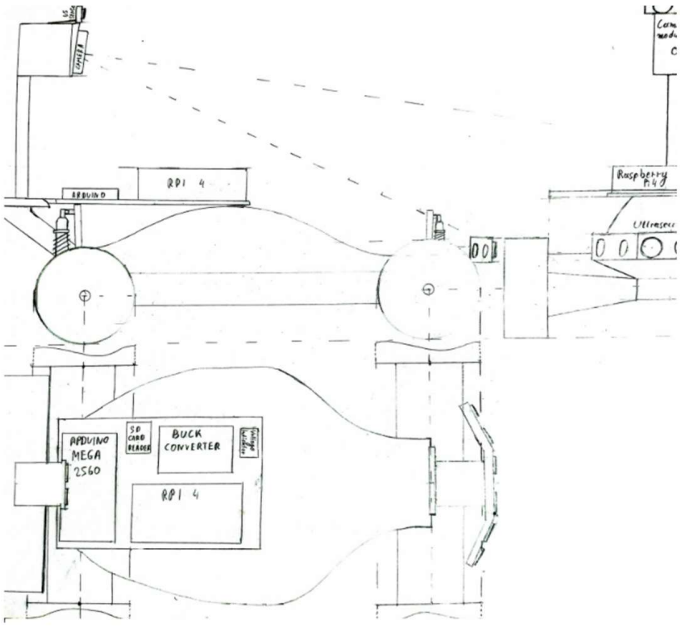
ΠΑΝΕΠΙΣΤΗΜΙΟ ΔΥΤΙΚΗΣ ΑΤΤΙΚΗΣ

ΣΧΟΛΗ ΜΗΧΑΝΙΚΩΝ

ΤΜΗΜΑ ΒΙΟΜΗΧΑΝΙΚΗΣ ΣΧΕΔΙΑΣΗΣ ΚΑΙ ΠΑΡΑΓΩΓΗΣ

ΘΕΜΑ ΔΙΠΛΩΜΑΤΙΚΗΣ ΕΡΓΑΣΙΑΣ

ΚΑΤΑΣΚΕΥΗ ΑΥΤΟΚΙΝΟΥΜΕΝΟΥ ΟΧΗΜΑΤΟΣ ΠΡΟΟΡΙΣΜΕΝΟΥ ΓΙΑ ΟΔΗΓΗΣΗ ΣΕ ΟΔΟΥΣ ΧΩΡΙΣ ΔΙΑΓΡΑΜΜΙΣΗ ΑΞΙΟΠΟΙΩΝΤΑΣ ΛΑΜΒΑΝΟΜΕΝΗ ΕΙΚΟΝΑ ΚΑΙ ΤΕΧΝΗΤΟ ΝΕΥΡΩΝΙΚΟ ΔΙΚΤΥΟ ΓΙΑ ΤΗΝ ΑΝΑΛΥΣΗ ΤΗΣ



ΟΝΟΜΑ ΦΟΙΤΗΤΗ:

ΓΕΩΡΓΙΟΣ ΠΕΤΡΟΥΛΙΑΣ ΜΕΪΜΕΤΗΣ

ΕΠΙΒΛΕΠΩΝ ΚΑΘΗΓΗΤΗΣ:

ΑΒΡΑΑΜ ΧΑΤΖΟΠΟΥΛΟΣ

ΑΙΓΑΛΕΩ, ΦΕΒΡΟΥΑΡΙΟΣ 2021

Η παρούσα διπλωματική εργασία εγκρίθηκε ομόφωνα από την τριμελή εξεταστική επιτροπή, η οποία ορίστηκε από την Γ.Σ. του Τμήματος Μηχανικών Βιομηχανικής Σχεδίασης και Παραγωγής του Πανεπιστημίου Δυτικής Αττικής, σύμφωνα με το νόμο και τον εγκεκριμένο Οδηγό Σπουδών του τμήματος.

Τα μέλη της Επιτροπής ήταν:

- Χατζόπουλος Αβραάμ (Επιβλέπων)
- Παπουτσιδάκης Μιχαήλ (Μέλος)
- Δρόσος Χρήστος (Μέλος)

Avraam Chatzopoulos
Digitally signed by Avraam Chatzopoulos
Date: 2021.03.01 09:49:12 +02'00'

MICHAIL PAPOUTSIDAKIS
Digitally signed by MICHAIL PAPOUTSIDAKIS
Date: 2021.03.05 10:01:09 +02'00'

Christos Drosos
Digitally signed by Christos Drosos
Date: 2021.03.01 10:08:40 +02'00'

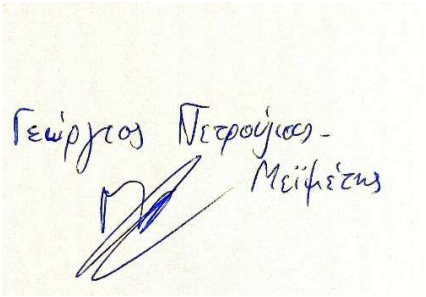
ΔΗΛΩΣΗ ΣΥΓΓΡΑΦΕΑ ΔΙΠΛΩΜΑΤΙΚΗΣ ΕΡΓΑΣΙΑΣ

Ο κάτωθι υπογεγραμμένος Πετρούλιας – Μεϊμέτης Γεώργιος του Νεκταρίου, με αριθμό μητρώου 71444819 φοιτητής του Πανεπιστημίου Δυτικής Αττικής της Σχολής Μηχανικών του Τμήματος Βιομηχανικής Σχεδίασης και Παραγωγής, δηλώνω υπεύθυνα ότι:

«Είμαι συγγραφέας αυτής της διπλωματικής εργασίας και ότι κάθε βοήθεια την οποία είχα για την προετοιμασία της είναι πλήρως αναγνωρισμένη και αναφέρεται στην εργασία. Επίσης, οι όποιες πηγές από τις οποίες έκανα χρήση δεδομένων, ιδεών ή λέξεων, είτε ακριβώς είτε παραφρασμένες, αναφέρονται στο σύνολό τους, με πλήρη αναφορά στους συγγραφείς, τον εκδοτικό οίκο ή το περιοδικό, συμπεριλαμβανομένων και των πηγών που ενδεχομένως χρησιμοποιήθηκαν από το διαδίκτυο. Επίσης, βεβαιώνω ότι αυτή η εργασία έχει συγγραφεί από μένα αποκλειστικά και αποτελεί προϊόν πνευματικής ιδιοκτησίας τόσο δικής μου, όσο και του Ιδρύματος.

Παράβαση της ανωτέρω ακαδημαϊκής μου ευθύνης αποτελεί ουσιώδη λόγο για την ανάκληση του πτυχίου μου».

Ο Δηλών



Θα ήθελα να ευχαριστήσω ιδιαίτερα τους καθηγητές του τμήματος Βιομηχανικής Σχεδίασης και Παραγωγής του Πανεπιστημίου Δυτικής Αττικής για τους ορίζοντες που μου άνοιξαν, την συμφοιτήτριά μου Αιμιλία Στυλιανού για την έμπρακτη βοήθειά της, την οικογένειά μου για την αμέριστη στήριξή της και τον εγκάρδιο φίλο μου Χάρη Χαλκιαδάκη, υποψήφιο διδάκτορα, που στέκεται πηγή έμπνευσης και εμψύχωσης όλα αυτά τα χρόνια.

ΠΕΡΙΛΗΨΗ

Η παρούσα διπλωματική εργασία ενασχολείται με τον σχεδιασμό και την υλοποίηση κατασκευής ικανής να πλοηγείτε αυτόνομα σε προσπελάσιμο δρόμο αφού τον αναγνωρίσει επιτυχώς, κάνοντας χρήση οπτικής πληροφορίας η οποία αναλύεται από τεχνητό νευρωνικό δίκτυο νευρώνων με συνάρτηση ενεργοποίησης ακτινικής βάσης. Το κατασκευασμένο όχημα κινείται σε πραγματικό και όχι προσομοιωμένο περιβάλλον και αποπειράται να αντιληφθεί το «δρόμο» όπως και κάποιος οδηγός που κοιτάζοντας, αντιλαμβάνεται ποιο μέρος του οπτικού του πεδίου αντιστοιχεί σε οδόστρωμα και στρέφει το όχημα του προς την ανάλογη κατεύθυνση. Έτσι και η κατασκευή, μετά την οπτική αναγνώριση στρέφεται προς αντίστοιχη κατεύθυνση. Γίνεται εκτενής χρήση συναρτήσεων της βιβλιοθήκης Υπολογιστικής Όρασης OpenCV για την διαμόρφωση και αξιοποίηση της ληφθείσας οπτικής πληροφορίας.

ΛΕΞΕΙΣ ΚΛΕΙΔΙΑ

Μηχανική όραση, Αυτόνομο όχημα, Μικροϋπολογιστής Raspberry Pi, Νευρώνες με συνάρτηση ενεργοποίησης Ακτινικής Βάσης

ABSTRACT

This thesis ‘s subject is the planning and construction of a vehicle able to autonomously navigate road that is successfully recognized via optical means, analyzed by an artificial neural network constituted of neurons with Radial Basis activation Function (RBF). The vehicle is designed for use in real environments where it attempts to perceive the road as a driver would, by seeing, understanding where the drivable road is and leading the vehicle to the correct direction. Like so the built project will drive to the corresponding direction after analyzing the environment. The Computer Vision OpenCV library is extensively used throughout the project for formatting and making the received motion picture suitable for analysis.

KEYWORDS

OpenCV computer vision, Autonomous vehicle, Raspberry Pi microcomputer, Radial Basis activation Function

ΠΙΝΑΚΑΣ ΠΕΡΙΕΧΟΜΕΝΩΝ

ΔΗΛΩΣΗ ΣΥΓΓΡΑΦΕΑ ΔΙΠΛΩΜΑΤΙΚΗΣ ΕΡΓΑΣΙΑΣ iii
ΠΕΡΙΛΗΨΗ v
ΛΕΞΕΙΣ ΚΛΕΙΔΙΑ v
ABSTRACT vi
KEYWORDS vi
ΠΙΝΑΚΑΣ ΠΕΡΙΕΧΟΜΕΝΩΝ vii
ΚΑΤΑΛΟΓΟΣ ΕΙΚΟΝΩΝ viii
ΚΑΤΑΛΟΓΟΣ ΓΡΑΦΗΜΆΤΩΝ ΚΑΙ ΠΙΝΑΚΩΝ ix
ΕΙΣΑΓΩΓΗ 1
ΘΕΩΡΗΤΙΚΟ ΥΠΟΒΑΘΡΟ 2
ΟΡΙΣΜΟΣ ΠΡΟΒΛΗΜΑΤΩΝ 4
ΜΕΘΟΔΟΙ 5
ΣΧΕΔΙΑΣΗ ΚΑΙ ΚΑΤΑΣΚΕΥΗ ΥΛΙΚΟΥ ΜΕΡΟΥΣ (HARDWARE) 11
Λειτουργικά Μέρη Κατασκευής 11
Μηχανολογικό σχέδιο 11
Ηλεκτρονικό κύκλωμα 15
ΑΝΑΛΥΣΗ ΚΑΙ ΣΥΓΓΡΑΦΗ ΛΟΓΙΣΜΙΚΟΥ (SOFTWARE) 17
Διάγραμμα Ροής 17
Κώδικας Προγράμματος 19
ΔΙΕΚΠΕΡΑΙΩΣΗ 33
ΑΠΟΤΕΛΕΣΜΑΤΑ 34
ΣΥΜΠΕΡΑΣΜΑΤΑ 35
Βιβλιογραφία 37

ΚΑΤΑΛΟΓΟΣ ΕΙΚΟΝΩΝ

Εικόνα 1: Μη επεξεργασμένο στιγμιότυπο 5
Εικόνα 2: Στιγμιότυπο που έχει επιδεχθεί επεξεργασία, αλλαγή προοπτικής και προσθήκη καίριων γραμμών.....5
Εικόνα 3: Ληφθέν στιγμιότυπο, χρησιμοποιούμενο για τη βαθμονόμηση της αλλαγής προοπτικής 5
Εικόνα 4: Το αποτέλεσμα της βαθμονομημένης επεξεργασίας της εικόν.....5
Εικόνα 5 : Παρουσιάζονται τα σημεία από τα οποία λαμβάνονται τα δείγματα που συντελούν στην εκπαίδευση του νευρωνικού δικτύου.....6
Εικόνα 6: Ανελυμμενη οπτική πληροφορία και διαχωρισμός τμημάτων οδοστρώματος (λευκός τόνος).....8
Εικόνα 7: Στυγμιότυπο όπου μετά την ανάλογη επεξεργασία εμφανίζονται οι γραμμές που καθορίζουν το όριο του δρόμου και την τροχιά του οχήματος 9
Εικόνα 8: Βάση στήριξης ηλεκτρονικών εξαρτημάτων 11
Εικόνα 9: Αεροτομή και ιστός κάμερας.....12
Εικόνα 10: Βάση στήριξης πρόσθιων αισθητήρων εγγύτητας.....12
Εικόνα 11: Μηχανολογικό σχέδιο.....13
Εικόνα 12: Παρουσίαση καίριων εξαρτημάτων μηχανολογικών(μπλέ πλαίσιο) και ηλεκτρολογικών/ ηλεκτρονικώ(κόκκινο πλαίσιο) απαραίτητων για τη κίνηση του οχήματος.....14
Εικόνα 13: Παρουσίαση καίριων ηλεκτρονικών εξαρτημάτων απαραίτητων για την πλοήγηση του οχήματος.....14
Εικόνα 14: Συνδεσμολογία εξαρτημάτων σε φυσική μορφή 16

ΚΑΤΑΛΟΓΟΣ ΓΡΑΦΗΜΆΤΩΝ ΚΑΙ ΠΙΝΑΚΩΝ

Γράφημα 1: Απεικόνιση της μεθόδου, βάση της οποίας χρησιμοποιούνται τα δεδομένα των δειγμάτων προτύπων για την εκπαίδευση των νευρώνων του δικτύου.....6
Γράφημα 2 : Η τοπολογία του νευρωνικού δικτύου και η λογική με την οποία αναλύεται κάθε στοιχείο της εικόνας 350x240 για την εξαγωγή συμπεράσματος 7
Γράφημα 3 : Μέθοδος εύρεσης αλλαγής κατεύθυνσης του οχήματος.....10
Γράφημα 4: Ηλεκτρολογικό σχέδιο διασύνδεσης εξαρτημάτων της κατασκευής.....15
Γράφημα 5: Διάγραμμα ροής του αλγορίθμου που εκτελείται στον μικροϋπολογιστή RaspberryPi 17
Γράφημα 6: Διάγραμμα ροής του αλγορίθμου που εκτελείται στον μικροελεγκτή Arduino Mega 2560.....18
Γράφημα 7: Διαδρομές ανταλλαγής πληροφοριών μεταξύ των επιμέρους συστημάτων.....32

Πίνακας 1: Συσχέτιση κλίσης σερβοκινητήρα διεύθυνσης και αλλαγής κατεύθυνσης του οχήματος.....10

ΕΙΣΑΓΩΓΗ

Η παρουσιαζόμενη διπλωματική εργασία πραγματεύεται την ικανότητα ενός αυτοοδηγούμενου οχήματος να αναγνωρίσει και να προσπελάσει επιτυχώς δρόμους, οι οποίοι δεν διαθέτουν διακριτά όρια (διαγράμμιση), χρησιμοποιώντας ζωντανή είσοδο εικόνας η οποία αναλύεται από νευρωνικό δίκτυο.

Ο ανθρώπινος παράγοντας εισάγει αβεβαιότητα και εν δυνάμει επικινδυνότητα στις μετακινήσεις με τροχοφόρα οχήματα τα οποία είναι ικανά να αναπτύσσουν υψηλές ταχύτητες. Μη τήρηση των κανόνων κυκλοφορίας μπορεί να αποβεί μοιραία για οδηγό και επιβάτες ενός οχήματος το οποίο ίσως και να μην είναι εκείνο που διέπραξε παράβαση.

Η αυτόνομη οδήγηση οχημάτων παράλληλα με την χρήση μεθόδων διασυνδεδεμένων μετακινήσεων μπορεί να επιφέρει ασφαλέστερες μετακινήσεις και μειωμένη ενεργειακή κατανάλωση, μέσα από την βέλτιστη διαχείριση πόρων. Όμως μεγάλο μέρος του οδικού δικτύου δεν διαθέτει κατάλληλη σήμανση, διαγράμμιση και επαρκή συνεκτικότητα μεταξύ περιοχών και οδών. Τα υπάρχοντα οχήματα με βαθμό αυτονομίας βασίζονται κυρίως στην αναγνώριση λευκών ή κίτρινων γραμμών σήμανσης στο οδόστρωμα ώστε να πορευθούν, επομένως δεν μπορούν να κινηθούν υπό συνθήκες ασάφειας.

Η κατασκευή που δημιουργήθηκε αποπειράται να αναγνωρίσει το προσπελάσιμο οδόστρωμα σε κάθε συνθήκη, έχοντας απλά ένα παράδειγμα εκείνου που περιμένει να ‘δει’ ως δρόμο. Οπτικός αισθητήρας λαμβάνει εικόνα που αναλύεται μέσω μασκών και μετασχηματισμών. Νευρωνικό δίκτυο καλείται να εκτελέσει διαδικασία αναγνώρισης προτύπων σε μικρής ισχύος υπολογιστή, να εξάγει συμπέρασμα για την τροχιά που είναι απαραίτητο να ακολουθηθεί και να επικοινωνήσει με ελεγκτή/ ενεργοποιητές υπεύθυνους για την κίνηση του οχήματος στον χώρο. Για μεγαλύτερη ασφάλεια, μια συστοιχία αισθητήρων εγγύτητας τύπου υπερήχων χρησιμοποιείται για την ακινητοποίηση του αυτοκινούμενου σε περίπτωση αποτυχίας στον σκοπό του.

Το αυτοκινούμενο όχημα που συστάθηκε είναι ένα «έμπειρο» σύστημα το οποίο προσεγγίζει τον τρόπο που ο συγγραφέας της εργασίας αυτής αντιλαμβάνεται την οδήγηση σε ιδανικές ,για αυτή, επιφάνειες. Γίνεται ,δηλαδή, απόπειρα εξαγωγής συμπερασμάτων όμοιων με εκείνα στα οποία ένας οδηγός θα κατέληγε για να προσπελάσει δρόμους.

ΘΕΩΡΗΤΙΚΟ ΥΠΟΒΑΘΡΟ

Η εξέλιξη της τεχνολογίας των υπολογιστών, σε συνδυασμό με την ανάπτυξη και την ευρεία διάθεση αλγορίθμων μηχανικής όρασης, επιτρέπει την επίτευξη του στόχου. Μηχανική όραση είναι η δυνατότητα ανάλυσης μιας εικόνας ή ακολουθίας αυτών από έναν υπολογιστή και η εξαγωγή χρήσιμων συμπερασμάτων από τα εισερχόμενα δεδομένα [1]. Κάνοντας χρήση μεθόδων μηχανικής όρασης και τεχνητών νευρωνικών δικτύων, μπορεί να προσεγγιστεί η ανθρώπινη λειτουργία της οδήγησης από ένα σύστημα και έτσι ο ανθρώπινος παράγοντας να μην είναι απαραίτητος για να συντελεστεί ο χειρισμός ενός οχήματος. Τα τεχνητά νευρωνικά δίκτυα προσπαθούν να προσομοιώσουν τη λειτουργία του ανθρώπινου εγκεφάλου και κυρίως την ικανότητα μάθησης μέσω εκπαίδευσης και εμπειρίας. Μπορούν να εκπαιδευτούν για να επιτελέσουν πολύπλοκες λειτουργίες όπως αναγνώριση προτύπων ή ταξινόμηση δεδομένων και βελτιστοποίηση. Χαρακτηρίζονται ανάλογα την τοπολογία τους και αποτελούνται από νευρώνες. Ο κάθε νευρώνας δέχεται σήματα εισόδου, τα οποία περνά μέσα από μια συνάρτηση ενεργοποίησης και το αποτέλεσμα αυτής βρίσκεται στην έξοδό του. [2]

Η βιβλιοθήκη μηχανικής μάθησης «OPENCV» προσφέρει πληθώρα επιλογών για την διαχείριση, τον μετασχηματισμό και την εξαγωγή συμπερασμάτων από οπτική πληροφορία και χρησιμοποιείται εκτενώς για να εκτελέσει λειτουργίες απαραίτητες για τον σκοπό της κατασκευής. Επίσης γίνεται χρήση βιβλιοθηκών επικοινωνίας και εκτέλεσης μαθηματικών πράξεων, που είναι εξίσου σημαντικό να πραγματοποιηθούν.

Τα αυτοκινούμενα οχήματα χωρίζονται σε κατηγορίες ανάλογα με τον βαθμό αυτονομίας τους και την ικανότητά τους να κινούνται ορθά χωρίς την συμμετοχή οδηγού. Οι κατηγορίες αυτές είναι [3]:

1. Προσφορά βοηθημάτων οδηγού: Διατήρηση σταθερής ταχύτητας ή υποβοήθηση πέδησης.
2. Μερικός αυτοματισμός: Ικανότητα ελέγχου επιτάχυνσης και αλλαγής πορείας. Ο οδηγός πρέπει να έχει τα χέρια του στο τιμόνι.
3. Υπό προϋποθέσεις Αυτοματισμός: Ο οδηγός μπορεί να μην απασχολείται με τον χειρισμό του οχήματος. Υπό ιδανικές συνθήκες, ταχύτητες ως και εξήντα χιλιομέτρων την ώρα μπορούν να επιτευχθούν χωρίς την βοήθεια του χειριστή.
4. Υψηλός αυτοματισμός: Τα οχήματα είναι ικανά αλλαγής πορείας, επιτάχυνσης και επιβράδυνσης κατά το δοκούν. Δύνανται να εντοπίσουν και να αντιδράσουν σε εμπόδια, δεδομένου ιδανικών συνθηκών.
5. Πλήρης αυτοματισμός: Δεν είναι απαραίτητος χειριστής για την κίνηση του οχήματος, σε οποιαδήποτε συνθήκη.

Αυτοκινητοβιομηχανίες ανά την υφήλιο αναπτύσσουν μεθόδους αυτόνομης οδήγησης χαμηλού ή μεγαλύτερου βαθμού, καθώς θεωρείται μείζονος σημασίας στόχος, η επίτευξη του οποίου θα επιφέρει χαμηλότερη ενεργειακή κατανάλωση και εκπομπή ρύπων, απαλλαγή από κυκλοφοριακές συμφορήσεις και θα ωφελήσει το επίπεδο ζωής. Τα αυτοκινούμενα οχήματα δεν

έχουν ακόμα φτάσει σε ικανοποιητικό επίπεδο, με τον βαθμό αυτονομίας τους να περιορίζεται στο βαθμό τρία, για μη πειραματικές εφαρμογές.

Κατά την δεκαετία του 1980, η εταιρία Mercedes-Benz [4] ξεκίνησε να αναπτύσσει και να επιδεικνύει αυτόνομο όχημα, το οποίο έκανε χρήση μηχανικής όρασης για να πορευθεί. Κατά την απόπειρα αυτή, το δοκιμαζόμενο όχημα ταξίδεψε με ταχύτητα άνω των εξήντα χιλιομέτρων ανά ώρα σε δρόμο χωρίς κίνηση. Σήμερα πολλές εταιρίες, μεταξύ τους και κολοσσοί της πληροφορικής, όπως η GOOGLE με το Waymo, προσπαθούν να δημιουργήσουν αυτοδηγούμενα οχήματα. Η διπλωματική αυτή εργασία ασχολείται με την κατασκευή μοντέλου, ικανού να ταξιδέψει με ταχύτητα περίπου πέντε χιλιομέτρων την ώρα σε δρόμο, να αναγνωρίσει εμπόδια και το όριο του δρόμου με οπτικό τρόπο και να αξιοποιήσει επιτυχώς τα εισερχόμενα δεδομένα.

Για να επιτευχθεί το επιθυμητό αποτέλεσμα χρησιμοποιούνται υλικά, λογισμικό και μέθοδοι, κατάλληλα οργανωμένα. Το μέγεθος και ο τύπος του τροχοφόρου οχήματος που επιλέχθηκε, τα αποκλειστικά, τρισδιάστατα, εκτυπωμένα εξαρτήματα που δημιουργήθηκαν, ο τύπος μικροελεγκτή και μικροϋπολογιστή για τους οποίους συγγράφηκαν κατάλληλοι αλγόριθμοι σε διαφορετικές προγραμματιστικές γλώσσες, αποτελούν αναπόσπαστο κομμάτι της εφαρμογής που εκτελέσθηκε.

ΟΡΙΣΜΟΣ ΠΡΟΒΛΗΜΑΤΩΝ

Το ζητούμενο αποτέλεσμα της διπλωματικής εργασίας αυτής, είναι η λήψη διαδοχικών εικόνων και κατάλληλη επεξεργασία και ανάλυση αυτών, ώστε να αναγνωρισθεί οδόστρωμα και η κατασκευή να το ακολουθήσει. Η επίτευξη του στόχου απαιτεί την επίλυση πολλών επιμέρους προβλημάτων όπως:

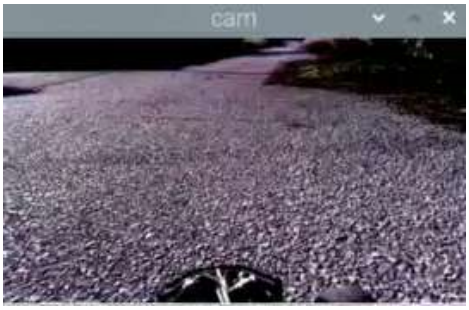
- Την μετατροπή της λαμβανόμενης πληροφορίας και την εξαγωγή χρήσιμων στοιχείων από αυτή.
- Τη σχεδίαση της τροχιάς, που πρέπει το όχημα να ακολουθήσει με βάση τα λαμβανόμενα δεδομένα.
- Την μετάφραση της τροχιάς αυτής, σε αξιοποιήσιμο από το σύστημα στοιχείο και την μετάδοσή της προς τους ενεργοποιητές.
- Οι προαναφερθείσες διεργασίες θα πρέπει να εκτελούνται σε μικρό χρονικό διάστημα, ώστε οι εντολές των ενεργοποιητών να είναι καίριες ως προς τον περιβάλλοντα χώρο.

ΜΕΘΟΔΟΙ

Τα προβλήματα που αντιμετωπίσθηκαν, δύναται να ξεπερασθούν με πληθώρα τρόπων. Δόθηκε, πάραυτα, ιδιαίτερη έμφαση στην επίλυσή τους με μεθόδους μικρού υπολογιστικού φόρτου, ώστε να μπορούν εύκολα να εκτελούνται από συσκευές μικρής ισχύος σε ικανή ταχύτητα.

Η ανάλυση της εικόνας γίνεται μέσω δικτύου νευρώνων. Της ανάλυσης όμως, προηγούνται κάποια βήματα, τα οποία θα φέρουν τα λαμβανόμενα δεδομένα σε μορφή αξιοποιήσιμη για την εξαγωγή σαφών ορίων του δρόμου και θα γίνει δυνατή η χάραξη τροχιάς που θα ακολουθηθεί από το όχημα, αφού ορισθεί η κλίση των εμπρόσθιων τροχών ως προς το επίπεδο οδήγησης.

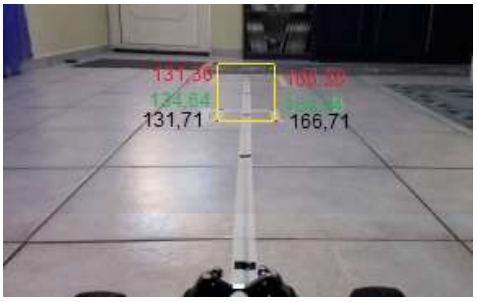
- Η λαμβανόμενη εικόνα μετατρέπεται, ως προς την άποψή της, σε ένα είδος κάτοψης. Έτσι είναι πιο εύκολο να αναγνωρισθεί η γωνία του ορίου του δρόμου ως προς το όχημα.



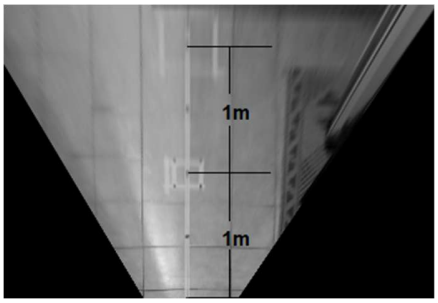
Εικόνα 1: Μη επεξεργασμένο στιγμιότυπο

Εικόνα 2: Στιγμιότυπο που έχει επιδεχθεί επεξεργασία, αλλαγή προοπτικής και προσθήκη καίριων γραμμών

Η αλλαγή άποψης γίνεται με μέθοδο «στρέβλωσης άποψης - warp perspective» και για να πραγματοποιηθεί, οι συντεταγμένες τεσσάρων εικονοστοιχείων επιλέγονται και έπειτα μετατοπίζονται κατάλληλα σε νέα θέση, ώστε να αντιστοιχούν σε μια ‘αφ’ υψηλού’ λήψη, bird eye-view [5]. Για να βρεθούν τα ιδανικά σημεία και οι νέες θέσεις τους, χρησιμοποιήθηκε βαθμονόμηση ενός επιπέδου, του οποίου η λαμβανόμενη εικόνα χρησιμοποιήθηκε ως οδηγός. Τα σημεία, των οποίων οι συντεταγμένες αναγράφονται με μαύρο παραμένουν σταθερά, ενώ εκείνα που φαίνονται με πράσινο μετατοπίστηκαν στις θέσεις εκείνες που γράφονται με κόκκινο χρώμα. Στην εικόνα που προκύπτει, 100 εικονοστοιχεία αντιστοιχούν σε περίπου ένα μέτρο.

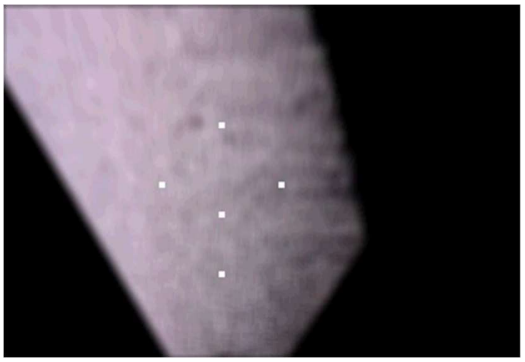


Εικόνα 3: Ληφθέν στιγμιότυπο, χρησιμοποιούμενο για τη βαθμονόμηση της αλλαγής προοπτικής



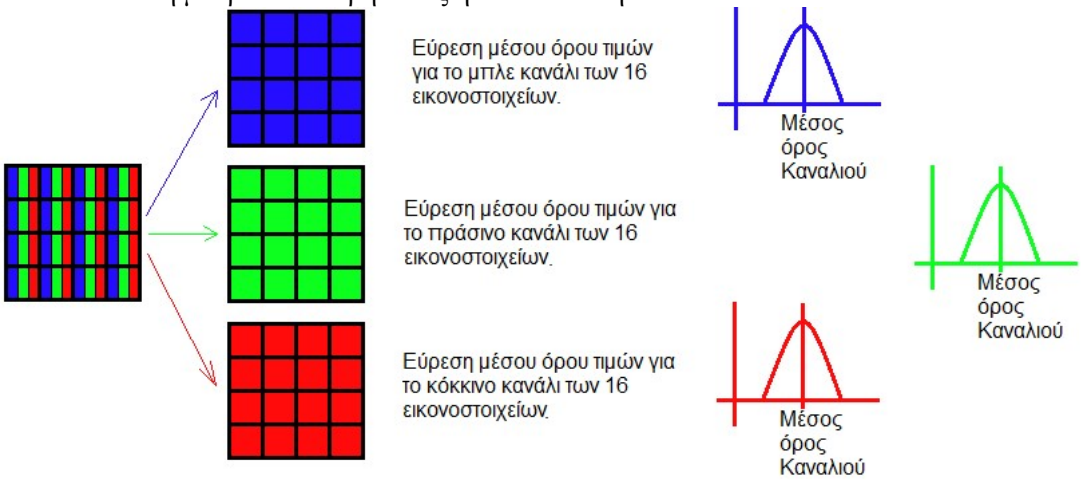
Εικόνα 4: Το αποτέλεσμα της βαθμονομημένης επεξεργασίας της εικόνας

- Πέντε δείγματα “δρόμου” λαμβάνονται. Τα δείγματα είναι μικρά μέρη ενός στιγμιότυπου της ζωντανής εικόνας με ανάλυση 350x240. Ο χειριστής του οχήματος θα πρέπει να το στρέψει προς κάποια κατεύθυνση, όπου ένα αντιπροσωπευτικό τμήμα του δρόμου, που πρόκειται να προσπελαστεί, θα βρίσκεται εντός του οπτικού πεδίου της συσκευής λήψης εικόνας.



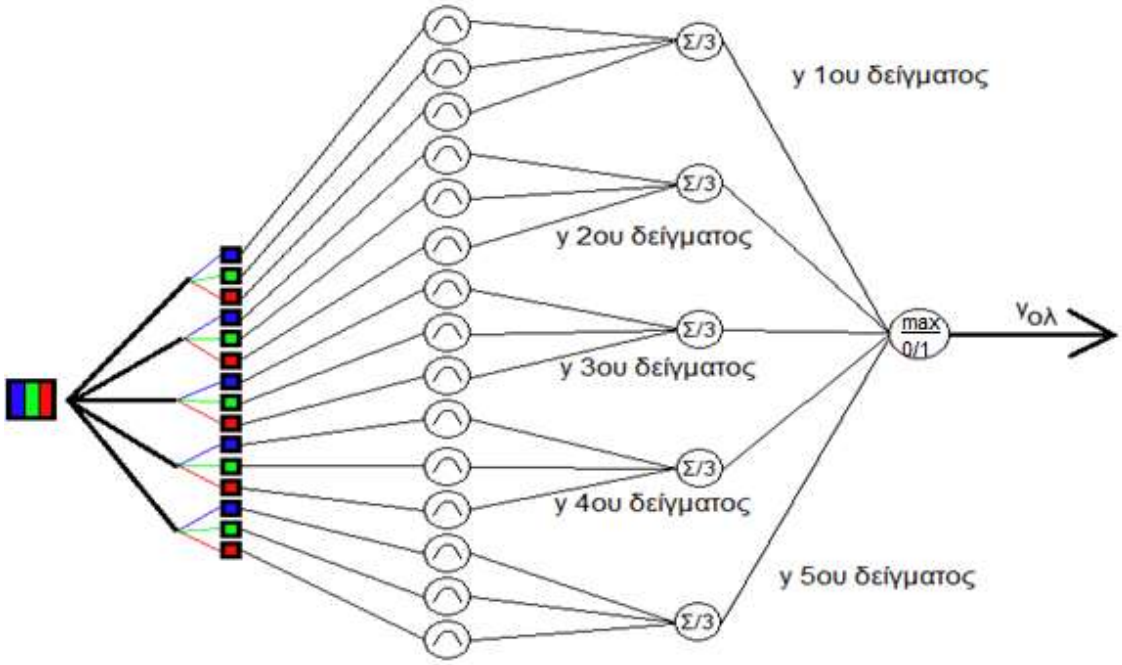
Εικόνα 5 : Παρουσιάζονται τα σημεία από τα οποία λαμβάνονται τα δείγματα που συντελούν στην εκπαίδευση του νευρωνικού δικτύου

Τα δείγματα αυτά έχουν ύψος και πλάτος ίσο με τέσσερα εικονοστοιχεία. Το μέγεθος αυτό επιλέχθηκε, καθώς πειραματικά έδινε τα καλύτερα αποτελέσματα στον διαχωρισμό προσπελάσιμου δρόμου και εμποδίων ή ορίου. Ο μέσος όρος κάθε «καναλιού» χρώματος, για τα δεκαέξι αυτά εικονοστοιχεία, ορίζεται ως το πρότυπο [6], το οποίο θα εισαχθεί στο νευρωνικό δίκτυο για την αναγνώριση προσπελάσιμου επιπέδου. Ένα νέο δείγμα λαμβάνεται μετά την εκκίνηση του προγράμματος ή όταν πιεσθεί το «1». Η διαδικασία αυτή μπορεί να θεωρηθεί ως η «εκπαίδευση» του δικτύου.



Γράφημα 1: Απεικόνιση της μεθόδου, βάση της οποίας χρησιμοποιούνται τα δεδομένα των δειγμάτων προτύπων για την εκπαίδευση των νευρώνων του δικτύου

- Το νευρωνικό δίκτυο αποτελείται από δεκαπέντε νευρώνες ακτινικής βάσης (RBF) και είναι μονού στρώματος. Όλα τα συναπτικά του βάρη είναι ίσα με το ένα για τον κάθε νευρώνα, αφού η βαρύτητα όλων των δεδομένων είναι ίση. Η «καμπάνα» της γκαουσιανής συνάρτησης έχει το κέντρο της στον αριθμό που προκύπτει από τον μέσο όρο κάθε καναλιού, κάθε προτύπου.
  Η γκαουσιανή συνάρτηση έχει ως εξής: (θέτω e = 2.72). Το εύρος της βάσης (σ) προκύπτει από την σχέση: $\boldsymbol{\sigma = \frac{D}{\sqrt{2P}}}$, όπου P είναι το σύνολο των προτύπων (5) και D η απόσταση μεταξύ των πιο απομακρυσμένων κέντρων [6, p. 71].
  Σημειώνεται πως: $\boldsymbol{\sqrt{2P} = \sqrt{2 * 5} = 3.16}$.



Γράφημα 2 : Η τοπολογία του νευρωνικού δικτύου και η λογική με την οποία αναλύεται κάθε στοιχείο της εικόνας 350x240 για την εξαγωγή συμπεράσματος

Η εικόνα που λαμβάνεται, μικραίνει σε ανάλυση κατά πέντε φορές -από 350x240 σε 70x48-, ώστε να υπάρχει μικρότερος όγκος πληροφορίας προς επεξεργασία. Όλα τα εικονοστοιχεία της αρχικής εικόνας συγκρίνονται με τα πρότυπα που λήφθηκαν και ανάλογα με την συνάρτηση ενεργοποίησης του κάθε προτύπου, λαμβάνουν έναν αριθμό, που επιδεικνύει τον βαθμό στον οποίο μπορεί να απεικονίζουν μέρος προσπελάσιμου πεδίου. Για κάθε ένα από τα κανάλια του κάθε ξεχωριστού στοιχείου της εικόνας εξόδου με μέγεθος 70x48, εξάγεται συμπέρασμα χρησιμοποιώντας τη σχέση:

$$\text{τιμή εικονοστοιχείου} = \frac{\sum_{\text{κανάλι}=0}^{2} \frac{\sum_{\text{στήλη στοιχείου}=0}^{4} \sum_{\text{γραμμή στοιχείου}=0}^{4} \mathbf{255} * \mathbf{2.72}^{\left(\frac{\boldsymbol{\text{μέσος όρος} - \text{κέντρο συνάρτησης νευρώνα}}}{\frac{\boldsymbol{\text{μέγιστη απόσταση μεταξύ δειγμάτων}}}{\sqrt{\mathbf{2}} * \boldsymbol{\text{αριθμός δειγμάτων}}}} \right)^{2}}{25}}{3}$$

Όπου κανάλι, το χρωματικό κανάλι που αντιστοιχεί στο μπλε το πράσινο ή το μαύρο. Στήλη και γραμμή στοιχείου είναι το ύψος και το πλάτος του δείγματος που λαμβάνουμε από την εικόνα 350x240 εικονοστοιχείων. Επιλέχθηκε να λαμβάνουμε στοιχεία (kernels) 5x5 στοιχείων ως δείγματα, καθώς το 5 είναι κοινός διαιρέτης του 350 και του 240 και η εικόνα που παράγεται έχει αρκετή πληροφορία για την πλοήγηση του οχήματος. Ο βαθμός αυτός λαμβάνει τιμή από 0 ως 255, με το μέγιστο να σημαίνει πως σίγουρα απεικονίζουν μέρος δρόμου. Κάθε εικονοστοιχείο συγκρίνεται και με τα πέντε ληφθέντα πρότυπα σειριακά και το αποτέλεσμα κάθε νευρώνα είναι μία εικόνα ανάλυσης 70x48, που αποτυπώνει τον βαθμό συμμετοχής κάθε εικονοστοιχείου της σε κάποιο πρότυπο.

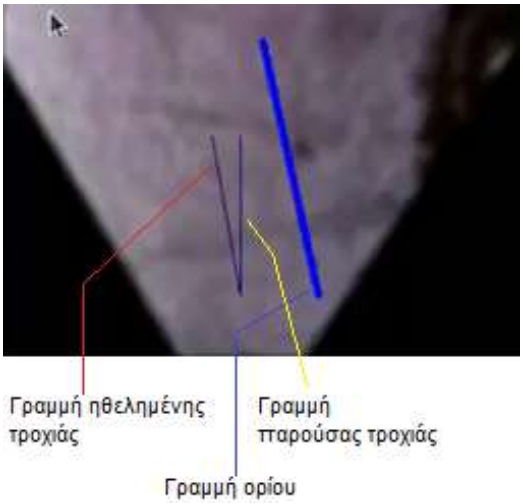
Εικόνα 6: Ανελυμμενη οπτική πληροφορία και διαχωρισμός τμημάτων οδοστρώματος (λευκός τόνος).

Οι πέντε εικόνες συγκρίνονται μεταξύ τους, με τρόπο τέτοιον ώστε τα μέγιστα στοιχεία κάθε μίας να υπερισχύσουν στο τέλος της σύγκρισης (max) και περνούν έπειτα από ένα φίλτρο όπου κάθε εικονοστοιχείο με τιμή μικρότερη του 240 γίνεται μηδενικό (thresholding), δεν συμβολίζει δηλαδή δρόμο.
Με το «3» ενεργοποιείται ή απενεργοποιείται το thresholding.

Η εύρεση προσπελάσιμου δρόμου σε ένα στιγμιότυπο, μπορούσε να γίνει με άλλες μεθόδους. Στο αρχικό στάδιο της κατασκευής δοκιμάστηκε να χρησιμοποιηθεί ανίχνευση ακμών [7] κάνοντας χρήση αλγορίθμου Canny [8] ή Sobel [9]. Μεθόδου Ολικά - Εμφωλευμένης Ανίχθευσης Ακμών (Holistically-Nested Edge Detection) και μεθόδου κατάτμησης εικόνας [10] (Image Segmentation) κάνοντας χρήση μασκών ή κατανομών. Δυστυχώς οι προαναφερθείσες μέθοδοι ήταν είτε ανακριβείς είτε υπολογιστικά απαιτητικές και έτσι δεν χρησιμοποιήθηκαν.

- Εν συνεχεία το δεξί μέρος της επεξεργασμένης εικόνας, είναι εκείνο που επιλέχθηκε να ακολουθηθεί ως όριο δρόμου, καθώς τα οχήματα στην Ελλάδα οφείλουν να κινούνται στην δεξιά μεριά του οδοστρώματος, ανάλογα με την κατεύθυνση κίνησής τους. Μια προκαθορισμένη στήλη ελέγχεται και το δεξιότερο εικονοστοιχείο της, που απεικονίζει «δρόμο», γίνεται μία άκρη του ευθύγραμμου τμήματος, που καθορίζει την τροχιά την οποία θα ακολουθήσει το όχημα. Η άλλη άκρη, βρίσκεται σε σημείο κοντινότερο του οχήματος, και οι συντεταγμένες βρίσκονται με αντίστοιχη διαδικασία. Έχοντας δύο σημεία, βρίσκεται το ‘λ’, που δηλώνει την κλίση του ευθύγραμμου τμήματος και μπορεί να εξαχθεί συμπέρασμα για το μέγεθος της γωνίας σε μοίρες, που πρέπει να περιστραφεί το όχημα. Είναι δυνατό να αλλάζει το απομακρυσμένο ,από το όχημα, σημείο στόχευσης μεταφερόμενο πιο κοντά σε αυτό, με τη χρήση του πλήκτρου «9».

- Αφού καθοριστεί η γωνία του ορίου, υπολογίζεται η απόστασή του από την θέση του οχήματος και γίνονται οι απαραίτητες διορθώσεις στην τροχιά, που πρόκειται να ακολουθηθεί. Είναι δυνατό να χρησιμοποιηθεί αλγόριθμος απόρριψης τμημάτων μετά από συγκεκριμένο αριθμό εικονοστοιχείων που δεν εκφράζουν ‘δρόμο’, πιέζοντας «2» στο πληκτρολόγιο. Έχει οριστεί επιθυμητή απόσταση του οχήματος από το όριο.



Εικόνα 7: Στυγμιότυπο όπου μετά την ανάλογη επεξεργασία εμφανίζονται οι γραμμές που καθορίζουν το όριο του δρόμου και την τροχιά του οχήματος

- Για την εντολή εκκίνησης του οχήματος, ο χρήστης μπορεί να πιέσει 6. Για να φρενάρει το όχημα πιέζεται το 7 και για να παύσει λειτουργίας, χρησιμοποιείται το 8. Ανάλογα επομένως την εντολή που έχει δοθεί, ο μικροϋπολογιστής τύπου «raspberry pi4» αποστέλλει στο μικροελεγκτή τύπου «Arduino mega» τιμές για την επιτάχυνση του οχήματος και την αλλαγή κατεύθυνσής του. Το «Arduino mega» αναλαμβάνει να χειριστεί τους ενεργοποιητές και να εξετάσει το περιβάλλον για κινδύνους μέσω των υπέρηχων αισθητήρων. Αν κίνδυνος εντοπιστεί, το όχημα παύει να κινείται.

- Απαραίτητη είναι η συσχέτιση της περιστροφής του σερβοκινητήρα αλλαγής κατεύθυνσης, με την αλλαγή που συντελείται για συγκεκριμένη είσοδο. Επιλέχθηκαν σαν μετρούμενο μέγεθος, οι μοίρες περιστροφής του οχήματος ανά μέτρο διανυθείσας απόστασης. Το όχημα τοποθετήθηκε σε δάπεδο και μετρήθηκε η περίμετρος των τροχών του, η οποία βρέθηκε 31cm. Επίσης η απόσταση μεταξύ του κέντρου των δύο τροχών μετρήθηκε 26cm. Πειραματικά διαπιστώθηκε, πως αν δοθεί εντολή διατήρησης 90^o στον σερβοκινητήρα, το όχημα θα κινηθεί ευθεία. Για εντολή 60^o, το όχημα θα στραφεί αριστερά και για 120^o δεξιά. Λήφθηκαν μετρήσεις για κάθε περίπτωση, εως ότου το όχημα διανύσει απόσταση λίγο μεγαλύτερη του ενός μέτρου.

Εντολή 60 μοιρών (Αριστερά)			Εντολή 120 μοιρών (Δεξιά)		
Περιστροφές Αριστερού Τροχού	Περιστροφές Δεξιού Τροχού	Διανυθείσα Απόσταση (cm)	Περιστροφές Αριστερού Τροχού	Περιστροφές Δεξιού Τροχού	Διανυθείσα Απόσταση (cm)
0.75	1	27.125	1	0.8	27.9
1.5	2	54.25	2	1.6	55.8
2.3	3	82.15	3	2.35	82.93
2.5	3.25	89.125	3.25	2.6	90.675
2.75	3.6	98.425	3.6	2.8	99.2
3.1	4	110.05	4	3.2	111.6

Πίνακας 1: Συσχέτιση κλίσης σερβοκινητήρα διεύθυνσης και αλλαγής κατεύθυνσης του οχήματος

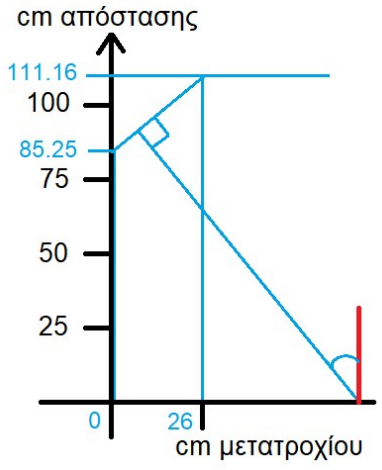
Για ένα μέτρο απόστασης, η γωνία κατά την οποία περιστράφηκε το όχημα στον άξονα διεύθυνσής του, για αλλαγή πορείας προς αριστερά θα είναι:

$$\lambda = \frac{Απόσταση\ που\ διάνυσε\ ο\ Αριστερός\ τροχός - Απόσταση\ που\ διάνυσε\ ο\ Δεξής\ τροχός}{-\ Εμπρόσθιο\ Μετατρόχιο} =>$$

$$=> \lambda = \frac{85.35 - \quad .6}{26} = 1.013462\ και\ tan^{-1}\lambda = 45.38°/m$$

Επομένως, αν δοθεί στον σερβοκινητήρα εντολή να διατηρήσει γωνία 60^o και το όχημα κινηθεί εως ότου το κέντρο του μετατροχίου του έχει διανύσει απόσταση ενός μέτρου, θα έχει στραφεί προς αριστερά κατά 45.38^o /m. Αντίστοιχα, για αλλαγή προς τα δεξιά, σε απόσταση ενός μέτρου, το όχημα θα έχει στραφεί προς τα δεξιά κατά 43.64^o/m.

$$(\lambda = \frac{Απόσταση\ που\ διάνυσε\ ο\ Δεξής\ τροχός - Απόσταση\ που\ διάνυσε\ ο\ Αριστερός\ τροχός}{Εμπρόσθιο\ Μετατρόχιο})$$



Γράφημα 3 : Μέθοδος εύρεσης αλλαγής κατεύθυνσης του οχήματος

Οι τιμές για αποστάσεις μικρότερες ή και μεγαλύτερες του ενός μέτρου θεωρούνται γραμμικές. Άρα, για 30^o αλλαγής γωνίας στον σερβοκινητήρα, λαμβάνουμε αλλαγή περίπου 44 ^{o} για ένα κάθε μέτρο που διανύεται. Η απόσταση στόχευσης είναι 1.5 με δύο μέτρα. Για 30^o και 1.5 μέτρα συντελούνται 66^o αλλαγής πορείας. Για 30^o και 2 μέτρα έχουμε 88^o αλλαγής πορείας. Μπορεί να τεθεί μέσος όρος 77 μοιρών και να οριστεί πολλαπλασιαστής μοιρών αλλαγής του σερβοκινητήρα ίσος με 30/77 = 0.38.

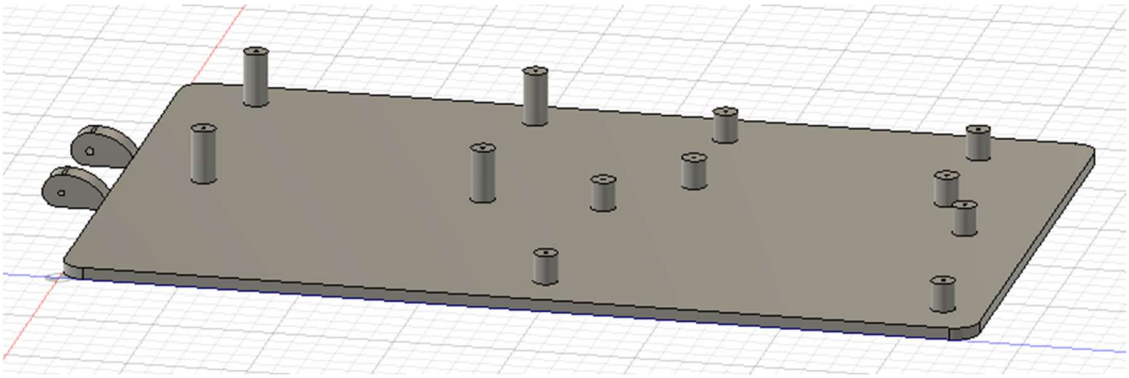
ΣΧΕΔΙΑΣΗ ΚΑΙ ΚΑΤΑΣΚΕΥΗ ΥΛΙΚΟΥ ΜΕΡΟΥΣ (HARDWARE)

Λειτουργικά Μέρη Κατασκευής

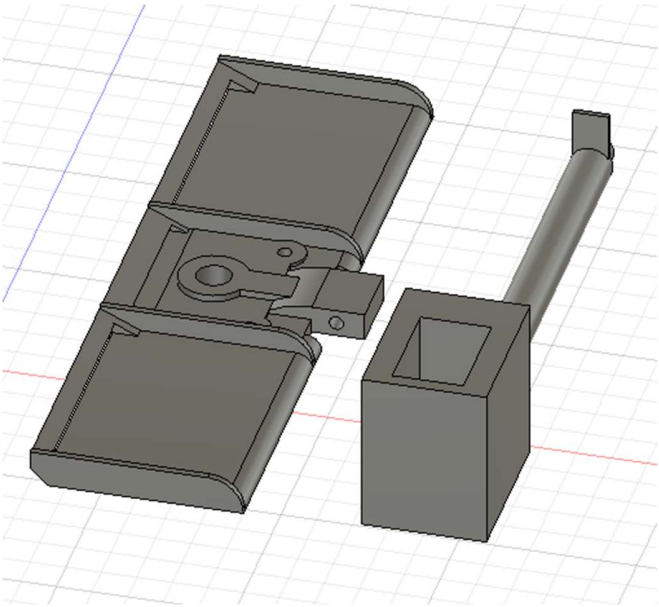
Η κατασκευή αποτελείται από το κύριο μέρος του οχήματος, απαρτιζόμενο από το σύστημα πέδησης, διεύθυνσης, τον ηλεκτρικό κινητήρα κίνησης και τα επιμέρους εξαρτήματα μετάδοσης κίνησης, τον συσσωρευτή, τον ελεγκτή ταχύτητας και τον σερβοκινητήρα αλλαγής διεύθυνσης. Διαθέτει επίσης έναν ιστό, όπου τοποθετείται ο αισθητήρας εικόνας και την θήκη συστοιχίας αισθητήρων υπερήχων. Σημαντικό μέρος είναι ακόμα και η βάση στην οποία στηρίζεται ο μικροϋπολογιστής Raspberry Pi 4, ο μικροελεγκτής Arduino Mega 2560, και ο buck converter που αναλαμβάνει την τροφοδοσία αυτών.

Μηχανολογικό σχέδιο

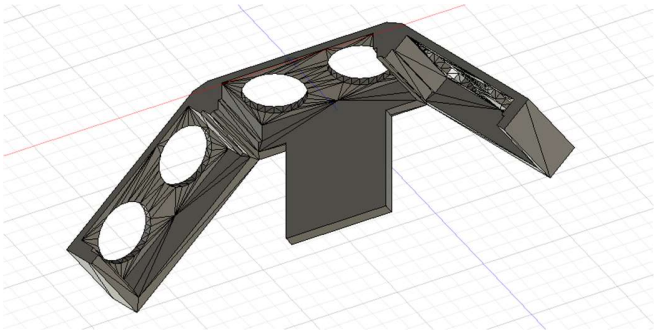
Παρουσιάζονται εξαρτήματα του οχήματος, τα οποία σχεδιάστηκαν στο πρόγραμμα Fusion 360 της Autodesk και εκτυπώθηκαν από τρισδιάστατο εκτυπωτή κάνοντας χρήση PLA (Polylactic acid ή πολυγαλακτκό οξύ) καθώς και το μηχανολογικό σχέδιο της κατασκευής.



Εικόνα 8: Βάση στήριξης ηλεκτρονικών εξαρτημάτων

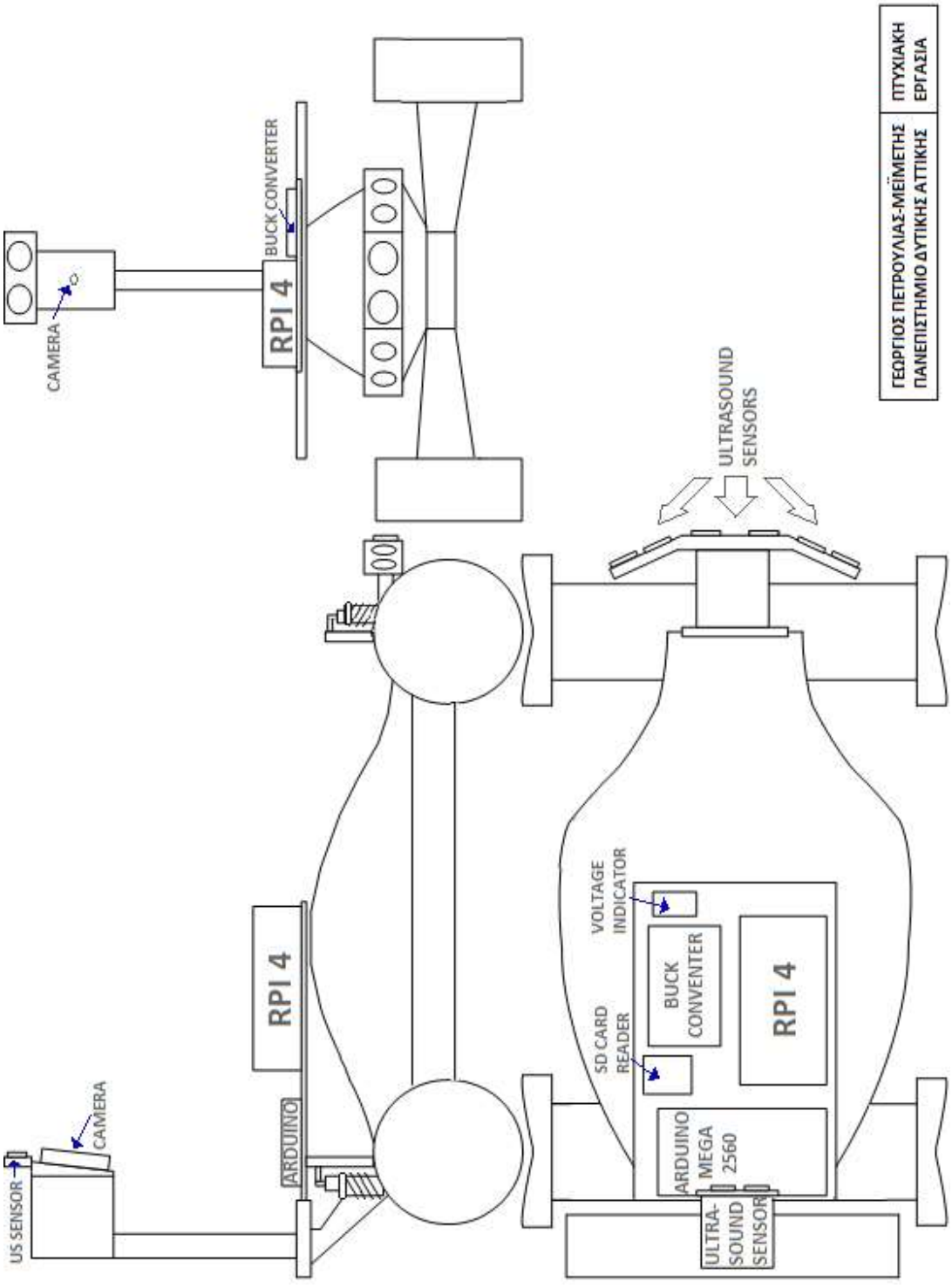


Εικόνα 9: Αεροτομή και ιστός κάμερας

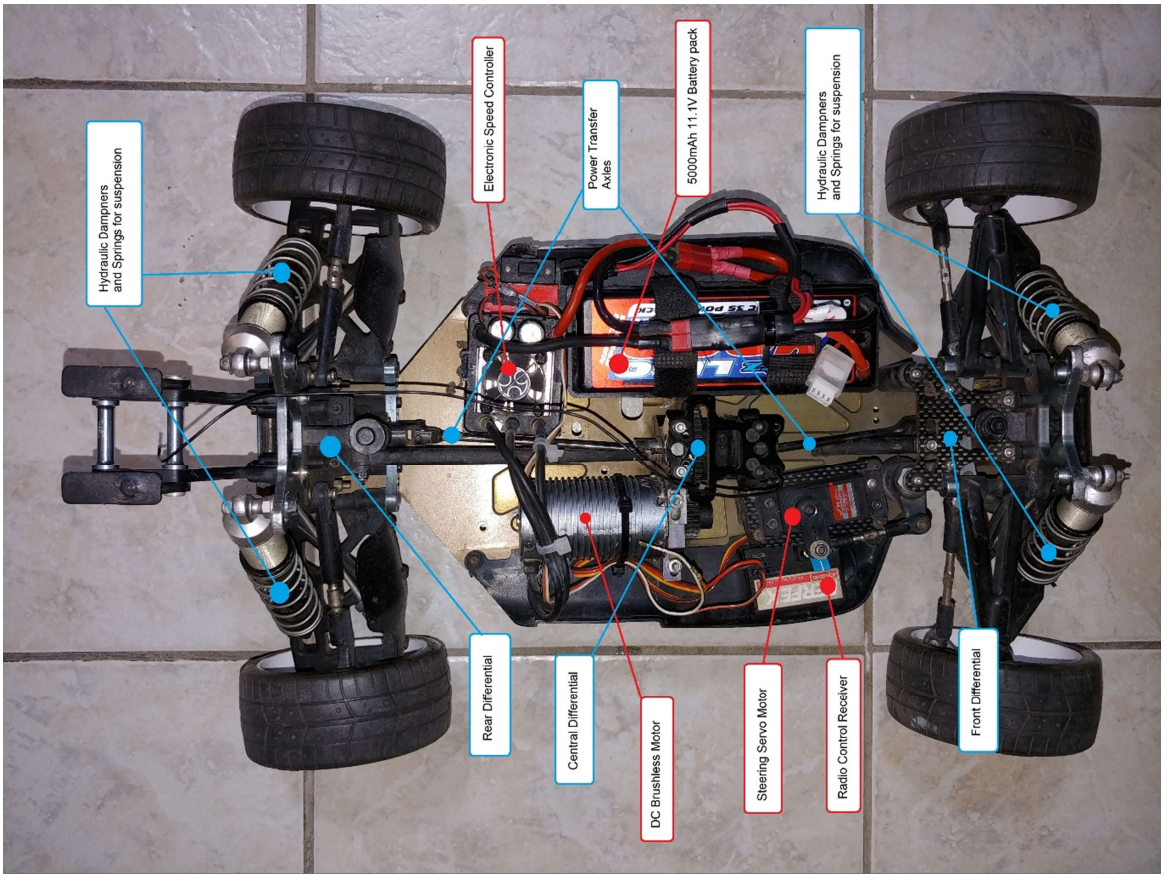


Εικόνα 10: Βάση στήριξης πρόσθιων αισθητήρων εγγύτητας

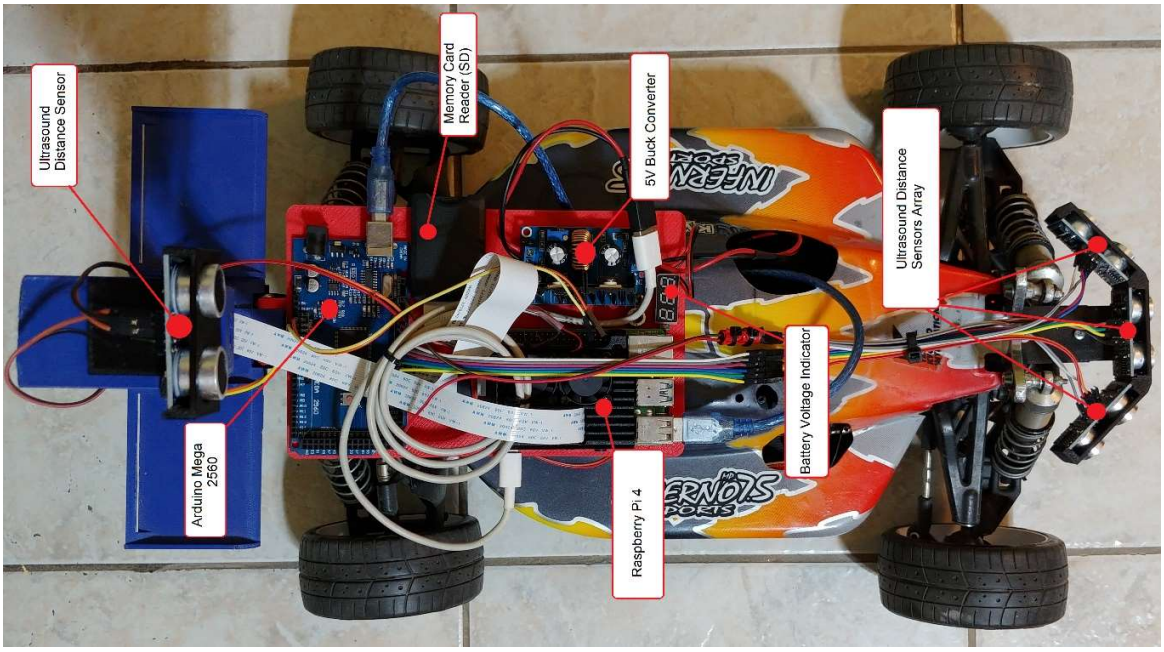
Κατασκευάστηκε πληθώρα προτύπων και υπήρξαν αρκετές αποτυχημένες δοκιμές, κατά τη διάρκεια των οποίων αριθμός εξαρτημάτων κατέστη μη επιδιορθώσιμος.



Εικόνα 11: Μηχανολογικό σχέδιο



Εικόνα 12: Παρουσίαση καίριων εξαρτημάτων μηχανολογικών(μπλέ πλαίσιο) και ηλεκτρολογικών/ ηλεκτρονικώ(κόκκινο πλαίσιο) απαραίτητων για τη κίνηση του οχήματος

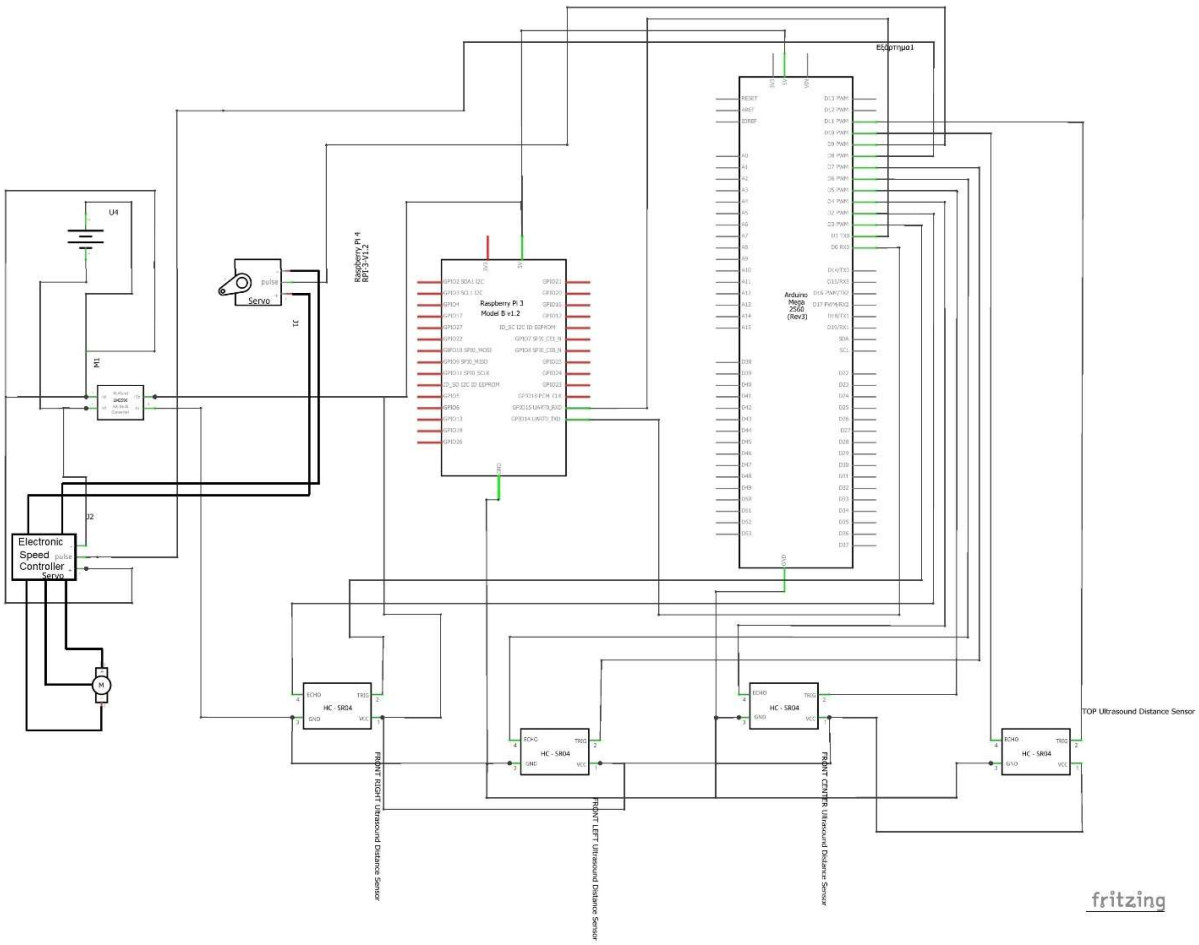


Εικόνα 13: Παρουσίαση καίριων ηλεκτρονικών εξαρτημάτων απαραίτητων για την πλοήγηση του οχήματος

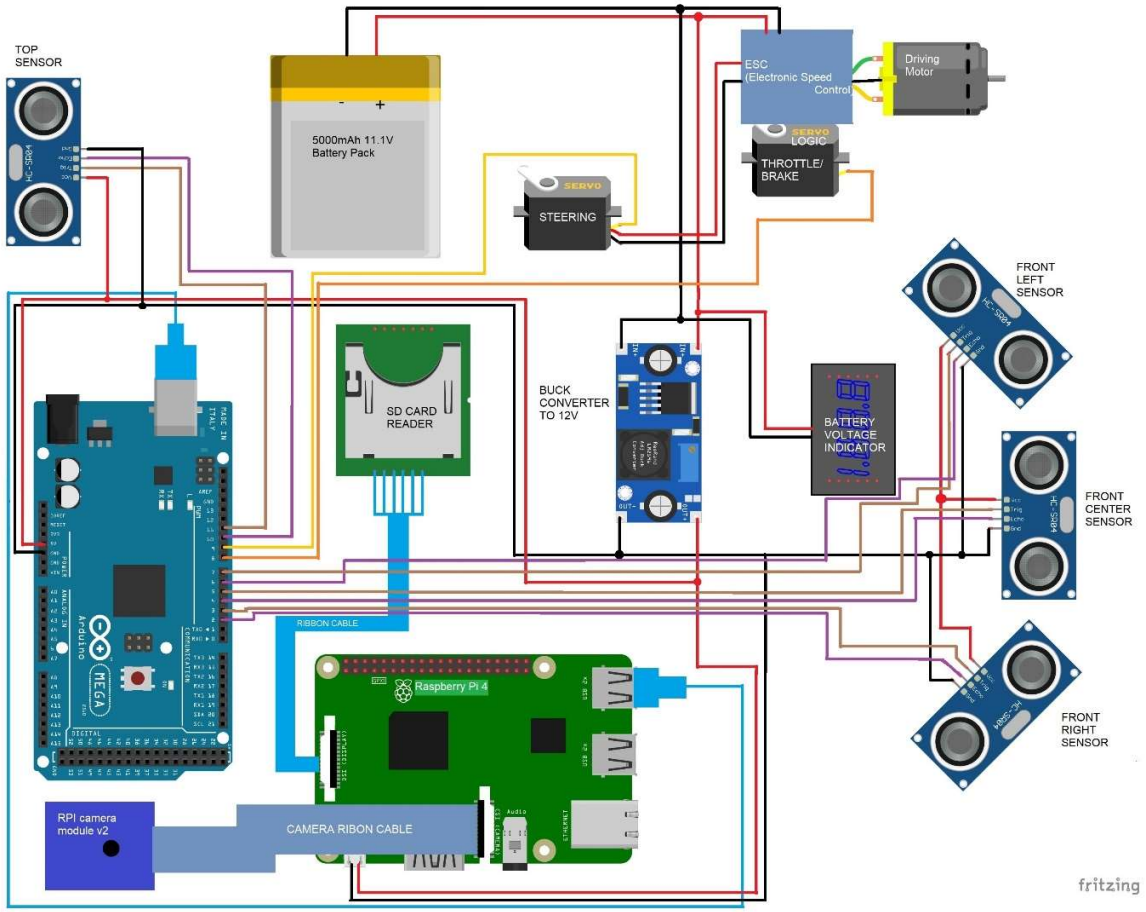
Ηλεκτρονικό κύκλωμα

Για την τροφοδοσία της κατασκευής, χρησιμοποιείται πηγή ιόντων λιθίου 11.1V, 5000mAh, η οποία τροφοδοτεί άμεσα τον Ηλεκτρονικό Ελεγκτή Ταχύτητας (electronic speed controller) και τον υποβιβαστή τάσης (buck converter). Ο HET παρέχει τάση στον σερβοκινητήρα αλλαγής κατεύθυνσης και τροφοδοτεί τον σύγχρονο κινητήρα κίνησης με παλμούς συχνότητας ανάλογης με την ταχύτητα που ζητείται να περιστραφεί. Ο υποβιβαστής τάσης διαμορφώνει στην έξοδό του τάση 5.1V και αναλαμβάνει να τροφοδοτήσει το μικροελεγτή Arduino Mega, το μικροϋπολογιστή Raspberry Pi, και τους τέσσερεις αισθητήρες απόστασης με υπέρηχους (ultrasound distance sensors). Ο Arduino Mega μέσω των ψηφιακών εισόδων και εξόδων του, σκανδαλίζει τους αισθητήρες υπερήχων οι οποίοι παράγουν ένα ηχητικό κύμα. Το κύμα ανακλάται σε επιφάνειες και επιστρέφει. Ο χρόνος από την παραγωγή του υπερήχου, έως την επιστροφή του, μετράται από τον ελεγκτή και υπολογίζεται η απόσταση της επιφάνειας.

.



Γράφημα 4: Ηλεκτρολογικό σχέδιο διασύνδεσης εξαρτημάτων της κατασκευής



Εικόνα 14: Συνδεσμολογία εξαρτημάτων σε φυσική μορφή

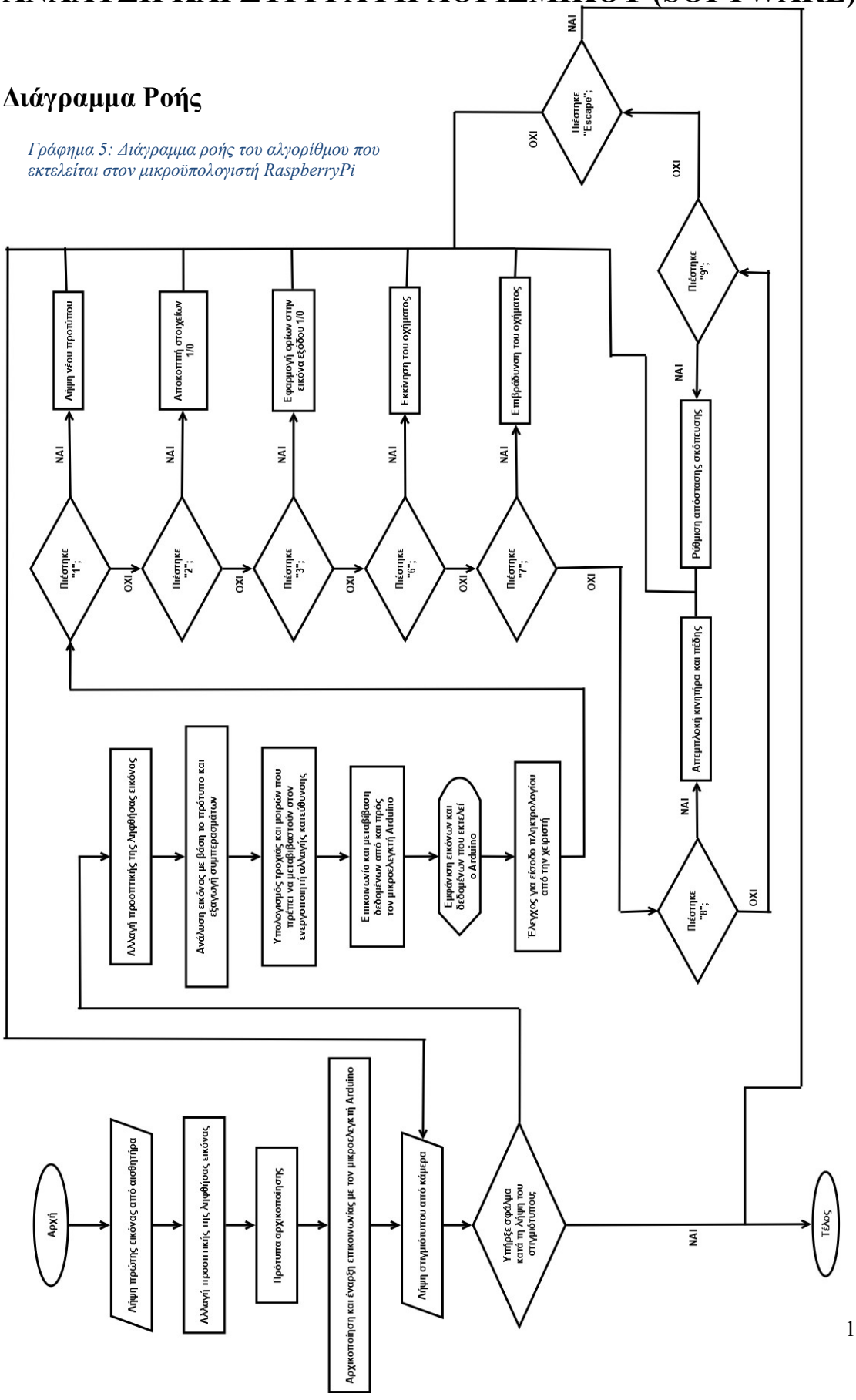
Ο ψηφιακός αισθητήρας εικόνας συνδέεται με το Raspberry Pi μέσω της ειδικής πόρτας και μεταφέρει δεδομένα διαμέσου ενός εύκαμπτου καλωδίου ταινίας. Με τον ίδιο τρόπο συνδέεται σε αυτό και ο αναγνώστης καρτών μνήμης.

Το Raspberry Pi και το Arduino επικοινωνούν μέσω ενιαίου σειριακού διαύλου, κάνοντας χρήση αντίστοιχης καλωδίωσης και πρωτοκόλλου σειριακής επικοινωνίας.

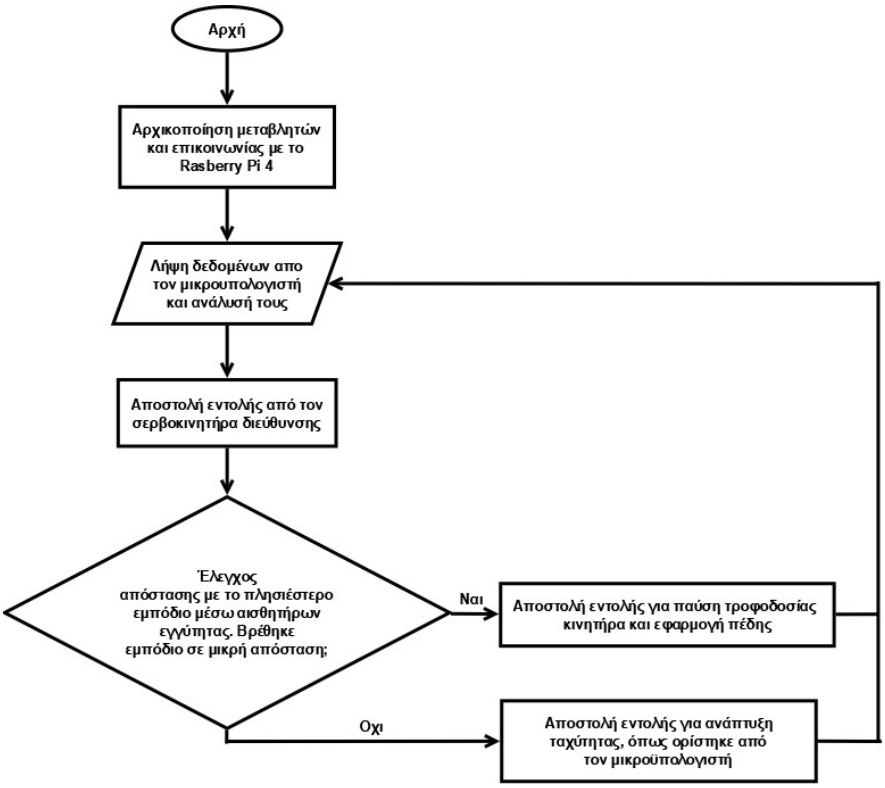
ΑΝΑΛΥΣΗ ΚΑΙ ΣΥΓΓΡΑΦΗ ΛΟΓΙΣΜΙΚΟΥ (SOFTWARE)

Διάγραμμα Ροής

Γράφημα 5: Διάγραμμα ροής του αλγορίθμου που εκτελείται στον μικροϋπολογιστή RaspberryPi



Παρουσιάζονται τα διαγράμματα ροής του μικροϋπολογιστή Raspberry Pi 4 (άνω) και του μικροελεγκτή Arduino Mega 2560 (ακολούθως).



Γράφημα 6: Διάγραμμα ροής του αλγορίθμου που εκτελείται στον μικροελεγκτή Arduino Mega 2560

Κώδικας Προγράμματος

Ακολουθεί ο κώδικας C++ που εκτελείται στον μικροϋπολογιστή Raspberry Pi 4.

```
//Purpose
//     A picture gets analyzed and a general course
//     is planned. Data is gathered and send to a system
//     that can make a vehicle move to accrdind to the desired
//     course, via serial communication.
//     NNtest_v1.4_stable
//     George Petroulias Meimetis
////////////////////////////////////////////////////////////////////////////

#include <iostream> // for standard I/O
#include <string>   // for strings
#include <iomanip>  // for controlling float print precision
#include <sstream>  // string to number conversion
#include <unistd.h> // for time on linux
#include <algorithm>// std::max,min
#include <thread>   // std::thread
#include <future>   // std::thread
#include <cmath>
#include <ctime>    //timer for communication purposes
#include <wiringPi.h>
#include <wiringSerial.h> //serial communication

#include <opencv2/core.hpp>     // Basic OpenCV structures (cv::Mat, Scalar)
#include <opencv2/videoio.hpp>
#include <opencv2/highgui.hpp>  // OpenCV window I/O
#include <opencv2/core/ocl.hpp> // OpenCL UMat calculations
#include <opencv2/imgproc.hpp>  // Image Processing
using namespace std;
using namespace cv;
//0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000

//Purpose
//     Gets a frame still from the connected camera
//     Returns the still for analysis
//Hardware
//     Connected image sensor
//Software
//     Uses a mat to store the frame
//     Uses OPENCV library functions
cv::Mat camera_work(cv::VideoCapture cap){
       cv::Mat Camera_Input_Image;
       if (!cap.isOpened())
    {
        cout << "--(!)Error opening video capture" << endl;
    }
       cap >> Camera_Input_Image;
       return Camera_Input_Image;
}

//Purpose
//     Calculates the centre and width of the gaussian bells that control the
//     output of the neurons. Creates an image that shows the used kernels.
//Software
//     Uses OPENCV library functions
void Bell_Centres(cv::Mat Warped_Input_Image, int *Centres_Array){
       //5 Analyse and compute Gaussian centers for neurons
       int collumns = 0, lines = 0;
////Centres_Array: centre_B1 0,centre_B2 1,centre_B2 2,centre_B4 3,centre_B5 4
```

```
////  centre_G1 5,centre_G2 6,centre_G2 7,centre_G4 8,centre_G5 9
////  centre_R1 10,centre_R2 11,centre_R2 12,centre_R4 13,centre_R5 14
////  Distance_of_Centres_B 15,Distance_of_Centres_G 16,
////  Distance_of_Centres_R 17
       for (lines = 200; lines < 204; lines ++){
              for (collumns = 146; collumns < 150; collumns ++){

                     Centres_Array[0] = (Centres_Array[0] + Warped_Input_Image.
                     at<cv::Vec3b>(lines - 20,collumns)[0]);
                     Centres_Array[1] = (Centres_Array[1] + Warped_Input_Image.
                     at<cv::Vec3b>(lines - 60,collumns)[0]);
                     Centres_Array[2] = (Centres_Array[2] + Warped_Input_Image.
                     at<cv::Vec3b>(lines - 120,collumns)[0]);
                     Centres_Array[3] = (Centres_Array[3] + Warped_Input_Image.
                     at<cv::Vec3b>(lines - 80,collumns - 40)[0]);
                     Centres_Array[4] = (Centres_Array[4] +
                     Warped_Input_Image.at<cv::Vec3b>
                     (lines - 80,collumns + 40)[0]);
                     Warped_Input_Image.at<cv::Vec3b>
                     (lines - 20,collumns)[0] = 255;
                     Warped_Input_Image.at<cv::Vec3b>
                     (lines - 60,collumns)[0] = 255;
                     Warped_Input_Image.at<cv::Vec3b>
                     (lines - 120,collumns)[0] = 255;
                     Warped_Input_Image.at<cv::Vec3b>
                     (lines - 80,collumns - 40)[0] = 255;
                     Warped_Input_Image.at<cv::Vec3b>
                     (lines - 80,collumns + 40)[0] = 255;
                     Centres_Array[5] = (Centres_Array[5] + Warped_Input_Image.
                     at<cv::Vec3b>(lines - 20,collumns)[1]);
                     Centres_Array[6] = (Centres_Array[6] + Warped_Input_Image.
                     at<cv::Vec3b>(lines - 60,collumns)[1]);
                     Centres_Array[7] = (Centres_Array[7] + Warped_Input_Image.
                     at<cv::Vec3b>(lines - 120,collumns)[1]);
                     Centres_Array[8] = (Centres_Array[8] + Warped_Input_Image.
                     at<cv::Vec3b>(lines - 80,collumns - 40)[1]);
                     Centres_Array[9] = (Centres_Array[9] +
                     Warped_Input_Image.at<cv::Vec3b>
                     (lines - 80,collumns + 40)[1]);
                     Warped_Input_Image.at<cv::Vec3b>
                     (lines - 20,collumns)[1] = 255;
                     Warped_Input_Image.at<cv::Vec3b>
                     (lines - 60,collumns)[1] = 255;
                     Warped_Input_Image.at<cv::Vec3b>
                     (lines - 120,collumns)[1] = 255;
                     Warped_Input_Image.at<cv::Vec3b>
                     (lines - 80,collumns - 40)[1] = 255;
                     Warped_Input_Image.at<cv::Vec3b>
                     (lines - 80,collumns + 40)[1] = 255;

                     Centres_Array[10] = (Centres_Array[10] + Warped_Input_Image.
                     at<cv::Vec3b>(lines - 20,collumns)[2]);
                     Centres_Array[11] = (Centres_Array[11] + Warped_Input_Image.
                     at<cv::Vec3b>(lines - 60,collumns)[2]);
                     Centres_Array[12] = (Centres_Array[12] + Warped_Input_Image.
                     at<cv::Vec3b>(lines - 120,collumns)[2]);
                     Centres_Array[13] = (Centres_Array[13] + Warped_Input_Image.
                     at<cv::Vec3b>(lines - 80,collumns - 40)[2]);
                     Centres_Array[14] = (Centres_Array[14] +
                     Warped_Input_Image.at<cv::Vec3b>
                     (lines - 80,collumns + 40)[2]);
                     Warped_Input_Image.at<cv::Vec3b>
                     (lines - 20,collumns)[2] = 255;
```

```
                        Warped_Input_Image.at<cv::Vec3b>
                        (lines - 60,collumns)[2] = 255;
                        Warped_Input_Image.at<cv::Vec3b>
                        (lines - 120,collumns)[2] = 255;
                        Warped_Input_Image.at<cv::Vec3b>
                        (lines - 80,collumns - 40)[2] = 255;
                        Warped_Input_Image.at<cv::Vec3b>
                        (lines - 80,collumns + 40)[2] = 255;

                }
        }
        Centres_Array[0] = Centres_Array[0] / 16;
        Centres_Array[1] = Centres_Array[1] / 16;
        Centres_Array[2] = Centres_Array[2] / 16;
        Centres_Array[3] = Centres_Array[3] / 16;
        Centres_Array[4] = Centres_Array[4] / 16;
        Centres_Array[5] = Centres_Array[5] / 16;
        Centres_Array[6] = Centres_Array[6] / 16;
        Centres_Array[7] = Centres_Array[7] / 16;
        Centres_Array[8] = Centres_Array[8] / 16;
        Centres_Array[9] = Centres_Array[9] / 16;
        Centres_Array[10] = Centres_Array[10] / 16;
        Centres_Array[11] = Centres_Array[11] / 16;
        Centres_Array[12] = Centres_Array[12] / 16;
        Centres_Array[13] = Centres_Array[13] / 16;
        Centres_Array[14] = Centres_Array[14] / 16;
        //Gausian bell function width
        float maxB = std::max({Centres_Array[0],
        Centres_Array[1],Centres_Array[2],
        Centres_Array[3],Centres_Array[4]});
        float maxG = std::max({Centres_Array[5],
        Centres_Array[6],Centres_Array[7],
        Centres_Array[8],Centres_Array[9]});
        float maxR = std::max({Centres_Array[10],
        Centres_Array[11],Centres_Array[12],
        Centres_Array[13],Centres_Array[14]});
        float minB = std::min({Centres_Array[0],
        Centres_Array[1],Centres_Array[2],
        Centres_Array[3],Centres_Array[4]});
        float minG = std::min({Centres_Array[5],
        Centres_Array[6],Centres_Array[7],
        Centres_Array[8],Centres_Array[9]});
        float minR = std::min({Centres_Array[10],
        Centres_Array[11],Centres_Array[12],
        Centres_Array[13],Centres_Array[14]});
        float Distance_of_Centres_B = maxB - minB;
        float Distance_of_Centres_G = maxG - minG;
        float Distance_of_Centres_R = maxR - minR;
        Centres_Array[15] = Distance_of_Centres_B;
        Centres_Array[16] = Distance_of_Centres_G;
        Centres_Array[17] = Distance_of_Centres_R;
        cv::imwrite("320,180warp.jpg",Warped_Input_Image);
}

//Purpose
//     Wraps the perspective of a framestill and returns it.
//Software
//     Uses OPENCV library functions
cv::Mat thread_warping(cv::Mat image, cv::Mat Image_Warp,
cv::Mat transformtable){
        cv::warpPerspective(image, Image_Warp, transformtable, cv::Size(350,
240));
        cv::blur(Image_Warp,Image_Warp,cv::Size(9,9),Point(-1,-1),0);
```

```
	return Image_Warp;
}

//Purpose
//	Every frame is analysed through the neural network, bit by bit and for
each
//	pixel a decision is made about how close it is to a preassigned as road
//	example, according to colour. Returns an analysed frame. The analysed
frame is
//	a mat that every element = 255 is road and the others (0) are not.
//	The roadside line points are calculated.
//Software
//	Uses OPENCV library functions
cv::Mat NN_frame_analysis (cv::Mat Image_Warp, cv::Mat mask7048,
 int *Centres_Array, int *linepoints, int Cut, int limit,
 int Forward_Targeting_Distance){
	cv::Mat Output_Image1(48,70,CV_8U);
	cv::Mat Output_Image2(48,70,CV_8U);
	cv::Mat Output_Image3(48,70,CV_8U);
	cv::Mat Output_Image4(48,70,CV_8U);
	cv::Mat Output_Image5(48,70,CV_8U);
	cv::Mat Output_Image(48,70,CV_8U);
	cv::Mat morphing_kernel = getStructuringElement(MORPH_RECT,cv::Size(3,3),
	cv::Size(-1,-1));
	int lines_Normalized, collumns_Normalized;
	long int Averages_Sum_B, Averages_Sum_G, Averages_Sum_R,
	state_change, sum_state_change;
	int Neuron_Output_BGR1, Neuron_Output_BGR2, Neuron_Output_BGR3,
	Neuron_Output_BGR4, Neuron_Output_BGR5;
	int Neuron_Output_B1, Neuron_Output_G1, Neuron_Output_R1;
	int Neuron_Output_B2, Neuron_Output_G2, Neuron_Output_R2;
	int Neuron_Output_B3, Neuron_Output_G3, Neuron_Output_R3;
	int Neuron_Output_B4, Neuron_Output_G4, Neuron_Output_R4;
	int Neuron_Output_B5, Neuron_Output_G5, Neuron_Output_R5;
	for (int collumns = 0; collumns < 70; collumns ++){
		for (int lines = 0; lines < 48; lines ++){
			long int sum_B = 0, sum_G = 0, sum_R = 0;
			for (int  collumns_Image_Portion = 0;
			collumns_Image_Portion < 5; collumns_Image_Portion ++){
				for (int  lines_Image_Portion = 0;
				lines_Image_Portion < 5; lines_Image_Portion ++){
					lines_Normalized = (lines*5) +
					lines_Image_Portion;
					collumns_Normalized = (collumns*5) +
					collumns_Image_Portion;
					sum_B = sum_B + Image_Warp.at<cv::Vec3b>
					(lines_Normalized,collumns_Normalized)[0];
					sum_G = sum_G + Image_Warp.at<cv::Vec3b>
					(lines_Normalized,collumns_Normalized)[1];
					sum_R = sum_R + Image_Warp.at<cv::Vec3b>
					(lines_Normalized,collumns_Normalized)[2];
				}
			}
			Averages_Sum_B = (sum_B / 25);
			Averages_Sum_G = (sum_G / 25);
			Averages_Sum_R = (sum_R / 25);

			Neuron_Output_B1 = 255*pow(2.72,-pow((abs(Averages_Sum_B -
			Centres_Array[0])/(Centres_Array[15]/3.16)),2));
			Neuron_Output_G1 = 255*pow(2.72,-pow((abs(Averages_Sum_G -
			Centres_Array[5])/(Centres_Array[16]/3.16)),2));
			Neuron_Output_R1 = 255*pow(2.72,-pow((abs(Averages_Sum_R -
			Centres_Array[10])/(Centres_Array[17]/3.16)),2));
```

```
                Neuron_Output_BGR1 = ((Neuron_Output_B1 + Neuron_Output_G1
                + Neuron_Output_R1)/3);
                Output_Image1.at<uchar>(lines,collumns) =
                Neuron_Output_BGR1;
                Neuron_Output_B2 = 255*pow(2.72,-pow((abs(Averages_Sum_B -
                Centres_Array[1])/(Centres_Array[15]/3.16)),2));
                Neuron_Output_G2 = 255*pow(2.72,-pow((abs(Averages_Sum_G -
                Centres_Array[6])/(Centres_Array[16]/3.16)),2));
                Neuron_Output_R2 = 255*pow(2.72,-pow((abs(Averages_Sum_R -
                Centres_Array[11])/(Centres_Array[17]/3.16)),2));
                Neuron_Output_BGR2 = ((Neuron_Output_B2 + Neuron_Output_G2
                + Neuron_Output_R2)/3);
                Output_Image2.at<uchar>(lines,collumns) =
                Neuron_Output_BGR2;

                Neuron_Output_B3 = 255*pow(2.72,-pow((abs(Averages_Sum_B -
                Centres_Array[2])/(Centres_Array[15]/3.16)),2));
                Neuron_Output_G3 = 255*pow(2.72,-pow((abs(Averages_Sum_G -
                Centres_Array[7])/(Centres_Array[16]/3.16)),2));
                Neuron_Output_R3 = 255*pow(2.72,-pow((abs(Averages_Sum_R -
                Centres_Array[12])/(Centres_Array[17]/3.16)),2));
                Neuron_Output_BGR3 = ((Neuron_Output_B3 + Neuron_Output_G3
                + Neuron_Output_R3)/3);
                Output_Image3.at<uchar>(lines,collumns) =
                Neuron_Output_BGR3;

                Neuron_Output_B4 = 255*pow(2.72,-pow((abs(Averages_Sum_B -
                Centres_Array[3])/(Centres_Array[15]/3.16)),2));
                Neuron_Output_G4 = 255*pow(2.72,-pow((abs(Averages_Sum_G -
                Centres_Array[8])/(Centres_Array[16]/3.16)),2));
                Neuron_Output_R4 = 255*pow(2.72,-pow((abs(Averages_Sum_R -
                Centres_Array[13])/(Centres_Array[17]/3.16)),2));
                Neuron_Output_BGR4 = ((Neuron_Output_B4 + Neuron_Output_G4
                + Neuron_Output_R4)/3);
                Output_Image4.at<uchar>(lines,collumns) =
                Neuron_Output_BGR4;

                Neuron_Output_B5 = 255*pow(2.72,-pow((abs(Averages_Sum_B -
                Centres_Array[4])/(Centres_Array[15]/3.16)),2));
                Neuron_Output_G5 = 255*pow(2.72,-pow((abs(Averages_Sum_G -
                Centres_Array[9])/(Centres_Array[16]/3.16)),2));
                Neuron_Output_R5 = 255*pow(2.72,-pow((abs(Averages_Sum_R -
                Centres_Array[14])/(Centres_Array[17]/3.16)),2));
                Neuron_Output_BGR5 = ((Neuron_Output_B5 + Neuron_Output_G5
                + Neuron_Output_R5)/3);
                Output_Image5.at<uchar>(lines,collumns) =
                Neuron_Output_BGR5;

        }
}

bitwise_or(Output_Image1, Output_Image2, Output_Image);
bitwise_or(Output_Image, Output_Image3, Output_Image);
bitwise_or(Output_Image, Output_Image4, Output_Image);
bitwise_or(Output_Image, Output_Image5, Output_Image);
bitwise_and(Output_Image, mask7048, Output_Image);
if (limit == 1){
        cv::threshold(Output_Image,Output_Image,240,255,THRESH_BINARY);
}
morphologyEx(Output_Image,Output_Image,MORPH_CLOSE,morphing_kernel);
morphologyEx(Output_Image,Output_Image,MORPH_OPEN,morphing_kernel);
morphologyEx(Output_Image,Output_Image,MORPH_CLOSE,morphing_kernel);
//cut pixels after zero_sum (not road even if it seems so after number
```

```
        //of pixels = zero) > 0001111000011111101 >> 0001111000000000000000 >>>
        if (Cut == 1){
               int zero_sum = 2;
               for (int lines = 0; lines < 48; lines ++){
                      state_change = 0; sum_state_change = 0;
                      for (int collumns = 1; collumns < 70; collumns ++){
                             if (((Output_Image.at<uchar>
                             (lines,(collumns-1))) == 255) &&
                             (Output_Image.at<uchar>(lines,collumns)==0)){
                                    state_change = 1;
                                    sum_state_change = sum_state_change + 1;
                             }
                             if ((state_change == 1) &&
                             (Output_Image.at<uchar>(lines,collumns)==0)){
                                    sum_state_change = sum_state_change + 1;
                             }
                             if(sum_state_change > zero_sum){
                                    Output_Image.at<uchar>(lines,collumns)=0;
                             }
                      }
               }
        }
        ////////////////////////////////////////////////////////////////////////
/
        int step = 1, repetition = 0, starting_X=0, starting_Y=0,
        ending_X=0, ending_Y=0;
        while ((step == 1) && (repetition < 2)){
               for (int lines = Forward_Targeting_Distance;
               lines < 48; lines ++){
               //lines = 8(default) Affects the height of target line.
               //Closer to 48, the line gets smaller
                      for (int collumns = 69; collumns > 0; collumns --){
                             if ((Output_Image.at<uchar>(lines,collumns) > 200) &
                             (step ==1)){
                                    starting_X = collumns;
                                    starting_Y = lines;
                                    step ++ ;
                             }
                             else {
                                    repetition ++ ;
                             }
                      }
               }
        }
        repetition = 0;
        while ((step == 2) && (repetition < 2)){
               for (int collumns = 69; collumns > 0; collumns --){
                      if ((Output_Image.at<uchar>(40,collumns) > 240) &
                      (step ==2)){
                             ending_X = collumns;
                             ending_Y = 40;
                             step = 1;
                      }
                      else{
                             repetition = repetition + 1;
                      }
               }
        }
        if (((ending_X == 0) && (ending_Y == 0)) || (ending_X < 30)){
               ending_Y = 205/5, ending_X = 200/5;


        }
```

```
	//point (collumn,line) meaning (width,height). Begins up and left(0.0)
	linepoints[0] = starting_X, linepoints[1] = starting_Y,
	linepoints[2] = ending_X, linepoints[3] = ending_Y;
	line(Output_Image,Point(starting_X,starting_Y),
	Point(ending_X,ending_Y),(100),1,8);
	return Output_Image;
}
//Purpose
//	The intended line to be followed is calculated along with the degrees
that are
//	to be sent to the arduino about the steering angle to be attained.
//Software
//	Uses OPENCV library functions
void thread_target_line(int *linepoints, int *degrees, int *CarDirectionPOINTS){
	float lamda = 0, PI = 3.14159265;
	float Distance_from_Roadside = 0;
	lamda = (((47.0 -linepoints[1]) - (47.0 - linepoints[3]))/
	((0.0 + linepoints[0]) - (0.0 + linepoints[2])));
	degrees[0] = atan(lamda)*180/PI;
	if (lamda > 12){
		degrees[0] = 90;
	}
	if (lamda < -12){
		degrees[0] = 90;
	}
	if ((lamda < 0) && (lamda >= -12)){
		degrees[0] =180 + atan(lamda)*180/PI;
	}
	//It was found that 12 and -12 are close to 90 degrees
	//Distance from roadside
	Distance_from_Roadside = ((linepoints[2]*5) -
	CarDirectionPOINTS[2])*0.01;
	//cout << Distance_from_Roadside << endl;
	//
	if (Distance_from_Roadside >= 0.20){
		degrees[0] = degrees[0] - ((Distance_from_Roadside - 0.40) * 30);
	}
	//ideal distance stands for the distance that is attempded to be
	//between the vehicle and the roadside
	else if (Distance_from_Roadside < 0.20){
		degrees[0] = degrees[0] + ((Distance_from_Roadside) * 30);
	}
	// else if (Distance_from_Roadside < 0){
		// degrees[0]=degrees[0]-((Distance_from_Roadside-0.20)*50);
	// }
	//the x in function ((Distance_from_Roadside - ideal distance)) * x)
	//stands for the speed at which the limit line is aproached
	//(the course line angle gets bigger or lessens)

	if (isnan(degrees[0])){
		degrees[0] = 100;
	}
	if (degrees[0] > 120){
		degrees[0] = 120;
	}
	if (degrees[0] < 60){
		degrees[0] = 60;
	}
	//eleghos gia beltisti prosseggish grammhs
	//cout << lamda << endl;
	//cout << degrees[0] << endl;
}
```

```
//Purpose
//     Shows in windows the images obtained after analysis.
//Software
//     Uses OPENCV library functions
void thread_showimages(cv::Mat Input_Image,cv::Mat Warped_Image,
cv::Mat Output_Image, int *PointsArray, int *CarDirectionPOINTS, int *degrees){
       cv::Mat Output_Image_RES(240,350,CV_8UC3);
       cv::Mat Warped_output = Warped_Image.clone();
       float PI = 3.14159265;
       float Tan_of_Lines_Angle = tan((degrees[0]*PI)/180);
       //angle of line that the vehicle currently follows and the one
       //that is to be followed
       cv::resize(Output_Image,Output_Image_RES,cv::Size(350,240));
       float Xcoord_Driving_Line = (151 + 100/Tan_of_Lines_Angle);
       // +151 because the line does not start from 0
       line(Warped_output,Point((PointsArray[0]*5),(PointsArray[1]*5)),
       Point((PointsArray[2]*5),(PointsArray[3]*5)),(255),3,8);
       line(Warped_output,Point(CarDirectionPOINTS[0],CarDirectionPOINTS[1]),
       Point(CarDirectionPOINTS[2],CarDirectionPOINTS[3]),(100),1,8);
       line(Warped_output,Point(Xcoord_Driving_Line,CarDirectionPOINTS[1]),
       Point(CarDirectionPOINTS[2],CarDirectionPOINTS[3]),(100),1,8);
       cv::imshow("cam",Input_Image);
       cv::imshow("warp",Warped_output);
       cv::imshow("out",Output_Image_RES);
}

//Purpose
//     Sends data about throttle and steering actuation to arduino.
//Hardware
//     Connected arduino microcontroller
//Software
//     Uses wiringPi library functions and serial communication protocol.
void thread_com(int device, int *degrees){
       int drive = (degrees[1]*1000) + (180 - degrees[0]);
       string serial_out = to_string (drive);
       serialPuts(device,serial_out.c_str());
       serialPuts(device,"\n");
       serialFlush(device);

       char ser_rec; // ser_rec = serial recieve
       for (int i = 30; i >0; i--){ // i string length (30 full, 3 Trottle only)
              ser_rec = (serialGetchar(device));
              //if (ser_rec == 'x'){
                     //i = 0;
              //}
              cout << ser_rec;
       }

       serialFlush(device);
       cout << endl;
}

int main(int argc, char** argv){
       std::clock_t start;
       cv::Mat Input_Image(180,320,CV_8UC3);
       cv::VideoCapture cap;
       cv::Mat Output_Image(240,350,CV_8UC3);
       cap.open(0);
       cap.set(cv::CAP_PROP_FPS, 30);
       cap.set(cv::CAP_PROP_BRIGHTNESS, 40);
       cap.set(cv::CAP_PROP_CONTRAST,80);
       int Centres_Array[18];
       int degrees[2]; //(steering & throttle||| float =
```

```
   //=(Throttle*1000)+steer ||| (0->steering, 1->throttle)
   int linedraw[4], Cut = 0, limit = 0;
   int CarDirectionPOINTS[4] = {151,100,151,200};
   //points of current vehicle direction line
   int throttle_deg_int = 90; //idle
   int Forward_Targeting_Distance = 8;
   cap.set(cv::CAP_PROP_FRAME_WIDTH, 320);
   cap.set(cv::CAP_PROP_FRAME_HEIGHT, 180);
   cap.set(cv::CAP_PROP_BUFFERSIZE, 1);
               // map for perspective warp 2m
   cv::Point2f src_points[4];
   src_points[2] = cv::Point2f(131, 43);
   src_points[3] = cv::Point2f(167, 43);
   src_points[0] = cv::Point2f(128, 48);
   src_points[1] = cv::Point2f(171, 48);

   cv::Point2f dst_points[4];
   dst_points[2] = cv::Point2f(128, 6);
   dst_points[3] = cv::Point2f(170, 6);
   dst_points[0] = cv::Point2f(128, 48);
   dst_points[1] = cv::Point2f(171, 48);
   cv::Mat Warped_Image;
   cv::Mat trantable = cv::getPerspectiveTransform(src_points, dst_points);
         // recieve first picture for initiation
   cv::Mat First_Image;
   cv::Mat First_Image_warped(240,350,CV_8UC3);;
   Warped_Image = First_Image_warped.clone();
   cv::Mat mask7048(48,70,CV_8U);
   mask7048 = imread("70,180mask.jpg",0);
   if (!cap.isOpened())
{
    cout << "--(!)Error opening video capture" << endl;
}
   cap >> First_Image;
   cv::warpPerspective(First_Image, First_Image_warped, trantable,
   cv::Size(350, 240));
   Bell_Centres(First_Image_warped,Centres_Array);
   int mega2560 = serialOpen("/dev/ttyUSB0",38400);

   while(1){

         Input_Image = camera_work(cap);
         if (Input_Image.empty()){
               cout << "--(!)Error" << endl;
               break;
         }
         auto warped_video = async(std::launch::async, thread_warping,
         Input_Image, Warped_Image, trantable);
         Warped_Image = warped_video.get();
         auto out_picture = async(std::launch::async,NN_frame_analysis,
         Warped_Image, mask7048, Centres_Array, linedraw, Cut, limit,
         Forward_Targeting_Distance);
         Output_Image = out_picture.get();
         async(std::launch::async,thread_target_line, linedraw,
         degrees, CarDirectionPOINTS);
         async(std::launch::async,thread_showimages, Input_Image,
         Warped_Image, Output_Image, linedraw, CarDirectionPOINTS,degrees);
         async(std::launch::async,thread_com, mega2560, degrees);

         char c=(char)waitKey(25);
         if(c==27){ //press esc
               degrees[1] = 90;
               async(std::launch::async,thread_com, mega2560, degrees);
```

```
                break;
        }
        if(c==49){ //press 1
                Centres_Array[18] = {0};
                Bell_Centres(Warped_Image,Centres_Array);
                for (int i = 0; i < 15; i ++){
                        if (Centres_Array[i] <= 0){
                                Centres_Array[i] = 1 ;
                        }
                        if (Centres_Array[i] > 255){
                                Centres_Array[i] = 255;
                        }
                }
                for (int i = 0; i < 18; i++){
                        cout << Centres_Array[i] << " ";
                }
                cout << endl;
                //cout << Centres_Array[0] << endl;
                cout << "sample renewed" << endl;
        }
        if(c==50){ //press 2
                if (Cut == 0){
                        Cut = 1;
                }
                else{
                        Cut = 0;
                }
        }
        if(c==51){ //press 3
                if (limit == 0){
                        limit = 1;
                }
                else{
                        limit = 0;
                }
        }
        if(c==54){ //press 6 throttle
                throttle_deg_int = 100;
        }
        if(c==55){ //press 7 brake
                throttle_deg_int = 60;
        }
        if(c==56){ //press 8 neutral (90)
                throttle_deg_int = 90;
        }
        degrees[1] = throttle_deg_int;
        if(c==57){ // press 9 adjustment of targeting distance
                Forward_Targeting_Distance ++ ;
                if (Forward_Targeting_Distance >= 20){
                        Forward_Targeting_Distance = 8;
                }
        }
    }

    serialClose(mega2560); //>>ls -l /dev>>arduino(ttyUSB0 number)
    cap.release();
    destroyAllWindows();

    return 0;
}
```

Συνεχίζοντας, αναγράφεται ο κώδικας, γλώσσας C που εκτελείται στον μικροελεγκτή Arduino.

```
//Purpose
//  The microcontroller communicates via serial with the RPi4
//  and controls the servos for steering and throttle accordingly.
//  An extra layer of security is found in the face of four ultrasonic
//  proximity sensors that are poled and if the is an object closely,
//  the throttle is cut of and brakes are applied.
//  servo_sensor_v1.0
//  George Petroulias Meimetis
#include <Servo.h>
//front right  1
#define trigPin1 3
#define echoPin1 2
//front center 2
#define trigPin2 5
#define echoPin2 4
//front left   3
#define trigPin3 7
#define echoPin3 6
//top          4
#define trigPin4 10
#define echoPin4 11

long duration, distance, frontR, frontL, frontC, top;

Servo STEER;
Servo THROTTLE;

long posS = 90; //steering
long posT = 90; //throttle
int brake = 60;
long drive = 90090;
int metritisNB = 0; //counter Neutral Brake
int metritisT = 0; // counter Throttle
int posTtemp = 0;
int SENSE = 0, sensePREV = 0;

// array of values from which an average of steering angle is measured
// the smaller the number the less the previous values used
// v_MO is the number of values to be averaged
const int v_MO = 1;
int SteerTemp[v_MO];
void construct_array(){
  int i = 0;
  for (i = 0; i < v_MO; i++){
    SteerTemp[i] = 90;
  }
}
////////////////////////////////////////////////////////////////////////////
String inputString = "";         // a String to hold incoming data
boolean stringComplete = false;  // whether the string is complete

void setup() {
  Serial.begin(38400);
```

```
  // reserve 200 bytes for the inputString:
  inputString.reserve(200);
  STEER.attach(9);
  THROTTLE.attach(8);

  pinMode(trigPin1, OUTPUT);
  pinMode(trigPin2, OUTPUT);
  pinMode(trigPin3, OUTPUT);
  pinMode(trigPin4, OUTPUT);
  pinMode(echoPin1, INPUT);
  pinMode(echoPin2, INPUT);
  pinMode(echoPin3, INPUT);
  pinMode(echoPin4, INPUT);
  construct_array();
}

void loop() {
  // print the string when a newline arrives:
  sensePREV = SENSE;
  if (sensePREV == 0) {
    sensePREV = 4;
  }
  SENSE = SENSE + 1;

  if (stringComplete) {
    drive = inputString.toInt();

    SteerTemp[0] = drive % 1000;
    posT = drive / 1000;
  }
  ////////////////////////////////////////////
  for (int i = (v_MO-1); i >= 0; i--){
    SteerTemp[i+1] = SteerTemp [i];
  }
  posS = 0;
  for (int i = 0; i < v_MO; i++){
    posS = posS + SteerTemp[i];
  }
  posS = posS / v_MO;
  SteerTemp[0] = posS;
  ////////////////////////////////////////////
  if (SENSE == 1) {
    SonarSensor(trigPin1, echoPin1);
    frontR = distance;
  }
  if (SENSE == 2) {
    SonarSensor(trigPin2, echoPin2);
    frontC = distance;
  }
  if (SENSE == 3) {
    SonarSensor(trigPin3, echoPin3);
    frontL = distance;
  }
  if (SENSE == 4) {
    SonarSensor(trigPin4, echoPin4);
```

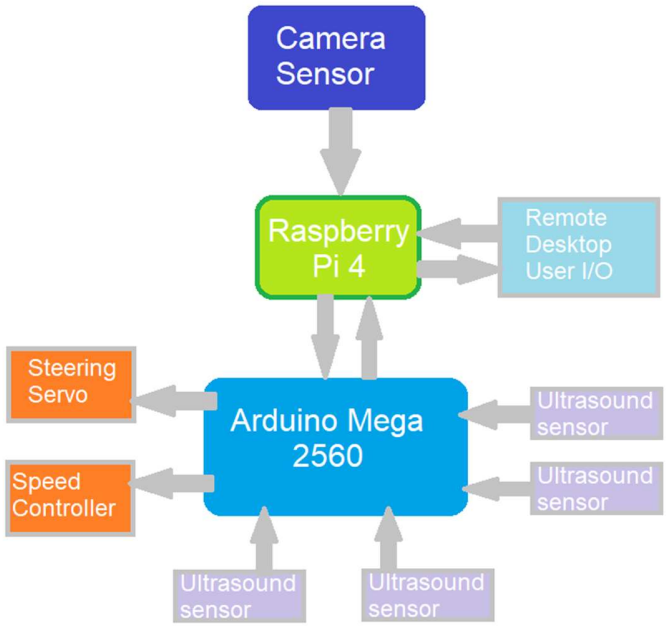
```
    top = distance;
  }

  if (distance < 100) {
    SENSE = sensePREV;
    if (posT != 90){
      posT = brake;
    }
  }
  if (SENSE > 3) {
    SENSE = 0;
  }
  inputString = "";
  stringComplete = false;
  STEER.write(posS);
  THROTTLE.write(posT);
  delay(15); // waits 15ms for the servo to reach the position


  if (posS < 100) {
    int temp = 0;
    Serial.print(temp);
  }
  Serial.print(posS);
  Serial.print(" < steer ||| throttle > ");
  if (posT < 100) {
    int temp = 0;
    Serial.print(temp);
  }
  Serial.println(posT);
  //Serial.println(" x");
}


void serialEvent() {
  while (Serial.available()) {
    //get the new byte:
    char inChar = (char)Serial.read();
    //add it to the inputString:
    inputString += inChar;
    //if the incoming character is a newline, set a flag so the main loop can
    //do something about it:
    if (inChar == '\n') {
      stringComplete = true;
    }
  }
}
void SonarSensor(int trigPin, int echoPin)
{
  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);
  duration = pulseIn(echoPin, HIGH, 8700);
```

```
  if (duration == 0) {
    duration = 100000;
  }
  distance = (duration / 2) / 29.1;
}
```



Γράφημα 7: Διαδρομές ανταλλαγής πληροφοριών μεταξύ των επιμέρους συστημάτων

Η επικοινωνία και η ανταλλαγή πληροφοριών μεταξύ των επιμέρους συστημάτων, είναι πολύ σημαντική για την επιτυχή πλοήγηση του οχήματος.

ΔΙΕΚΠΕΡΑΙΩΣΗ

Η περάτωση του πρακτικού μέρους του οχήματος έλαβε μέρος αφού πρώτα είχε γίνει επιλογή του τρόπου με τον οποίο θα γινόταν η διαχείριση της οπτικής πληροφορίας και βρέθηκε πως η μέθοδος που εξετάσθηκε ήταν ικανή να φέρει τα επιθυμητά αποτελέσματα. Τότε επιλέχθηκε ένα όχημα, το οποίο θα είχε αρκετά μεγάλες διαστάσεις και θα μπορούσε να αναπτύξει ταχύτητα αρκετή, ώστε να δικαιολογηθεί η κίνησή του σε τοπική οδό και όχι σε κάποιο είδος εξομοιωμένου περιβάλλοντος. Επίσης, κρίθηκε απαραίτητη η εξαιρετικά ταχεία και ακριβής απόκριση του συνόλου σε αλλαγές κατεύθυνσης και η δυνατότητα να συνεργάζεται με ευρύ πεδίο διαφορετικών μεθόδων για λήψη εντολών κίνησης. Αναγκαία κατέστει και η ύπαρξη κινητήρα, χωρίς ψήκτρες, ώστε να μειωθεί ο ηλεκτρομαγνητικός θόρυβος, ο οποίος επηρέαζε αρνητικά την ικανότητα μεταβίβασης δεδομένων μεταξύ των καίριων συστημάτων.

Το όχημα ήταν αρχικά κινούμενο από έναν δίχρονο κινητήρα εσωτερικής καύσης, ο οποίος αντικαταστάθηκε από ένα σύστημα κίνησης ηλεκτροκινητήρα και σχετικού ελεγκτή ταχύτητας, καθώς και από ευμεγέθη συσσωρευτή ρεύματος τύπου ιόντων λιθίου. Η κίνηση η οποία μεταδίδεται και στους τέσσερις ανεξάρτητους τροχούς, εξασφαλίζει την ευελιξία και άμεση απόκριση του συνόλου σε αλλαγές κατεύθυνσης.

ΑΠΟΤΕΛΕΣΜΑΤΑ

Το όχημα φαίνεται ικανό να προσπελάσει οδούς χρησιμοποιώντας την αναγραφόμενη μεθοδολογία. Η διαδικασία της εκπαίδευσης είναι εξαιρετικά σύντομη μα πρέπει να γίνεται τακτικά, ιδιαιτέρως υπό συνθήκες μη σταθερού φωτισμού και χρωματισμού του οδοστρώματος. Οι αλλαγές στον φωτισμό δύνανται να δημιουργήσουν καθυστέρηση στην επεξεργασία και ανάλυση της εικόνας, κάτι το οποίο επηρεάζει αρνητικά την ορθή λειτουργία του συνόλου και ενδέχεται να το θέσει εκτός πορείας.

Για την επιτυχή αλλαγή πορείας του οχήματος σε διασταυρούμενο δρόμο και προς την δεξιά αυτού κατεύθυνση είναι σημαντικό ο «στόχος» του οχήματος να έχει οριστεί σε μικρή απόσταση. Επίσης η αποκοπή του «πλοηγίσιμου δρόμου» μετά από συγκεκριμένο διάστημα που δεν αναγνωρίζεται, εμποδίζει την στρέψη του οχήματος προς τα δεξιά σε διασταυρούμενο δρόμο αλλά εμφανίζει μεγαλύτερη επιτυχία στην αποφυγή εμποδίων.

ΣΥΜΠΕΡΑΣΜΑΤΑ

Ο σκοπός της διπλωματικής αυτής εργασίας επετεύχθη. Αυτό σημαίνει πως νευρωνικό δίκτυο με νευρώνες ακτινικής βάσης, είναι ικανό μέσω διαδικασιών αναγνώρισης προτύπων να αναγνωρίσει χρωματικά το οδόστρωμα σε σχέση με εμπόδια, τα οποία διαφέρουν χρωματικά. Η αναγνώριση όμως μέσω χρώματος επιφέρει κινδύνους και περιορισμούς. Ένας τοίχος ή ένα διαχωριστικό κατασκευασμένο από σκυρόδεμα, μπορεί να είναι όμοιο χρωματικά με το οδόστρωμα. Ένα γκρίζο εμπόδιο ή κάποιος πεζός με ατυχή επιλογή ενδυμάτων μπορεί να μην αναγνωριστεί σαν εμπόδιο και το όχημα να μην σταματήσει εγκαίρως. Για αυτό κρίνεται απαραίτητο να υπάρχουν πρόσθετα συστήματα ασφάλειας όπως αισθητήρες εγγύτητας τύπου time of flight, lidar ή υπερήχων που μπορούν να διακρίνουν μη προσπελάσιμες επιφάνειες με πολύ μεγάλη ταχύτητα, σε απόσταση αρκετή ώστε να μειωθούν τυχών επικίνδυνες καταστάσεις.

Άλλες μέθοδοι οπτικής αναγνώρισης επίσης, είναι δυνατό να χρησιμοποιηθούν σε συνδυασμό με αλγορίθμους εντοπισμού ακμών και οπτικούς αισθητήρες υψηλότερης ευκρίνειας για ικανότερη εξαγωγή συμπερασμάτων από την λαμβανόμενη οπτική πληροφορία. Αυτό όμως, απαιτεί υψηλότερη επεξεργαστική ισχύ και ίσως ειδικού σκοπού τυπωμένα κυκλώματα, εξειδικευμένα στην ανάλυση πολυδιάστατων πινάκων μεγάλου μεγέθους.

Επίσης, θα μπορούσαν να χρησιμοποιηθούν αλγόριθμοι, έχοντες πρόσβαση σε μεγάλο αριθμό εικόνων με διασαφημένη την διάκριση μεταξύ δρόμου και εμποδίων, οι οποίοι θα είχαν τον έλεγχο της εκ νέου εκπαίδευσης του παρόντος δικτύου, προσφέροντας μεγαλύτερη ευελιξία, ανάμεσα σε ποικιλία φωτισμού και διαφορετικών περιοχών. Η ενσωμάτωση χαρτογράφησης σε συνεργασία με σύστημα εντοπισμού θέσης, θα μπορούσε να προσφέρει μεγαλύτερη κατανόηση του περιβάλλοντος και την ικανότητα να ακολουθηθεί ορισμένη πορεία.

Ακόμα, με την εξάπλωση του δικτύου πέμπτης γενιάς, όσων αφορά τις τηλεπικοινωνίες, το παρόν σύστημα θα μπορούσε να συμμετέχει σε μια ομάδα διασυνδεδεμένων οχημάτων, που πληροφορίες για τους δρόμους που πλοηγήθηκαν, διαμοιράζονται με τα υπόλοιπα οχήματα για καλύτερη διαχείριση πόρων και μείωση φόρτου των οδικών αξόνων, καθώς και για δημοσίευση κινδύνων και αστοχιών στο οδόστρωμα.

Ενδιαφέρουσες διορθώσεις που θα μπορούσαν να πραγματοποιηθούν, είναι η μεγαλύτερη παραλληλοποίηση του κώδικα για τον μικροϋπολογιστή RaspberryPi 4, ώστε να γίνεται ταχύτερη επεξεργασία των ληφθέντων δεδομένων. Επίσης, ίσως φαινόταν σημαντική η ενσωμάτωση lidar, που σε συνεργασία με την εικονοληψία θα μπορούσε να κάνει μια πολύ ικανή χαρτογράφηση με μεγάλη ανθεκτικότητα σε τυχόν θορύβους. Επιπλέον θα μπορούσε να συντελεστεί καλύτερη μοντελοποίηση του οχήματος με ακριβέστερο έλεγχο, που θα προσέφεραν μεγαλύτερη ταχύτητα μετακίνησης, χωρίς άσκοπες αλλαγές πορείας και κλυδωνισμούς.

Το μέρος του κώδικα που εκτελεί οπτική αναγνώριση, αρχικά συγγραφόταν σε γλώσσα PYTHON λόγο ευχρηστίας. Καθώς ο αλγόριθμος γινόταν περισσότερο πολύπλοκος, η μεταφορά του έργου σε C++ επέφερε μεγάλη επιτάχυνση στην εκτέλεση των διαδικασιών και τυγχάνει να προσφέρει αυξημένη συμβατότητα με το λειτουργικό σύστημα ANDROID [12], το οποίο ενσωματώνεται σε πολύ μεγάλο αριθμό φορητών συσκευών ανά το κόσμο, που διαθέτουν

αισθητήρες εικόνας και αρκετή επεξεργαστική ισχύ για την εκτέλεση των πολύπλοκων υπολογισμών που απαιτούνται, καθώς και αυξημένες ικανότητες συνδεσιμότητας. Μια τέτοια συσκευή, μπορεί να συνδεθεί διαδικτυακά για ταχύτατη ανταλλαγή δεδομένων, χρησιμοποιώντας το ανάλογο πρωτόκολλο, με τον υπολογιστή που ελέγχει την κίνηση κάποιου οχήματος και να του μεταφέρει τις εντολές που χρειάζονται για την πλοήγησή του, ώστε το όχημα να τις εκτελέσει ανάλογα. Αντίστοιχης λειτουργικότητας εφαρμογή, ονομαζόμενη openpilot, έχει δημιουργηθεί, παρεχόμενη από την comma.ai [13] και προσφέρει δυνατότητες αυτόματης πλοήγησης σε οχήματα των οποίων τα ηλεκτρονικά συστήματα δεν έχουν σχεδιαστεί ώστε να εκτελούν διαδικασίες αυτοοδήγησης.

Με την προώθηση της τεχνολογίας αυτοκινούμενων οχημάτων, εξελιγμένες μαθηματικοί [11] μέθοδοι δύνανται να αναπτυχθούν για καλύτερη αναγνώριση οδοστρώματος ή μη. Ο συνδυασμός τους με τον εντοπισμό ευάλωτων χρηστών του δρόμου ή τυχών εμποδίων, κάνοντας χρήση οπτικής πληροφορίας αναλυόμενης με ανώτερα, πιο διασαφημένα κριτήρια και λειτουργώντας συνδυαστικά με βάσεις δεδομένων, που διαρκώς ανανεώνονται ανταλλάσσοντας μεγάλο όγκο πληροφοριών με πολύ υψηλές ταχύτητες, θα μπορέσουν να επιτευχθούν βαθμοί αυτονομίας επιπέδου 5 για το σύνολο του οδικού δικτύου. Έτσι θα υπάρξει η ικανότητα ασφαλούς προσπέλασης οποιασδήποτε οδού, υπό κάθε πιθανή συνθήκη. Ίσως ακόμα φανεί ικανή η συνύπαρξη οχημάτων, οδηγούμενων από τον άνθρωπο-χρήστη και πλήρως αυτοκινούμενων.

Βιβλιογραφία

[1] J. Cicolani, Beginning Robotics with Rasberry Pi and Arduino Using Python and OpenCV, Pflugerville, Texas, USA: Apress Media LLC, 2018, p. 297.

[2] Ν. Αναστάσιος, Σημειώσεις Ευφυούς Ελέγχου, ΤΕΙ Πειραιά,Σχολή Τεχνολογικών Εφαρμογών, Τμήμα Μηχανικών Αυτοματισμού ΤΕ, Τομέας Βιομηχανικής Πληροφορικής, 2013, p. 92.

[3] J. T. /. E. Granath, «https://www.mes-insights.com/,» 11 03 2020. [Ηλεκτρονικό]. Available: https://www.mes-insights.com/amp/the-5-levels-of-autonomous-driving-explained-a-912868/?cmp=go-ta-art-trf-MES_DSA-20200217&gclid=CjwKCAiAuoqABhAsEiwAdSkVVBzteMvs2GJtUw9-YPCMmL3LtN_VjY61Y3ZI0lgogzW27umcMQ62PxoCQlEQAvD_BwE. [Πρόσβαση 10 01 2021].

[4] K. Bimbraw, « "Autonomous cars: Past, present and future a review of the developments in the last century, the present scenario and the expected future of autonomous vehicle technology,",» 12th International Conference on Informatics in Control, Automation and Robotics (ICINCO), Colmar, France, 2015.

[5] J. &. H. C. &. d. l. E. A. &. A. J. Collado, *Detection and classification of road lanes with a frequency analysis,* Las Vegas, NV, USA: IEEE Intelligent Vehicles Symposium, Proceedings, 2005.

[6] Κ. ΝΤΟΥΝΗΣ ΑΝΑΣΤΑΣΙΟΣ, *ΣΗΜΕΙΩΣΕΙΣ ΕΥΦΥΟΥΣ ΕΛΕΓΧΟΥ και ΥΠΟΛΟΓΙΣΤΙΚΗΣ ΝΟΗΜΟΣΥΝΗΣ,* ΠΕΙΡΑΙΑΣ: ΤΕΙ Πειραιά, , Σχολή Τεχνολογικών Εφαρμογών, Τμήμα Μηχανικών Αυτοματισμού ΤΕ , Τομέας Βιομηχανικής Πληροφορικής.

[7] R. (. R. Young-Woo Seo, *Detection and Tracking of Boundary of Unmarked Roads,* Pittsburgh, PA 15213 USA: Carnegie Mellon University, 2014.

[8] «https://docs.opencv.org/,» 10 1 2021. [Ηλεκτρονικό]. Available: https://docs.opencv.org/master/da/d22/tutorial_py_canny.html.

[9] «https://docs.opencv.org,» 10 1 2021. [Ηλεκτρονικό]. Available: https://docs.opencv.org/3.4/d2/d2c/tutorial_sobel_derivatives.html.

[10] Π. Σπυρίδων, «Ψηφιακή Επεξεργασία Εικόνας & Υπολογιστική,» ΠΜΣ Τεχνολογίες και Διοίκηση Πληροφοριακών και Επικοινωνιακών Συστημάτων.

[11] A. B. K. K. Neethu Johna, «A Reliable Method for Detecting Road Regions from a Single Image Based on Color Distribution and Vanishing Point Location,» Elsevier, 2015.

[12] «OpenCV,» OPENCV, 2021. [Ηλεκτρονικό]. Available: https://opencv.org/android/. [Πρόσβαση 7 1 2021].

[13] «comma.ai,» comma.ai, 2020. [Ηλεκτρονικό]. Available: https://comma.ai. [Πρόσβαση 7 1 2021].