print('back_left')
                sleep(0.6)
                motor.stop()
            motor.forward_right(speed)
            print('bfor_right')
            sleep(0.5)
            motor.stop()
```

Εικόνα 6.16: Single Line Drive

Η συνάρτηση **single_line_drive** καλείται όταν το όχημα “βλέπει” στο frame μία μόνο λωρίδα και πρέπει να αποφασίσει προς ποια κατεύθυνση θα κινηθεί. Δέχεται για είσοδο τις βάσεις των λωρίδων, το κενό frame, στο οποίο θα σχεδιαστούν οι νέες αναγνωρισμένες λωρίδες, και τη μέγιστη κλίση, που έχει υπολογιστεί έπειτα από το μετασχηματισμό Hough. Στη συνέχεια πραγματοποιείται έλεγχος με μία συνθήκη if εάν τα εικονοστοιχεία, που διαβάστηκαν, ανήκουν στην αριστερή λωρίδα ή στη δεξιά. Αν το πρώτο εικονοστοιχείο της βάσης έχει τιμή μικρότερη του 320 αντιστοιχεί στην αριστερή λωρίδα. Εφόσον αυτό ισχύει, τότε με μία ακόμη συνθήκη ελέγχου if ελέγχει τη μέγιστη κλίση εάν είναι αρνητική και σε αληθή περίπτωση με την κλήση της συνάρτησης **draw_lane_lines** σχεδιάζει πάνω στο κενό frame (line_frame), με τις συντεταγμένες της λευκής λωρίδας, μία μπλε. Με αντίστροφο μετασχηματισμό προοπτικής η **weighted_frame** θα έχει περιεχόμενο το κενό frame με τη σχεδιασμένη μπλε λωρίδα είτε είναι ευθεία, είτε καμπύλη.

Μία δεύτερη εμφωλευμένη if ελέγχει εάν η μέγιστη κλίση (max_slope) της λωρίδας είναι μεγαλύτερη του -0.62, τιμή, η οποία προέκυψε μετά από αρκετές δοκιμές. Σε αυτήν την περίπτωση η κλίση είναι μεγάλη, με αποτέλεσμα το όχημα να χρειαστεί να κινηθεί πρώτα πίσω και αριστερά για 0.6 δευτερόλεπτα. Το επόμενο βήμα θα είναι το όχημα να κινηθεί ευθεία και δεξιά για μισό δευτερόλεπτο. Στην περίπτωση που η μέγιστη κλίση είναι θετική θα πρέπει να κινηθεί ευθεία και αριστερά, δηλαδή θα ισχύσει η περίπτωση κατεύθυνσης του οχήματος που περιγράφεται στο επόμενο κομμάτι κώδικα.

Αντίστοιχα, εάν η τιμή του πρώτου εικονοστοιχείου της βάσης είναι μεγαλύτερη του 320, τότε ισχύει θα ισχύσουν οι ίδιες συνθήκες με την αριστερή λωρίδα, αυτή τη φορά όμως για τη δεξιά έχοντας μέγιστη κλίση θετική. Το όχημα θα κινηθεί πίσω και δεξιά για 0.6 δευτερόλεπτα, εάν η μέγιστη κλίση είναι μεγαλύτερη του ορίου 0.62, και στη συνέχεια ευθεία και αριστερά, ενώ εάν η μέγιστη κλίση είναι αρνητική το όχημα θα πρέπει να κινηθεί δεξιά. Και στις δύο περιπτώσεις παρατηρείται ο μηδενισμός του μετρητή cf, του οποίου η χρήση θα εξηγηθεί παρακάτω.

def show_frame

```
def show_frame(weighted_frame,roi,b_view):
    #weighted_frame = cv2.resize(weighted_frame, (0,0),fx = 1.3,fy = 1.3)
    cv2.imshow('Frame',weighted_frame)
    cv2.moveWindow('Frame',20,20)

    cv2.putText(roi, "Region of Interest", (20, 40), cv2.FONT_HERSHEY_SIMPLEX, 0.75, (255,255,255),2)
    roi = cv2.resize(roi, (0,0),fx = 0.5,fy = 0.5)
    cv2.putText(b_view, "Perspective Transform", (20, 40), cv2.FONT_HERSHEY_SIMPLEX, 0.75, (255,255,255),2)
    b_view = cv2.resize(b_view, (0,0),fx = 0.5,fy = 0.5)

    comb = np.concatenate((roi,b_view), axis=0)
    cv2.imshow('Thresholds',comb)
    cv2.moveWindow('Thresholds',weighted_frame.shape[1]+65,20)
    return
```

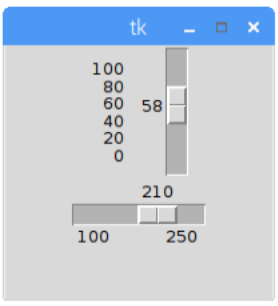
Εικόνα 6.17: Συνάρτηση Show Frame

Με την κλήση της show_frame γίνεται η εμφάνιση του frame ύστερα από όλα τα στάδια επεξεργασίας, δείχνοντας το τελικό αποτέλεσμα παράλληλα με την εμφάνιση δύο ακόμη παραθύρων με τα δυαδικά frames της περιοχής ενδιαφέροντος και του μετασχηματισμού προοπτικής.

Χειροκίνητη ρύθμιση με μπάρα κύλισης των white_low και speed (ελάχιστη τιμή λευκού και ταχύτητας) με χρήση της βιβλιοθήκης γραφικών tkinter

```
#scrollbar_settings
mw = tk.Tk()
w = 200
h = 200
x = 20
y = 520
mw.geometry('%dx%d+%d+%d' % (w, h, x, y))
mw.resizable(0, 0)
back = tk.Frame(master=mw)
back.pack_propagate(0) #Δεν επηρεάζονται οι διαστάσεις του frame από το περιεχόμενό του
back.pack(fill=tk.BOTH, expand=1) #Διευρύνει το frame για να γεμίσει το αρχικό παράθυρο

speed_ = Scale(master=back, from_=100, to=0, tickinterval=20)
speed_.set(58)
speed_.pack()
white_low = Scale(master=back, from_=100, to=255,tickinterval=150, orient=HORIZONTAL)
white_low.set(210)
white_low.pack()
```

Εικόνα 6.18: Ρύθμιση White Low και Ταχύτητας



Εικόνα 6.19: Παράθυρο Ρύθμισης

Το παραπάνω κομμάτι κώδικα δημιουργεί ένα παράθυρο 200x200 εικονοστοιχείων και τοποθετείται στη θέση (20,520) του frame. Χρησιμοποιεί δύο μπάρες κύλισης για τη χειροκίνητη ρύθμιση των τιμών της ταχύτητας των κινητήρων και της τιμής του κατώτερου λευκού κατωφλίου της μάσκας χρώματος. Οι τιμές που έχουν οριστεί

αρχικά είναι 58, για την ταχύτητα (κάθετη μπάρα), και 210 για το ελάχιστο λευκό (οριζόντια μπάρα).

```
#start_frame_reading
vs = Threaded_Frame(0).start()
time.sleep(0.5)

#initialize_motors
motor = Motor(27,22,19,13,25,24)
cf = 0
```

Εικόνα 6.20: Κλήση Κλάσεων

Στο κυρίως πρόγραμμα, προτού ξεκινήσει ο βρόχος επανάληψης while καλούνται οι δύο κλάσεις του προγράμματος, **Threaded_Frame** και **Motor**. Με την **Threaded_Frame** ξεκινάει το διάβασμα των frames έτσι ώστε όταν χρειαστεί μέσα στην while, να διαβαστεί ταχύτερα το απαιτούμενο frame. Με την κλήση της Motor γίνεται η ρύθμιση των παραμέτρων του εμπρόσθιου και του πίσω κινητήρα δίνοντας τους αριθμούς των ακίδων που είναι συνδεδεμένοι. Στη συνέχεια αρχικοποιείται ο μετρητής cf, όπου θα χρειαστεί κατά την πλοήγηση του οχήματος, σε περίπτωση ελάττωσης της ταχύτητας του πίσω κινητήρα.

Βρόχος επανάληψης while

```
while(True):
    back.update()

    "Read_frame"
    frame = vs.read()
    if frame == None:
        print("Error reading camera source...")
        print("Unplug and plug again the usb camera!")
        break

    frame_size = np.shape(frame)
```

Εικόνα 6.21: Συνθήκη While

Ξεκινώντας το βρόχο επανάληψης ορίζεται η τιμή του στην παρένθεση να είναι True, δηλαδή όσο πραγματοποιούνται τα περιεχόμενά του να εκτελείται διαρκώς. Αρχικά ανανεώνεται το παράθυρο γραφικών με τις μπάρες κύλισης κι έπειτα γίνεται κλήση της συνάρτησης read() μέσω της κλάσης **Threaded_Frame**, ώστε να διαβαστεί το frame που υπάρχει ήδη σε αυτή. Εάν δεν έχει διαβαστεί το frame, τότε αυτό οφείλεται κυρίως σε πρόβλημα ανάγνωσης της κάμερας από τη θύρα USB του Raspberry, επομένως τερματίζεται η επανάληψη με μήνυμα σφάλματος κι ενημέρωση του χρήστη να αποσυνδέσει και να επανασυνδέσει την κάμερα στη θύρα USB. Εφόσον διαβαστεί επιτυχώς το frame, οι διαστάσεις του αποθηκεύονται στη μεταβλητή **frame_size**.

```
"Apply_Sobel_filters_to_LS_channels"
frame_HLS = cv2.cvtColor(frame,cv2.COLOR_BGR2HLS)
wraped_LS = apply_sobel(frame_HLS)

"Apply_color_mask"
frame_HSV = cv2.cvtColor(frame,cv2.COLOR_BGR2HSV)
res = apply_color_mask(frame_HSV,frame)

combined_binary = cv2.bitwise_or(wraped_LS,res)
combined_binary = gaussian_blur(combined_binary)
```

Εικόνα 6.22: Φίλτρο Sobel και Μάσκα Χρώματος

Έπειτα, εφαρμόζονται φίλτρα Sobel στο frame με την κλήση της συνάρτησης **apply_sobel** και μάσκα χρώματος με την κλήση της **apply_color_mask**. Για την εφαρμογή μάσκας χρώματος, ο χώρος HSV είναι καταλληλότερος για τον εντοπισμό των χρωμάτων, καθώς διαχωρίζει τα χρώματα στην απόχρωση (Hue), στο μέγεθος του χρώματος (Κορεσμός-Saturation) και την τιμή φωτεινότητας (Value). Τα αποτελέσματα των δύο συναρτήσεων θα συνδυαστούν στο ίδιο frame με τη συνάρτηση της OpenCV **bitwise_or** όπου εκτελείται η λογική πράξη or. Το τελικό frame θα είναι δυαδικής μορφής κι εν συνεχεία θα αφαιρεθούν οι λεπτομέρειες με θόλωμα μέσω της **gaussian_blur**.

```
"Region_of_interest"
region = [np.array([(0,480),(0,400),(100,250),(540,250),(640,400),(640,480)])]
roi = region_of_interest(combined_binary, region)

"Birds_View"
corners = np.float32([(100,250),(0,480),(640,480),(540,250)])
M,M_inv = four_point_transform(corners)
b_view = np.array(cv2.warpPerspective(roi,M, (frame.shape[1], frame.shape[0]),flags=cv2.INTER_LINEAR))
b_view = cv2.dilate(b_view,None,iterations=2)
b_view = cv2.erode(b_view,None,iterations=2)

line_frame = np.zeros_like(frame)
lane_base = get_lane_base(b_view) #first value empty -> [] (IndexError)
left_base, right_base = lane_base
weighted_frame = frame

speed = speed_.get()
```

Εικόνα 6.23: Περιοχή Ενδιαφέροντος και Bird's Eye View

Για να αφαιρεθούν τα σημεία που δε χρειάζονται στο frame και να εστιαστεί η περιοχή του δρόμου-πίστας και των λωρίδων, ορίζουμε μια περιοχή ενδιαφέροντος, όπου σύμφωνα με τις επιθυμητές συντεταγμένες η συνάρτηση **region_of_interest** την επιστρέφει προσθέτοντας μαύρο χρώμα στα υπόλοιπα σημεία που δεν ανήκουν σε αυτή. Έπειτα ορίζοντας τέσσερα νέα σημεία, το frame θα δεχτεί μετασχηματισμό τεσσάρων σημείων(**four_point_transform**) και μετασχηματισμό προοπτικής (**wrapPerspective**). Το αποτέλεσμα θα είναι ένα δυαδικό frame που θα εμφανίζει μόνο την περιοχή του δρόμου και τις δύο λευκές λωρίδες χωρίς τις περιττές λεπτομέρειες.

Σύμφωνα με τη σελίδα της OpenCV, κάνοντας μορφολογική επεξεργασία σε μια εικόνα με τις δύο βασικές μορφολογικές διαδικασίες τις dilate και erode (διαστολή και διάβρωση) επιτυγχάνεται :

- Αφαίρεση θορύβου
- Απομόνωση μεμονωμένων στοιχείων και ένωση διαφόρων στοιχείων σε μια εικόνα
- Εύρεση ανωμαλιών έντασης ή οπών σε μια εικόνα, επομένως θα έχουμε καλύτερο αποτέλεσμα στο frame όσων αφορά την εξάλειψη θορύβου.

Συνεχίζοντας, ορίζεται ένα καινούριο frame γεμάτο μηδενικά (μαύρο), ίδιων διαστάσεων με το αρχικό, στο οποίο θα σχεδιαστούν οι ανιχνευμένες λωρίδες με νέο χρώμα. Η κλήση της συνάρτησης **get_lane_base** θα γίνει με παράμετρο το frame από τον μετασχηματισμό προοπτικής και θα επιστραφούν οι συντεταγμένες

των βάσεων των δύο λωρίδων στις μεταβλητές **left_base** για την αριστερή και **right_base** για τη δεξιά.

Στη μεταβλητή **weighted_frame** δέχεται ως τιμή το αρχικό frame, έτσι ώστε αν δεν είναι επιτυχής η εύρεση των λωρίδων να εμφανιστεί το αρχικό.

Η τιμή της ταχύτητας που θα κινηθούν οι κινητήρες καθορίζεται από τη μεταβλητή speed, η οποία αλλάζει με βάση τη μπάρα κύλισης εάν χρειαστεί.

```
lines = cv2.HoughLinesP(roi, rho=2, theta=.02, threshold=9, minLineLength=80, maxLineGap=70)
slopes = []
max_slope = 0
if lines is not None:
    for line in lines:
        #format line to be drawn
        x1, y1, x2, y2 = line[0]
        if x2==x1:
            continue # ignore a vertical line
        slope = (y2-y1)/(x2-x1)
        slopes.append(slope)
    max_slope = max(slopes)
```

Εικόνα 6.24: Μετασχηματισμός Hough

Με τη χρήση του μετασχηματισμού Hough (cv2.HoughLinesP) επιτυγχάνεται η εύρεση ευθείων γραμμών στο δυαδικό frame. Οι παράμετροι που είναι αναγκαίο να ρυθμιστούν είναι οι εξής:

- rho: απόσταση ανάλυσης συσσωρευτή σε εικονοστοιχεία
- theta: ανάλυση γωνίας του συσσωρευτή σε ακτίνια
- threshold: παράμετρος κατωφλίου συσσωρευτή. Επιστρέφονται μόνο οι γραμμές με περισσότερες ψήφους από το κατώτατο όριο.
- minLineLength: ελάχιστο μήκος για να θεωρηθεί γραμμή ένα τμήμα σημείων
- maxLineGap: μέγιστο επιτρεπόμενο κενό μεταξύ σημείων στην ίδια γραμμή για τη σύνδεση τους.

Η τιμή που έχει οριστεί για κάθε παράμετρο προέκυψε ύστερα από αρκετές δοκιμές με την κάμερα και σε διαφορετικά επίπεδα φωτισμού.

Εφόσον εντοπιστούν οι γραμμές, με έναν βρόχο επανάληψης for γίνεται ο υπολογισμός της κλίσης κάθε ευθείας και αφού ολοκληρωθεί αποθηκεύεται στη μεταβλητή max_slope η μέγιστη κλίση. Ο υπολογισμός αυτός θα χρειαστεί στο αμέσως επόμενο κομμάτι κώδικα για την πλοήγηση του εμπρόσθιου κινητήρα.

```
if left_base[0] == right_base[0]:
    left_pixels  = get_lane_pixels(b_view, left_base)
    right_pixels = get_lane_pixels(b_view, right_base)
    single_line_drive(lane_base,left_pixels,right_pixels,line_frame,frame,max_slope)
```

Εικόνα 6.25: Πλοήγηση Εμπρόσθιου Κινητήρα

Η παραπάνω συνθήκη if ελέγχει εάν το προηγούμενο αποτέλεσμα της **get_lane_base** συμπίπτει με τις δύο βάσεις. Εάν οι τιμές τους είναι ίσες, τότε με τη συνάρτηση get_lane_pixels επιστρέφονται τα pixels των λωρίδων. Στη συνέχεια καλείται η συνάρτηση **single_line_drive** όπου θα γίνει αναγνώριση σε ποια από τις δύο λωρίδες ανήκουν τα εικονοστοιχεία που διαβάστηκαν ώστε να πλοηγηθεί προς τη σωστή κατεύθυνση το όχημα.

```
    else:
        left_pixels  = get_lane_pixels(b_view, left_base)
        right_pixels = get_lane_pixels(b_view, right_base)
        if (left_pixels[0].any() and right_pixels[0].any()):
            #Curved Lane from Birds View
            curved_lane,dist = draw_lane_lines(line_frame, left_pixels, right_pixels, left_base, right_base)
            #Curved Lane inverted to Roi
            curved_lane_frame = np.array(cv2.warpPerspective(curved_lane,M_inv, (frame.shape[1],
frame.shape[0]),flags=cv2.INTER_LINEAR))
            weighted_frame = cv2.addWeighted(frame,1.0,curved_lane_frame,0.7,0.0)
            #White info Bar
            info = np.zeros_like(weighted_frame)
            cv2.rectangle(info, (10, 10), (350, 60), (180,180,180), -1)
            weighted_frame = cv2.addWeighted(weighted_frame,1.0,info,0.55,0.0)
            cv2.putText(weighted_frame, "Distance from center: " + str(dist), (20, 40),
cv2.FONT_HERSHEY_SIMPLEX, 0.65, (0,0,0),2)
            show_frame(weighted_frame,roi,b_view)

            if dist < -0.028:
                cf = 0
                motor.forward_left(speed)
                print('wfor_left')
                sleep(0.5)
                motor.stop()
            elif dist > 0.028:
                cf = 0
                motor.forward_right(speed)
                print('wfor_right')
                sleep(0.5)
                motor.stop()
            else:
                cf = cf +1
                motor.forward(speed-10)
                print('wforward')
                sleep(0.3)
                motor.stop()
                if cf == 4:
                    motor.backward(speed)
                    sleep(0.1)
                    motor.stop()
                    cf = 0
                    print('freno')
```

Εικόνα 6.26: Περίπτωση Δύο Λωρίδων

Στην περίπτωση που οι δύο βάσεις δεν συμπίπτουν, δηλαδή υπάρχουν δύο λωρίδες, τότε ισχύει η παραπάνω εξωτερική else κατά την οποία αρχικά διαχωρίζονται οι αριστερές από τις δεξιές βάσεις. Με την εμφωλευμένη if γίνεται έλεγχος ώστε να υπάρχουν ταυτόχρονα συντεταγμένες και για τις δύο λωρίδες κι εν συνεχεία ακολουθείται η ίδια διαδικασία που περιγράφηκε προηγουμένως. Η **draw_lane_lines** σχεδιάζει στο κενό frame τις δύο μπλε λωρίδες στις αντίστοιχες

συντεταγμένες με τις λευκές, μόνο που αυτή τη φορά χρωματίζει την ενδιάμεση περιοχή τους με πράσινο και προσθέτει το αποτέλεσμα με αντεστραμμένη προοπτική στο αρχικό frame με χρήση βάρους τιμής 0.7. Επίσης προστίθεται στο frame ένα λευκό πινακάκι, όπου εμφανίζει την απόκλιση του οχήματος από το κέντρο της πράσινης περιοχής, στην οποία κινείται κι εμφανίζεται το τελικό αποτέλεσμα με την **show_frame**.

Τέλος, γίνεται έλεγχος της απόκλισης του οχήματος από το κέντρο με την τιμή που ορίστηκε στο 0.028 ύστερα από αρκετές δοκιμές στην πίστα. Αν η απόσταση του οχήματος είναι μικρότερη του -0.028 τότε θα πρέπει να κινηθεί ευθεία και αριστερά για μισό δευτερόλεπτο, αν είναι μεγαλύτερη του 0.028 θα πρέπει να κινηθεί ευθεία και δεξιά για μισό δευτερόλεπτο, αλλιώς η κίνησή του θα είναι ευθεία με μειωμένη ταχύτητα για τρία δέκατα του δευτερολέπτου. Στην τελευταία περίπτωση αν το όχημα κινείται για πολύ ώρα ευθεία θα έχει αυξημένη ταχύτητα και φόρα. Επομένως με μία ακόμα συνθήκη if, γίνεται έλεγχος μέσω του μετρητή cf, κι εφόσον έχει δεχθεί τέσσερεις φορές εντολή να προχωρήσει ευθεία, ελαττώνει την ταχύτητα των πίσω κινητήρων θέτοντας τους να κινηθούν αντίστροφα για ένα δέκατο του δευτερολέπτου.

```
elif (left_pixels[0].any()):
    single_line_drive(lane_base,left_pixels,right_pixels,line_frame,frame,max_slope)
elif (right_pixels[0].any()):
    single_line_drive(lane_base,left_pixels,right_pixels,line_frame,frame,max_slope)
```

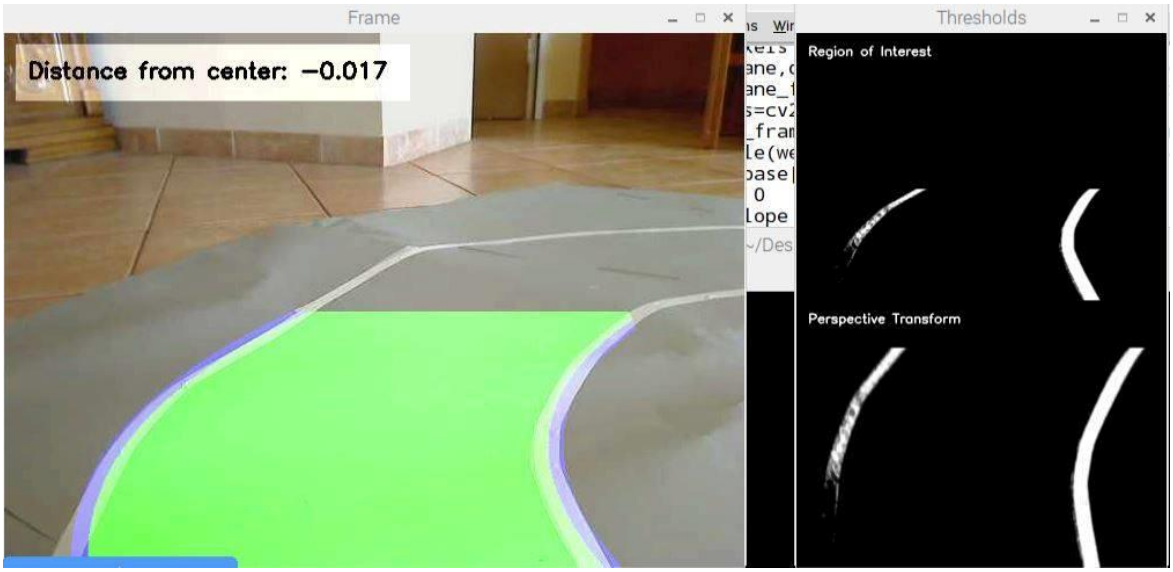
Εικόνα 6.27: Έλεγχος Ύπαρξης Εικονοστοιχείων

Οι δύο τελευταίες συνθήκες ελέγχουν αν υπάρχουν εικονοστοιχεία που δεν έχουν ανιχνευτεί προηγουμένως είτε για την αριστερή είτε για τη δεξιά λωρίδα και εφόσον δεν υπάρχουν κενά εικονοστοιχεία γίνεται η κλήση της συνάρτησης single_line_drive για τη σωστή πλοήγηση του οχήματος.

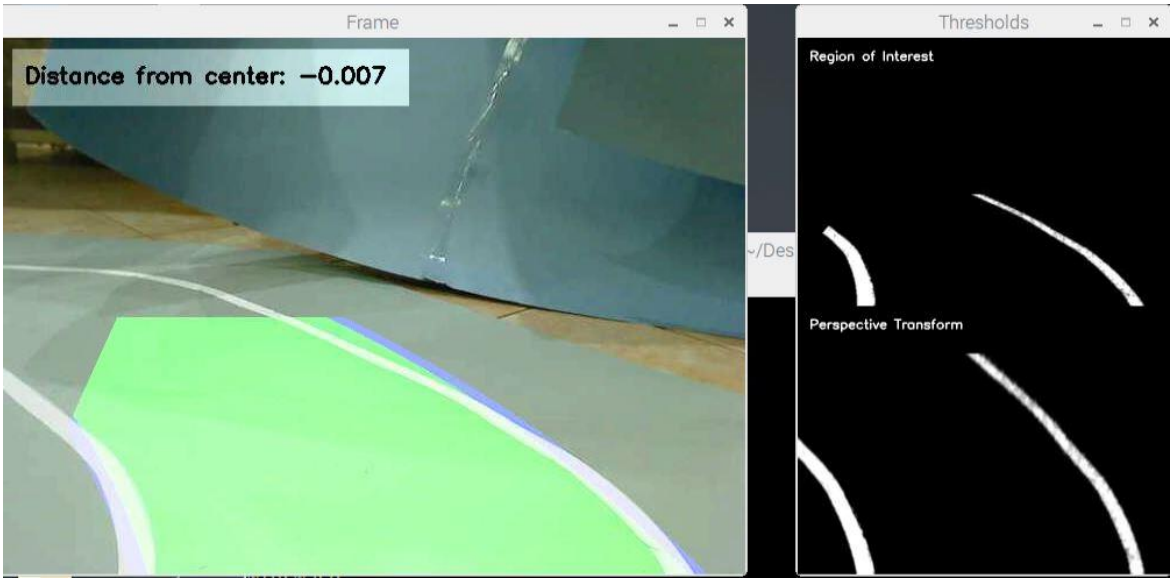
```
        show_frame(weighted_frame,roi,b_view)
        motor.stop()

        if cv2.waitKey(1) & 0xFF == ord('q'):
            break

# capture release
vs.stop()
cv2.destroyAllWindows()
GPIO.cleanup()
```

Εικόνα 6.25: Τερματισμός Προγράμματος

Με την κλήση της συνάρτησης **show_frame** γίνεται η εμφάνιση του τελικού frame έπειτα από την επεξεργασία των λωρίδων, ειδάλλως εμφανίζεται το αρχικό. Παράλληλα εμφανίζονται δύο παράθυρα με την περιοχή ενδιαφέροντος και του μετασχηματισμού προοπτικής.



Εικόνα 6.28: Δεξιά στροφή



Εικόνα 6.29: Αριστερή Στροφή

Μετά την εμφάνιση των αποτελεσμάτων, καλείται η motor.stop() για τυχόν αστοχία του αλγορίθμου ώστε να τεθούν οι κινητήρες σε αδρανή κατάσταση και γίνεται έλεγχος αν έχει δοθεί από τον χρήστη το γράμμα q από το πληκτρολόγιο.

Εφόσον ο χρήστης έχει πατήσει το γράμμα q πάνω σε κάποιο από τα εμφανιζόμενα παράθυρα τότε ο βρόχος επανάληψης θα διακοπεί και θα εκτελεστούν οι εντολές:

- vs.stop(): για παύση διαβάσματος των frames από τη συνάρτηση Threaded_Frame
- cv2.destroyAllWindows(): θα κλείσουν όλα τα ανοικτά παράθυρα
- GPIO.cleanup(): θα γίνει καθαρισμός του GPIO τερματίζοντας το πρόγραμμα.

Εάν ο χρήστης δεν πληκτρολογήσει το γράμμα q ή δεν διακόψει το πρόγραμμα με διαδικασία άμεσου τερματισμού από το τερματικό, τότε η επανάληψη θα συνεχιστεί διαβάζοντας το επόμενο frame από τη συνάρτηση **Threaded_Frame**.

(Κενό φύλλο)

ΣΥΝΟΨΗ

Σε αυτό το κεφάλαιο πραγματοποιείται η σύνοψη της πτυχιακής εργασίας, παρουσιάζοντας πιθανές βελτιώσεις και προοπτικές εξέλιξης του αυτόνομου οχήματος, καθώς επίσης τα πορίσματα από την εκπόνησή της.

7.1 Παρατηρήσεις – Συμπεράσματα

Το μεγαλύτερο ζήτημα ήταν οι ξαφνικές μεταβολές των φωτεινών συνθηκών. Σε αυτές τις περιπτώσεις, οι γραμμές έχουν είτε χαθεί εντελώς (πηγαίνοντας από φωτεινό σε σκοτεινό) είτε η εικόνα γεμίζει με θόρυβο, που προέρχεται από τα λευκά σημεία. Πρέπει να εξεταστούν πιο προηγμένες τεχνικές φιλτραρίσματος και ισοστάθμισης φωτεινότητας. Αυτό ήταν ένα πολύ απαιτητικό έργο που αφορούσε τον συντονισμό πολλών παραμέτρων εφόσον η όλη διαδικασία γίνεται με την επεξεργασία της εικόνας σε πραγματικό χρόνο. Οι τεχνικές ηλεκτρονικής όρασης είναι ευαίσθητες στις επιλεγμένες παραμέτρους και εάν οι παράμετροι δεν επιλεγούν σωστά, αποτυγχάνουν.

Αρχικά χρησιμοποιείτο η αυθεντική κάμερα του Raspberry Pi αποδείχθηκε πολύ ευαίσθητη στο θέμα του φωτισμού και δεν αναγνώριζε τις γραμμές τις λωρίδας, ενώ το φως σε όλες τις δοκιμές ήταν υπερεπαρκές.

Η Pi Cam αντικαταστάθηκε με μια Web Camera, η οποία ανταποκρινόταν αξιοσημείωτα καλύτερο στις ίδιες συνθήκες φωτισμού, είχε ευρύτερο πλάνο λήψης και ήταν πιο φθηνή.

Οι υπέρογκες απαιτήσεις του προγράμματος και το σύνολο των συναρτήσεων, που εκτελούνται, προκαλούν μερικό κόλλημα (lag) στη γενικότερη λειτουργία κι απόδοση του Raspberry Pi.

Λόγω του lag, χρησιμοποιήθηκε νήμα ούτως ώστε να μην κολλάνε τόσο τα frames που λαμβάνει η κάμερα.

Το όχημα κινείται με μικρή ταχύτητα για να προλάβουν να διαβαστούν τα frame, να γίνει η αντίστοιχη επεξεργασία εικόνας και να ανταποκριθεί έγκαιρα στις αλλαγές του δρόμου.

Το αντικείμενο της παρούσας πτυχιακής εργασίας είναι ιδιαίτερα ενδιαφέρον, ωστόσο συνιστάται η αγορά κάποιου υπολογιστή με καλύτερη επεξεργαστική ισχύ.

7.2 Πιθανές Βελτιώσεις

Παρακάτω αναφέρονται επιγραμματικά τρόποι με τους οποίους το αυτόνομο όχημα θα είναι πιο αποτελεσματικό:

1. Συλλογή δεδομένων δρόμου από την κάμερα
2. Συλλογή δεδομένων από αισθητήρες απόστασης και φωτεινότητας
3. Εκμάθηση του δρόμου μέσω της συλλογής δεδομένων. Συνεπάγεται ταχύτερος και πιο ακριβής υπολογισμός πλοήγησης σε ευθεία πορεία και ακόμη σημαντικότερο σε στροφές

Βιβλιογραφικές Αναφορές

BENCHOFF, B. 2016. INTRODUCING THE RASPBERRY PI 3. Available: https://hackaday.com/2016/02/28/introducing-the-raspberry-pi-3/ [Accessed 25/9/17].

BLOG, R. P. (n.d). POWER SUPPLY. Available: https://www.raspberrypi.org/documentation/hardware/raspberrypi/power/README.md [Accessed 15/5/17].

BREEMER, G. 2017. Project 4: Advanced Lane Finding. Available: https://github.com/GeoffBreemer/SDC-Term1-P4-Advanced-Lane-Finding [Accessed 16/7/17].

CLAUDIU. 2017. Advanced Lane Finding using OpenCV. Available: http://www.coldvision.io/2017/03/02/advanced-lane-finding-using-opencv/ [Accessed 22/7/17].

CONOR, L. 2015. A History Of The Raspberry Pi. Available: http://novadigitalmedia.com/history-raspberry-pi/ [Accessed 4/9/17].

DAVID, L. 2017. Detect Lanes. Available: https://github.com/udacity/CarND-LaneLines-P1/blob/master/P1.ipynb [Accessed 20/7/17].

GALEN, B. 2017. Building a lane detection system using Python 3 and OpenCV. Available: https://medium.com/@galen.ballew/opencv-lanedetection-419361364fc0 [Accessed 12/6/17].

GALLEN, B. 2017. OpenCV For Lane Detection in Self Driving Cars. Available: https://medium.com/@galen.ballew/opencv-lanedetection-419361364fc0 [Accessed 12/6/17]

GERTRUDE, D. D. 2017. P4: Advanced Lane Lines. Available: https://github.com/Deborah-Digges/SDC-ND-term-1/blob/master/P4-Advanced-Lane-Lines/P4.ipynb [Accessed 10/8/17].

H, S. C. (n.d.). A Byte Of Python

LLC, W. A. W. (n.d.). How to use the L298N Dual H-Bridge Motor Driver. Available: https://www.bananarobotics.com/shop/How-to-use-the-L298N-Dual-H-Bridge-Motor-Driver [Accessed 2/10/17].

MARTINALLO, S. 2017. Advanced Lane Finding Project - Writeup. Available: https://github.com/Nallo/CarND-P4-Advanced-Lane-Lines [Accessed 18/9/17].

NAOKI, S. 2017. Finding Lane Lines on the Road. Available: https://github.com/naokishibuya/car-finding-lane-lines [Accessed 20/7/17].

NICK(NCONDO). 2017. Available: https://github.com/ncondo/Lane-and-Vehicle-Detection/blob/master/lane_tracker.py#L147 [Accessed 5/8/17].

NIKOLIC, M. N. 2017. Advanced Lane Finding. Available: https://medium.com/@ajsmilutin/advanced-lane-finding-5d0be4072514 [Accessed 5/8/17].

PHIL HOWARD , R. (n.d.). Raspberry Pi Pinout. Available: https://pinout.xyz/ [Accessed 22/4/17].

RAPIDTABLES. (n.d.). RapidTables. Available: http://www.rapidtables.com/convert/color/rgb-to-hsv.htm [Accessed 6/7/17].

REALVNC (n.d.). VNC Connect and Raspberry Pi. Available: https://www.realvnc.com/en/connect/docs/raspberry-pi.html [Accessed 6/3/17].

REICHENSTEIN7. 2014. Arduino Modules - L298N Dual H-Bridge Motor Controller. Available: http://www.instructables.com/id/Arduino-Modules-L298N-Dual-H-Bridge-Motor-Controll/ [Accessed 25/5/17].

ROSCOE, W. 2017. Line detection autopilot using Python + OpenCV. Available: https://wroscoe.github.io/compound-eye-autopilot.html [Accessed 20/7/17].

ROSEBROCK, A. 2016a. Install guide: Raspberry Pi 3 + Raspbian Jessie + OpenCV 3. Available: https://www.pyimagesearch.com/2016/04/18/install-guide-raspberry-pi-3-raspbian-jessie-opencv-3/ [Accessed 5/3/17].

ROSEBROCK, A. 2016b. Unifying picamera and cv2.VideoCapture into a single class with OpenCV. Available: https://www.pyimagesearch.com/2016/01/04/unifying-picamera-and-cv2-videocapture-into-a-single-class-with-opencv/ [Accessed 10/9/17].

SINGH, A. K. 2016. 11 Raspberry Pi OS for Everyday Computing – Best of. Available: http://www.hongkiat.com/blog/pi-operating-systems/ [Accessed 13/6/17].

THEMAGPI 2015. RASPBERRY PI 3 IS OUT NOW! SPECS, BENCHMARKS & MORE. Available: https://www.raspberrypi.org/magpi/raspberry-pi-3-specs-benchmarks/ [Accessed 10/9/17].

WIKIPEDIA (n.d.). Autonomous car. *Wikipedia, the free encyclopedia.* Available: https://en.wikipedia.org/wiki/Autonomous_car [Accessed 26/9/17].

YADAV, V. 2017. P4_AdvancedLaneFinding. Available: https://github.com/vxy10/P4_AdvancedLaneFinding/blob/master/main_lane_frame_by_frame_analysis.ipynb [Accessed 22/7/17].

ZUCCOLO, R. 2017. Self-driving Cars—Advanced computer vision with OpenCV, finding lane lines. Available: https://chatbotslife.com/self-driving-cars-advanced-computer-vision-with-opencv-finding-lane-lines-488a411b2c3d [Accessed 20/7/17].

ΠΡΩΤΟΝΟΤΆΡΙΟΣ, Ι. 2011. *Ανάπτυξη Συστήματος Ενσωματωμένων Αισθητήρων για Ασύρματη Μετάδοση Εικόνας και Δεδομένων.* Πανεπιστήμιο Πατρών. Available:http://nemertes.lis.upatras.gr/jspui/bitstream/10889/4394/1/Protonotarios.Ioannis.Thesis.pdf [Accessed 20/7/17].