ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ

ΣΧΟΛΗ ΤΕΧΝΟΛΟΓΙΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ
ΤΜΗΜΑ ΨΗΦΙΑΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Υλοποίηση Τηλεκατευθυνόμενου Αυτόνομου Μικρο-οχήματος Μηχανικής Μάθησης με Ενσωματωμένο Μικροελεγκτή Raspberry Pi

Γεώργιος Ι. Βιέννας

Επιβλέπων Καθηγητής:
Απόστολος Μηλιώνης, Αναπληρωτής Καθηγητής

ΠΕΙΡΑΙΑΣ

ΙΟΥΛΙΟΣ 2021

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Υλοποίηση Τηλεκατευθυνόμενου Αυτόνομου Μικρο-οχήματος Μηχανικής Μάθησης με Ενσωματωμένο Μικροελεγκτή Raspberry Pi

Γεώργιος Βιέννας

Α.Μ.: Ε15017

Πίνακας περιεχομένων

1.ΕΙΣΑΓΩΓΗ

1.1 Περίληψη

Η παρούσα πτυχιακή εργασία παρουσιάζει όλη την διαδικασία για την μελέτη και υλοποίηση ενός αυτόνομου μικρο-οχήματος μηχανικής μάθησης με χρήση ενός μικροελεγκτή. Η εργασία αποτελείται από 3 κύρια σκέλη: πρώτον το θεωρητικό υπόβαθρο που αφορά την μηχανική μάθηση κυρίως για επεξεργασία και κατηγοριοποίηση εικόνων, δεύτερον την υλοποίηση του μικρο-οχήματος που ελέγχεται από τον μικροελεγκτή και τρίτον τον σχεδιασμό και την ανάπτυξη του πηγαίου κώδικα για τον έλεγχο του οχήματος.

1.2 Περιγραφή του προβλήματος

Με την εξέλιξη της τεχνολογίας και την διαρκή αύξηση των απαιτήσεων ασφαλείας στα αυτοκίνητα και στους δρόμους, γίνεται όλο ένα και πιο προφανές ότι ο ανθρώπινος παράγοντας είναι η πιο επικίνδυνη μεταβλητή σε θέμα ασφάλειας στο οδόστρωμα. Όσο ασφαλές και να γίνει ένα αυτοκίνητο εξακολουθεί να το ελέγχει ένας άνθρωπος που δεν είναι μια τέλεια μηχανή, κάνει λάθη, αποσπάται εύκολα και κουράζεται. Έτσι λοιπόν πρέπει να βρούμε έναν εναλλακτικό τρόπο πλοήγησης των οχημάτων ο οποίος δεν εξαρτάται από κάποια ανθρώπινη παρέμβαση για την ασφαλή πλοήγηση του οχήματος στον προορισμό του. Με την εξέλιξη της τεχνολογίας και το μειωμένο κόστος για ισχυρά υπολογιστικά συστήματα, γίνεται ολοένα και πιο εφικτό το να έχουμε αυτοκίνητα με την υπολογιστική δύναμη για αυτόνομη πλοήγηση με την χρήση αλγορίθμων μηχανικής μάθησης. Οι αλγόριθμοι αυτοί μπορούν να εκπαιδευτούν και να μάθουν όχι μόνο να πλοηγούν ένα αυτοκίνητο στον προορισμό του αλλά και να προβλέπουν και να αποφεύγουν ατυχήματα, με την χρήση πολλών καμερών και ραντάρ που είναι τοποθετημένα στο αυτοκίνητο. Είναι προφανές ότι το μέλλον της αυτοκίνησης είναι η αυτόνομη οδήγηση για την καλύτερη ασφάλεια οδηγών, επιβατών και πεζών.

1.3 Σκοπός και στόχος της εργασίας

Η εργασία αυτή έχει δύο βασικούς σκοπούς. Ο πρώτος είναι η υλοποίηση ενός μικρο-οχήματος το οποίο είναι ικανό να πλοηγηθεί αυτόνομα με την χρήση μιας κάμερας, ενός μικροελεγκτή και ενός αλγόριθμου μηχανικής μάθησης. Έτσι γίνεται φανερό ότι η αυτόνομη οδήγηση είναι κάτι εφικτό σε συμβατικά οχήματα. Ο δεύτερος σκοπός είναι η ανάπτυξη του αλγορίθμου μηχανικής μάθησης και η επίδειξη της ικανότητας του να μάθει τι είναι δρόμος και να πάρει αποφάσεις για την ορθή κατεύθυνση του οχήματος σε διαδρομές τις οποίες ο αλγόριθμος δεν έχει ξανά δει.

Στόχος της εργασίας είναι η υλοποίηση του μικρο-οχήματος, η ανάπτυξη του αλγορίθμου και ο συνδυασμός αυτών των δυο για τον έλεγχο και πλοήγηση του οχήματος σε μια πίστα όπου τα όρια της αναπαρίστανται από γραμμές στο δάπεδο. Για να είναι προφανές ότι ο αλγόριθμος δεν απομνημονεύει την διαδρομή και απλά χρησιμοποιεί τις εντολές που έχει αποθηκεύσει, η πίστα στην οποία θα εκπαιδευτεί ο αλγόριθμος θα είναι διαφορετική από αυτή που θα πρέπει να πλοηγηθεί αυτόνομα.

2.ΘΕΩΡΗΤΙΚΟ ΥΠΟΒΑΘΡΟ – ΜΗΧΑΝΙΚΗ ΜΑΘΗΣΗ

2.1 Τι είναι η μηχανική μάθηση

Η μηχανική μάθηση αποτελεί ένα υποπεδίο της τεχνητής νοημοσύνης και γενικότερα της επιστήμης των υπολογιστών. Μια από τις πρώτες αναφορές στην μηχανική μάθηση ήταν το 1959, όπου ο Άρθουρ Σάμουελ ορίζει την μηχανική μάθηση ως "Πεδίο μελέτης που δίνει στους υπολογιστές την ικανότητα να μαθαίνουν, χωρίς να έχουν ρητά προγραμματιστεί". Οι αλγόριθμοι μηχανικής μάθησης διαφέρουν σε σχέση με τους υπόλοιπους αλγόριθμους διότι έχουν την δυνατότητα να μαθαίνουν από τα δεδομένα. Οι "συμβατικοί" αλγόριθμοι με την χρήση κανόνων και δεδομένων μπορούν να δώσουν αποτελέσματα, ενώ οι αλγόριθμοι μηχανικής μάθησης με την χρήση δεδομένων και αποτελεσμάτων μπορούν να βγάλουν κανόνες. Έτσι αυτοί οι αλγόριθμοι μας δίνουν την δυνατότητα να βρούμε μοτίβα και κανόνες στα δεδομένα, τα οποία ένας άνθρωπος δύσκολα θα έβρισκε.

2.2 Κατηγορίες και είδη αλγορίθμων μηχανικής μάθησης

Οι αλγόριθμοι μηχανικής μάθησης χωρίζονται συνήθως σε 3 βασικές κατηγορίες.

Επιτηρούμενη Μάθηση: αλγόριθμοι στους οποίους δίνουμε τα δεδομένα και τα επιθυμητά αποτελέσματα και στόχος τους είναι να παράγουν γενικευμένους κανόνες οι οποίοι χαρτογραφούν τα δεδομένα εισόδου με τα επιθυμητά αποτελέσματα χωρίς την παρέμβαση ανθρώπου. Τα νευρωνικά δίκτυα είναι ο πιο διαδεδομένος αλγόριθμος επιτηρούμενης μάθησης.

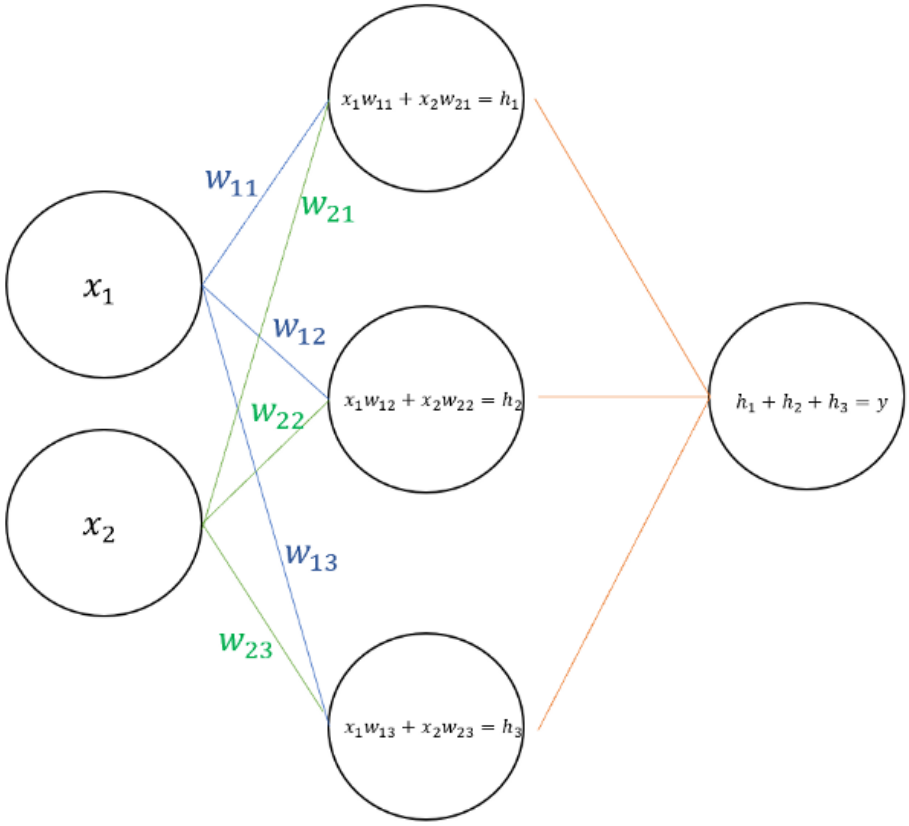
Μη Επιτηρούμενη Μάθηση: αλγόριθμοι στους οποίους δίνουμε δεδομένα χωρίς κάποιο χαρακτηρισμό και πρέπει να βρουν κάποια δομή στα δεδομένα. Χωρίς να έχουν κάποια εμπειρία πάνω στα δεδομένα αυτοί οι αλγόριθμοι μπορούν να βρουν μοτίβα και κρυμμένες συσχετίσεις σε πολύπλοκα δεδομένα. Τέτοιοι αλγόριθμοι χρησιμοποιούνται για ομαδοποίηση δεδομένων και ανίχνευση ανωμαλιών. Μερικοί γνωστοί αλγόριθμοι που ανήκουν σε αυτή την κατηγορία είναι η " συσταδοποίηση κ-μέσων" και η "ανάλυση κύριων συνιστωσών".

Ενισχυτική μάθηση: αλγόριθμοι οι οποίοι μαθαίνουν να φτάνουν σε έναν στόχο ή αποτέλεσμα μέσω επιβράβευσης και ποινής. Συνήθως τέτοιο αλγόριθμοι χρησιμοποιούνται σε διαδραστικά περιβάλλοντα όπως παιχνίδια ή προσομοιώσεις όπου υπάρχει ένας διακριτός στόχος, κάθε βήμα του αλγορίθμου επιβραβεύεται ή του επιβάλλεται ποινή αν με αυτό το βήμα έφτασε πιο κοντά στον στόχο ή όχι.

2.3 Τεχνητά νευρωνικά δίκτυα

Τα τεχνητά νευρωνικά δίκτυα είναι εμπνευσμένα από τα φυσικά νευρωνικά δίκτυα που βρίσκονται στον εγκέφαλο οποιουδήποτε ανθρώπου αλλά δεν έχουν σχέση στο πώς λειτουργούν. Τα δίκτυα αυτά αποτελούνται από επίπεδα κόμβων ή αλλιώς νευρώνων τα οποία είναι συνδεδεμένα μεταξύ τους, ένα δίκτυο λέγεται πυκνά συνδεδεμένο όταν κάθε κόμβος ενός επιπέδου δίνει την πληροφορία του σε κάθε κόμβο του επόμενου επιπέδου. Οι συνδέσεις μεταξύ των κόμβων

αναπαριστούν τα βάρη της σύνδεσης και είναι οι τιμές τις οποίες μαθαίνει το δίκτυο μέσω της εκπαίδευσης.



Εικόνα 2. 1 Αρχιτεκτονική βαθιά συνδεδεμένου νευρωνικού δικτύου

Το επίπεδο εισαγωγής (πρώτο επίπεδο) κόμβων αναπαριστά τα δεδομένα τα οποία λαμβάνει το δίκτυο σαν είσοδο και τα προωθεί στο επόμενο επίπεδο κόμβων. Οι κόμβοι του κρυφού επιπέδου (δεύτερο επίπεδο) αναπαριστούν το άθροισμα των τιμών του προηγούμενου επιπέδου επί των συντελεστών σύνδεσης των κόμβων. Οι κόμβοι του επιπέδου εξόδου (τρίτο επίπεδο) αναπαριστούν το αποτέλεσμα, συνήθως είναι ίσοι με τον αριθμό των κλάσεων στις οποίες θέλουμε να αντιστοιχίσουμε τα δεδομένα εισόδου ή υπάρχει μοναδικός κόμβος, εάν το αποτέλεσμα είναι συνεχής τιμή. Λειτουργούν όπως το κρυφό επίπεδο αλλά δεν προωθούν πουθενά τα αποτελέσματα των κόμβων.

Ο κόμβος με τον μεγαλύτερο αριθμό στο επίπεδο εξόδου αναπαριστά την κλάση στην οποία το δίκτυο θεωρεί ότι αντιστοιχούν τα δεδομένα. Κάθε κόμβος, όμως, πλην αυτών που ανήκουν στο επίπεδο εισόδου, έχει και μια συνάρτηση ενεργοποίησης από την οποία περνάει το άθροισμα των γινομένων των προηγούμενων κόμβων επί των συντελεστών πριν φτάσει στον κόμβο αυτό. Οι συναρτήσεις αυτές είναι είτε δυαδικές, δηλαδή αντιστοιχούν το αποτέλεσμα σε 0 για αρνητικές τιμές ή 1 για θετικές, είτε γραμμικές όπου πολλαπλασιάζουν το αποτέλεσμα με έναν σταθερό όρο ή μη γραμμικές όπως είναι η λογιστική σιγμοειδής συνάρτηση ή η υπερβολική εφαπτομένη. Ένα τεχνητό νευρωνικό δίκτυο ονομάζεται βαθύ νευρωνικό δίκτυο όταν έχει πάνω από ένα κρυφά επίπεδα.

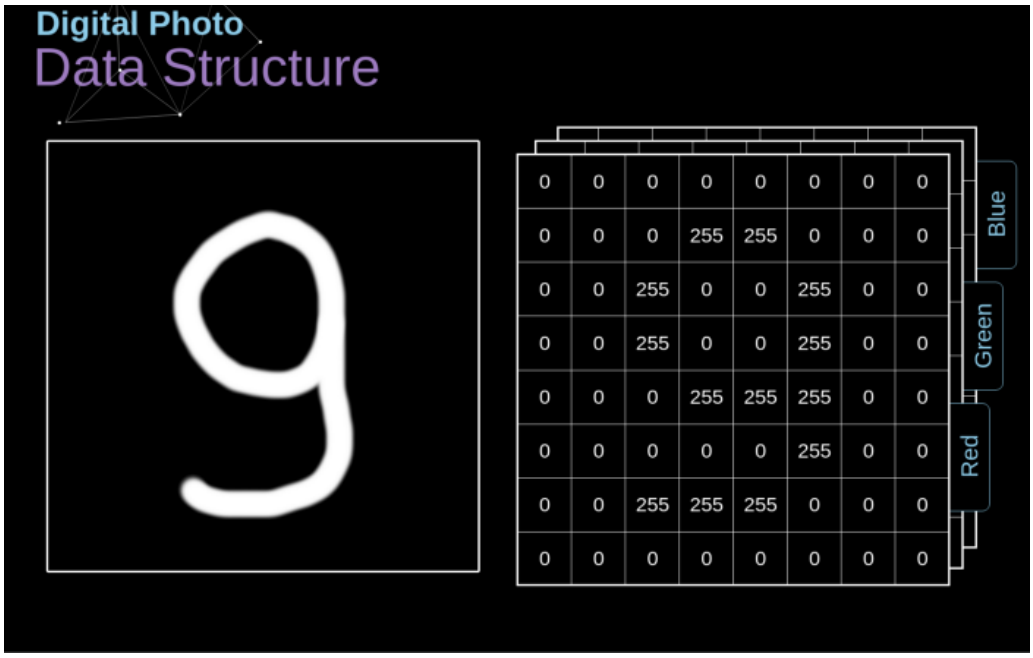
2.4 Συνελικτικά νευρωνικά δίκτυα

Τα συνελικτικά νευρωνικά δίκτυα είναι μια εξέλιξη των τεχνητών νευρωνικών δικτύων που αναφέραμε στην προηγούμενη υποενότητα. Τα δίκτυα αυτά χρησιμοποιούνται κυρίως για κατηγοριοποίηση εικόνων και ήχου διότι μπορούν να αποσπάσουν πληροφορίες μέσα από τα δεδομένα ανεξάρτητα από το που βρίσκονται μέσα στην εικόνα ή ήχο με την χρήση φίλτρων (πυρήνων). Αυτό που διαφέρει τα συνελικτικά δίκτυα με τα απλά είναι τα δύο βασικά επίπεδα της συνέλιξης και της συνάθροισης. Το επίπεδο της συνέλιξης περιέχει τα φίλτρα με συντελεστές τους οποίους ο αλγόριθμος μαθαίνει μέσω της εκπαίδευσης. Το κάθε φίλτρο είναι ένας πίνακας μικρότερος της εικόνας, το οποίο σαρώνει την εικόνα και αλλάζει τις τιμές του πίνακα της εικόνας με την χρήση του πολλαπλασιασμού. Έτσι το κάθε φίλτρο δημιουργεί έναν νέο πίνακα. Στην συνέχεια αυτός ο πίνακας περνάει από το επίπεδο της συνάθροισης όπου τα στοιχεία του πίνακα ομαδοποιούνται σε γειτονιές και γίνεται σύμπτυξη των στοιχείων της κάθε γειτονιάς με την χρήση του μέσου, ελάχιστου ή μέγιστου στοιχείου. Έτσι το μέγεθος της εικόνας μειώνεται και έχουμε έναν πίνακα με "συμπυκνωμένη" πληροφορία τον οποίο είτε περνάμε σε ένα νευρωνικό δίκτυο είτε σε άλλο ένα επίπεδο συνέλιξης και συνάθροισης.

Πιο συγκεκριμένα οι εικόνες που χρησιμοποιούνται ως δεδομένα εισόδου είναι συνήθως στην μορφή τρισδιάστατων πινάκων, όπου οι δύο διαστάσεις αναπαριστούν το ύψος και το πλάτος της εικόνας, ενώ η τρίτη διάσταση αναπαριστά τα τρία βασικά χρώματα που αποτελούν την εικόνα (κόκκινο, μπλε, πράσινο). Κάθε στοιχείο του πίνακα αυτού παίρνει μια τιμή από 0 έως 255 που είναι το πόσο έντονο είναι αυτό το χρώμα στην αντίστοιχη περιοχή του πίνακα.

$$G[m,n] = (f * h)[m,n] = \sum_j \sum_k h[j,k] f[m-j, n-k]$$

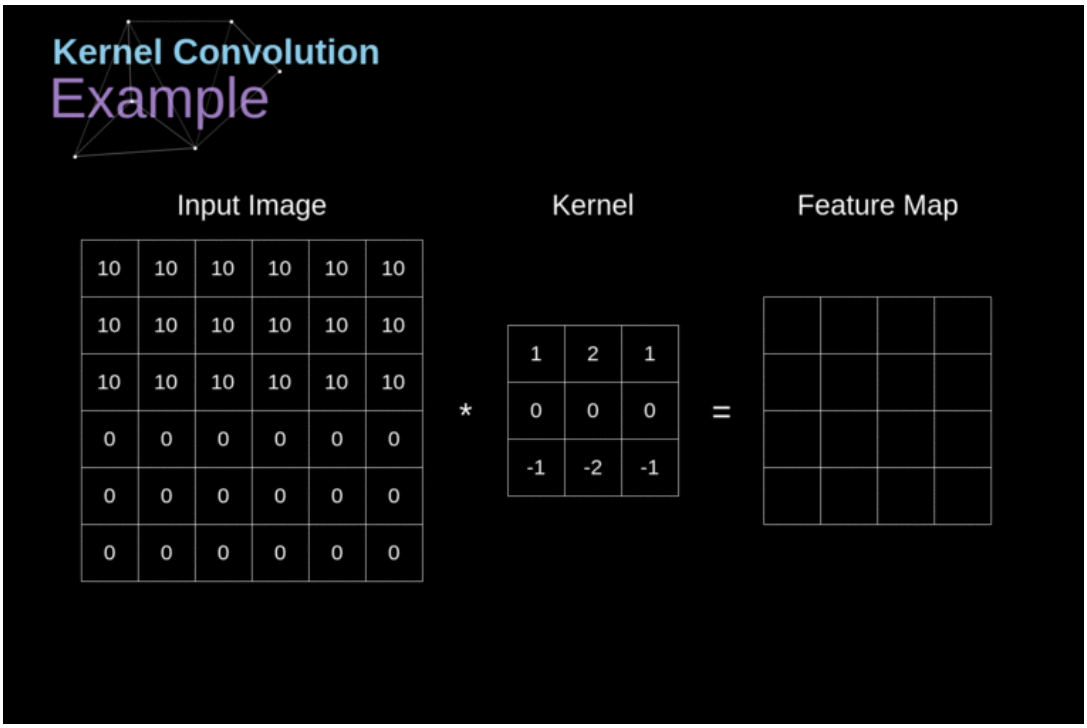
Εικόνα 2. 2 Μαθηματική συνάρτηση συνελικτικού επιπέδου



Εικόνα 2. 3 Δομή δεδομένων ψηφιακής εικόνας

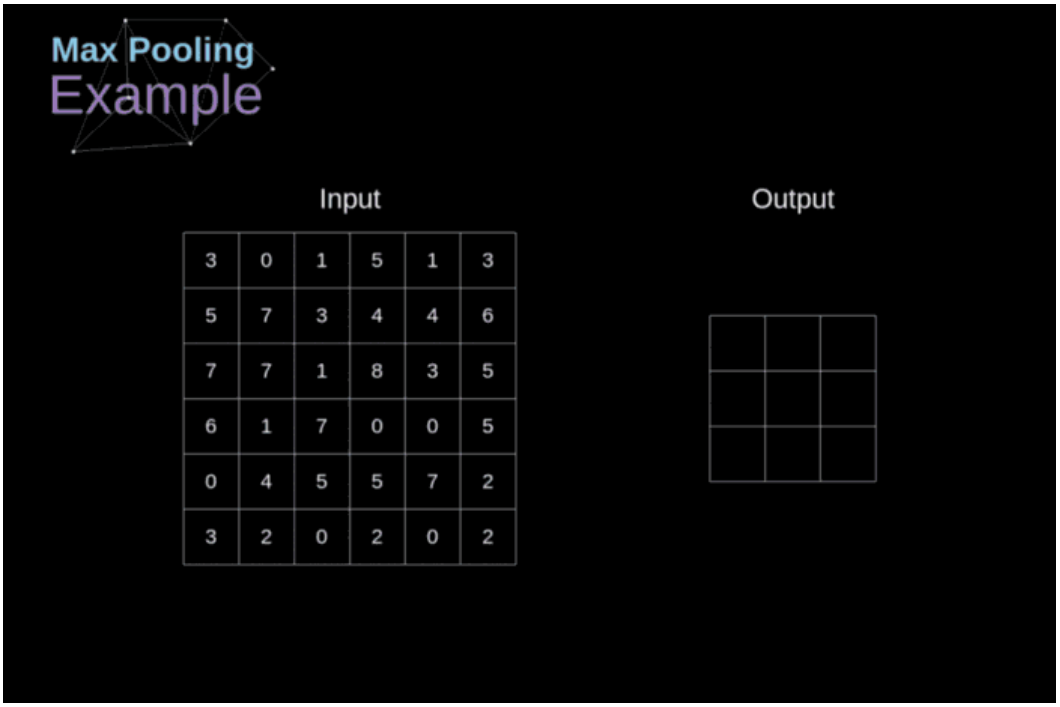
Όταν το συνελικτικό επίπεδο του αλγορίθμου δεχτεί μια εικόνα σαν είσοδο, όπως είπαμε, σαρώνει την εικόνα με το φίλτρο και δημιουργεί έναν νέο πίνακα. Ο νέος πίνακας προκύπτει από τον πολλαπλασιασμό στοιχείο επί στοιχείο των δεδομένων του πίνακα- εικόνα που βρίσκονται στο παράθυρο σάρωσης με τα στοιχεία του φίλτρου. Τα στοιχεία που προκύπτουν αθροίζονται και το άθροισμα αποτελεί ένα στοιχείο του νέου πίνακα. Στην εικόνα 2.2 φαίνεται ο μαθηματικός τύπος όπου το γράμμα f αντιστοιχεί στο παράθυρο του πίνακα εισόδου και το γράμμα h στο φίλτρο, τα γράμματα m και n αντιστοιχούν στις γραμμές και στήλες αντίστοιχα του τελικού πίνακα.

Στην κινούμενη εικόνα 2.1 φαίνεται η διαδικασία της σάρωσης και συνέλιξης της εικόνας. Το μωβ τετράγωνο είναι το παράθυρο σάρωσης στην εικόνα εισόδου, ο πίνακας τρία επί τρία είναι το φίλτρο και το μπλε τετράγωνο δείχνει την θέση του αποτελέσματος στον πίνακα που προκύπτει από την πράξη του πολλαπλασιασμού και της πρόσθεσης.



Κινούμενη Εικόνα 2. 1 Διαδικασία συνέλιξης

Στον πίνακα που προκύπτει από την συνέλιξη συνήθως κάνουμε ομαδοποίηση γειτόνων. Γίνεται ομαδοποίηση γειτονικών στοιχείων του πίνακα. Συνήθως το μέγεθος είναι δύο επί δύο, ανάλογα με το μέγεθος της εικόνας και έτσι προκύπτει ένας τελικός πίνακας - εικόνα. Από αυτή την ομαδοποίηση συνήθως υπερτερεί το μέγιστο στοιχείο καθώς σκοπός είναι να αναδειχθούν τα έντονα στοιχεία της εικόνας όπως είναι οι γραμμές. Στην κινούμενη εικόνα 2.2 φαίνεται η διαδικασία της ομαδοποίησης γειτόνων με βάση το μέγιστο στοιχείο.



Κινούμενη Εικόνα 2. 2 Ομαδοποίηση μέγιστου γείτονα

2.5 Εκπαίδευση νευρωνικών δικτύων

Για την εκπαίδευση των νευρωνικών δικτύων είναι απαραίτητη η δημιουργία ενός συνόλου δεδομένων (dataset) όπου αποτελείται από τα δεδομένα και τις ετικέτες που αντιστοιχούν στην κατηγορία που ανήκουν τα δεδομένα. Το σύνολο αυτό χωρίζεται σε δύο υποσύνολα. Το πρώτο υποσύνολο ονομάζεται σύνολο εκπαίδευσης και το δεύτερο ονομάζεται σύνολο επαλήθευσης. Το σύνολο επαλήθευσης συνήθως είναι το 20% με 40% του αρχικού συνόλου και ο αλγόριθμος δεν εκπαιδεύεται ποτέ πάνω στα δεδομένα του.

Η εκπαίδευση του αλγορίθμου γίνεται με ανάποδο πέρασμα των δεδομένων (backpropagation) στο μοντέλο. Αρχικά το δίκτυο δίνει τυχαίες τιμές στις μεταβλητές του, έπειτα κάνει την διαδικασία της πρόβλεψης με ένα από τα δεδομένα εκπαίδευσης και φτάνει σε μια πρόβλεψη. Συγκρίνοντας την πρόβλεψη με την αναμενομένη τιμή των δεδομένων (ετικέτα) κάνει βήματα ανάποδα στο δίκτυο, δηλαδή ξεκινάει από το τελευταίο επίπεδο των κόμβων και με την χρήση παραγώγων αλλάζει τις τιμές στις μεταβλητές που είναι οι συνδέσεις μεταξύ των κόμβων. Ανάλογα με το πόσο επηρέασαν το τελικό αποτέλεσμα θα μειωθούν ή θα αυξηθούν.

Η διαδικασία της εκπαίδευσης αποτελείται από βήματα, σε κάθε βήμα ο αλγόριθμος εκπαιδεύεται πάνω στα δεδομένα εκπαίδευσης και παράλληλα αξιολογείται πάνω σε αυτά τα δεδομένα. Τα δύο βασικά σημεία αξιολόγησης είναι η ακρίβεια, δηλαδή πόσες φορές κατηγοριοποίησε κάτι ορθά και το δεύτερο είναι η απώλεια που μετριέται με την χρήση ενός αλγορίθμου απώλειας, συνήθως αναπαριστά το ποσό μακριά ήταν η πρόβλεψη του αλγορίθμου σε σχέση με την κατηγορία ή την τιμή που ανήκουν τα δεδομένα. Στο τέλος του κάθε βήματος ο αλγόριθμος καλείται να κατηγοριοποιήσει τα δεδομένα επαλήθευσης και προκύπτουν τα δύο βασικά σημεία αξιολόγησης του, η ακρίβεια και η απώλεια. Με βάση αυτές τις τέσσερις μετρήσεις μπορούμε να αξιολογήσουμε τον αλγόριθμο και να καταλάβουμε αν κάνει υπερβολική προσαρμογή (over fitting), ελάχιστη

προσαρμογή (under fitting) ή αν εξελίσσεται ομαλά. Αν η ακρίβεια εκπαίδευσης είναι πολύ μεγαλύτερη σε σχέση με την ακρίβεια επαλήθευσης τότε έχουμε υπερβολική προσαρμογή, ο αλγόριθμος δεν θα αποδίδει καλά σε πιο γενικευμένα δεδομένα και θα μαθαίνει να κατηγοριοποιεί μόνο τα δεδομένα που έχει ήδη δει. Αν η ακρίβεια εκπαίδευσης και επαλήθευσης είναι σχετικά μικρές και δεν βελτιώνονται σε κάθε βήμα τότε έχουμε ελάχιστη προσαρμογή, δηλαδή ο αλγόριθμος δυσκολεύεται να ξεχωρίσει τα δεδομένα και χρειαζόμαστε έναν πιο πολύπλοκο αλγόριθμο ή περισσότερα και πιο καθαρά δεδομένα. Αν όμως η ακρίβεια εκπαίδευσης και επαλήθευσης είναι σχεδόν ίδιες και ο αλγόριθμος σε κάθε βήμα βελτιώνει και τις δυο τιμές, μας δείχνει ότι η εκπαίδευση έχει αποτελέσματα. Ενώ η ακρίβεια θέλουμε να ανεβαίνει, η απώλεια θέλουμε να κατεβαίνει. Συνήθως η απώλεια χρησιμοποιείται αντί της ακρίβειας όταν ο αλγόριθμος προβλέπει πραγματικές τιμές και όχι κατηγορίες χωρίς όμως να σημαίνει ότι δεν είναι χρήσιμη στην περίπτωση των κλάσεων.

2.6 Χρήση αλγορίθμων μηχανικής μάθησης

Οι αλγόριθμοι μηχανικής μάθησης μπορούν να χρησιμοποιηθούν στην λύση πολλών προβλημάτων όπως: ανίχνευση ελαττωματικών προϊόντων σε μια γραμμή παραγωγής, συγγραφή κειμένου, μετάφραση φωνής σε πραγματικό χρόνο και για αυτόνομη πλοήγηση αυτοκινήτου. Πιο συγκεκριμένα, για την αυτόνομη πλοήγηση ενός αυτοκινήτου χρειάζονται παραπάνω από ένας αλγόριθμοι οι οποίοι παράγουν δεδομένα τα οποία το αυτοκίνητο μπορεί να χρησιμοποιήσει για να κινηθεί στον προορισμό του με ασφάλεια. Για παράδειγμα ένας αλγόριθμος είναι υπεύθυνος για την αναγνώριση πινακίδων και φαναριών μέσω της κυρίας μπροστινής κάμερας, ένας άλλος είναι υπεύθυνος για την ανίχνευση των γύρω οχημάτων και πεζών όπως και για την πρόβλεψη της πορείας τους. Τέλος άλλος ένας αλγόριθμος είναι υπεύθυνος για την ανίχνευση των γραμμών στο οδόστρωμα. Όλοι αυτοί οι διαφορετικοί αλγόριθμοι παράγουν δεδομένα τα οποία θα ήταν αδύνατο να παράγουμε χωρίς την χρήση αλγορίθμων μηχανικής μάθησης. Έτσι λοιπόν αυτοί οι αλγόριθμοι μας δίνουν την δυνατότητα να κάνουμε πράγματα τα οποία έως τώρα θεωρούσαμε αδύνατα.

3.ΜΙΚΡΟ-ΟΧΗΜΑ - ΣΥΣΤΗΜΑ

3.1 Περιγραφή μικρο-οχήματος

Για την υλοποίηση του μικρο-οχήματος ή αλλιώς συστήματος χρειάζονται 2 βασικά εξαρτήματα. Το πρώτο είναι ένα τηλεκατευθυνόμενο όχημα το οποίο μας προσφέρει μια έτοιμη πλατφόρμα, πάνω στην οποία μπορούμε να τοποθετήσουμε τα εξαρτήματα που θα είναι υπεύθυνα για τον έλεγχο του αυτοκινήτου με την χρήση ενός αλγορίθμου. Το δεύτερο βασικό εξάρτημα είναι ένας μικροελεγκτής ο οποίος μπορεί να ελέγξει το όχημα, να επεξεργαστεί δεδομένα από την κάμερα και να τρέχει τον αλγόριθμο μηχανικής μάθησης. Στον *Πίνακα 3.1* υπάρχει η λίστα με όλα τα εξαρτήματα του συστήματος.

Πίνακας 3.1: Πίνακας εξαρτημάτων

Αριθμός Εξ.	Εξαρτήματα
1	Τηλεκατευθυνόμενο όχημα κλίμακας 1:18
2	Μικροελεγκτής Raspberry Pi 4 4GB
3	Κάμερα Raspberry Pi camera module 5MP
4	Μπαταρία με έξοδο 5V/2.7A
5	Μπαταρία με έξοδο 7.4V
6	Πλακέτα ελέγχου μοτέρ L298N (h-bridge)
7	Μοτέρ servo SG90
8	Ασύρματο τηλεχειριστήριο τύπου Xbox

3.2 Ανάλυση εξαρτημάτων

Όλα τα παραπάνω εξαρτήματα είναι απαραίτητα για την σωστή και ομαλή λειτουργία του συστήματος, το καθένα όμως εξυπηρετεί έναν μοναδικό σκοπό. Πιο συγκεκριμένα:
Τηλεκατευθυνόμενο όχημα: Προσφέρει το σασί και όλα τα απαραίτητα μηχανικά μέρη για την κίνηση του αυτοκινήτου.

Εικόνα 3. 1: Τηλεκατευθυνόμενο όχημα

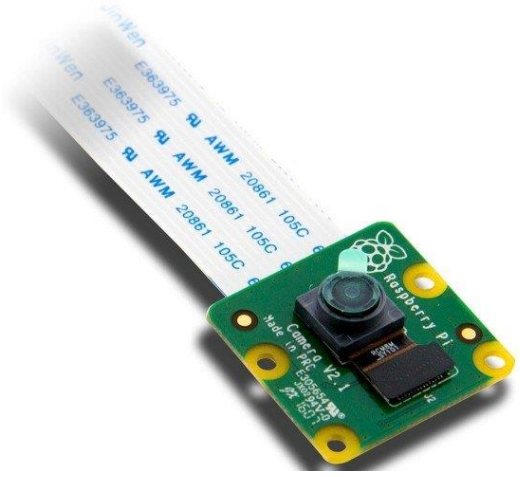
Μικροελεγκτής Raspberry Pi : Είναι ο "εγκέφαλος" του συστήματος καθώς είναι υπεύθυνο για τον έλεγχο της κίνησης του αυτοκίνητου και τον έλεγχο της κάμερας μέσω του προγράμματος. Ένα μεγάλο πλεονέκτημα και ο κύριος λόγος χρήσης του συγκεκριμένου μικροελεγκτή στο σύστημα είναι οι ακροδέκτες εισόδου - εξόδου γενικής χρήσης (GPIO - General purpose input output). Οι συγκεκριμένοι

ακροδέκτες μας δίνουν την δυνατότητα να ελέγχουμε διάφορα μηχανικά μέρη με την χρήση παλμών και συνεχούς ρεύματος.

Εικόνα 3. 2: Μικροελεγκτής Raspberry Pi

Εικόνα 3. 4: Ακροδέκτες-GPIO

Κάμερα : Απαραίτητη για την λήψη φωτογραφιών κατά την διάρκεια της εκπαίδευσης και της αυτόνομης πλοήγησης.



Εικόνα 3. 3: Κάμερα Raspberry Pi

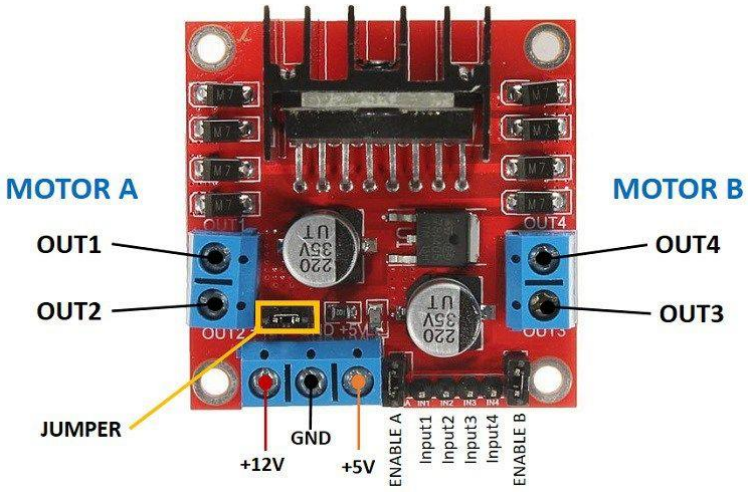
Μπαταρία 5V/2.7A : Τροφοδοτεί τον μικροελεγκτή με την τάση ρεύματος που συστήνει ο κατασκευαστής.

Εικόνα 3. 5: Μπαταρία – Power bank 5V/2.7A

Μπαταρία 7.4V : Τροφοδοτεί το μοτέρ του οχήματος, απαραίτητο για την κίνηση. Οποιαδήποτε μπαταριά μέχρι 12V θα μπορούσε να δουλέψει για αυτό το σκοπό.

Εικόνα 3. 6: Μπαταρία 7.4V

Πλακέτα L298N : Η πλακέτα αυτή μας δίνει την δυνατότητα να ελέγχουμε την ταχύτητα του μοτέρ.



Εικόνα 3. 7: Πλακέτα L298N

Μοτέρ servo : Το τηλεκατευθυνόμενο είχε ήδη ένα παρόμοιο μοτέρ το οποίο δεν ήταν συμβατό με τον μικροελεγκτή γιατί ήταν διαφορετικών προδιαγραφών. Αυτό το μοτέρ είναι παρόμοιου μεγέθους με το εργοστασιακό και ελέγχεται ευκολά από τον μικροελεγκτή. Το μοτέρ αυτό είναι υπεύθυνο για την κλίση του τιμονιού.

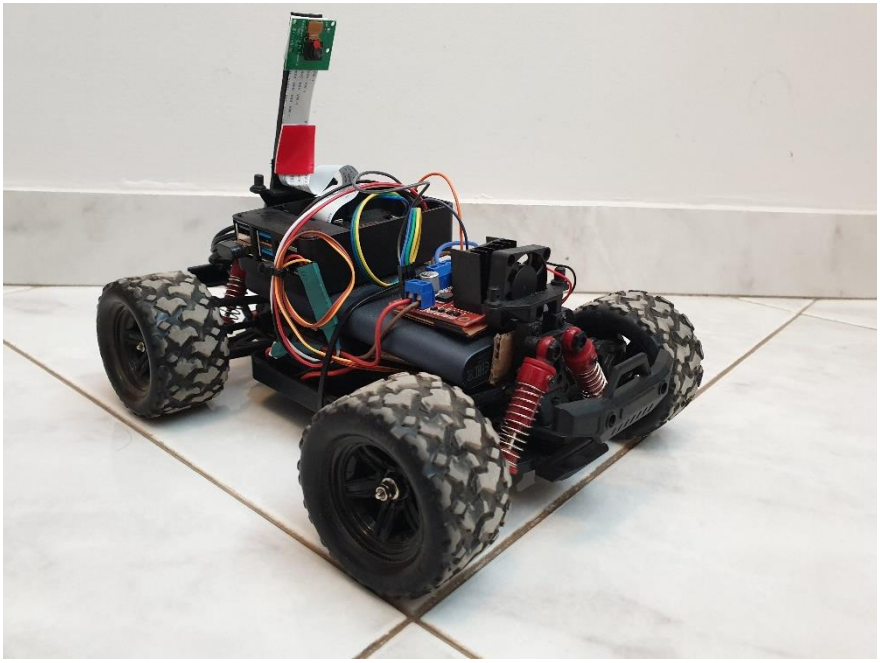
Εικόνα 3. 8: Μοτέρ-Servo τιμονιού

Ασύρματο τηλεχειριστήριο : Με το τηλεχειριστήριο μπορούμε να ελέγξουμε το όχημα ασύρματα και κάνει πιο εύκολο το στάδιο της εκπαίδευσης του αλγορίθμου μηχανικής μάθησης.

Εικόνα 3. 9: Ασύρματο τηλεχειριστήριο

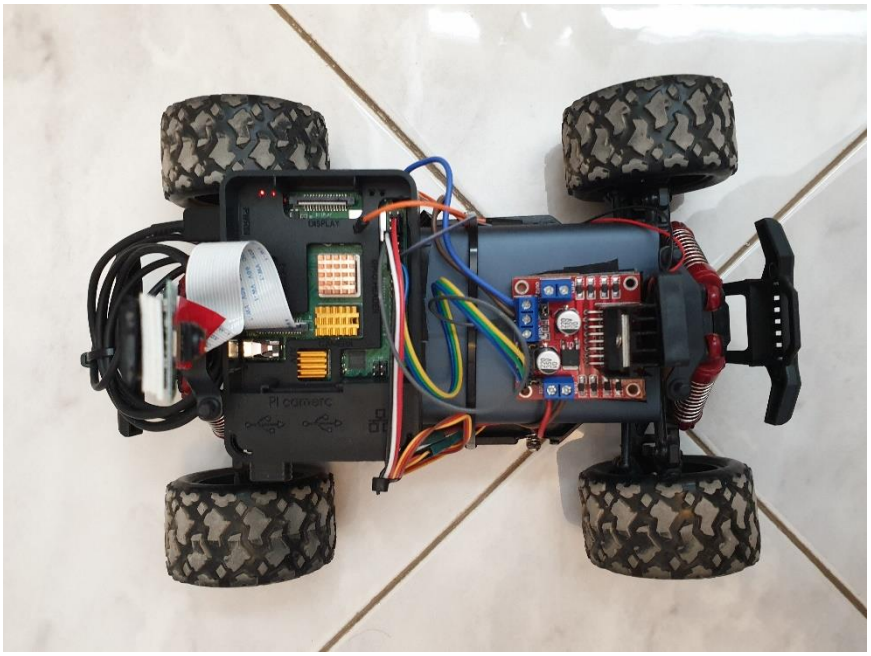
3.3 Συνδεσμολογία εξαρτημάτων

Για να λειτουργήσει το σύστημα πρέπει όλα αυτά τα εξαρτήματα να είναι συνδεδεμένα έτσι ώστε να αλληλοεπιδρούν. Η μπαταρία 5V/2.7A τροφοδοτεί με ρεύμα τον μικροελεγκτή μέσω ενός καλωδίου με ακροδέκτες τύπου USB - type C. Η κάμερα είναι συνδεδεμένη στην ειδική υποδοχή κάμερας του μικροελεγκτή. Το πλεονέκτημα αυτού του τύπου σύνδεσης είναι ότι η κάμερα συνδέεται απευθείας στον γραφικό επεξεργαστή του μικροελεγκτή. Έτσι μπορούμε να έχουμε καλύτερη απόδοση από την κάμερα και λιγότερη χρήση πόρων, σε σχέση με την εναλλακτική σύνδεση κάμερας μέσω θύρας USB. Το μοτέρ (servo) είναι συνδεδεμένο στους ακροδέκτες 4, 6 και 7 του μικροελεγκτή, όπου ο 4 του παρέχει θετικό (ρεύμα), ο 6 αρνητικό (ρεύμα) και ο 7 χρησιμοποιείται για την επικοινωνία και τον έλεγχο του. Οι ακροδέκτες 11,13 και 15 του μικροελεγκτή είναι συνδεδεμένοι στους ακροδέκτες IN3, IN4 και ENB της πλακέτας L298N αντίστοιχα και επιτρέπουν τον έλεγχο του μοτέρ. Το μοτέρ είναι συνδεδεμένο στις εξόδους OUT3 και OUT4 της πλακέτας από όπου παίρνει το ρεύμα για να λειτουργήσει. Η μπαταρία 7.4V τροφοδοτεί με ρεύμα την πλακέτα και είναι συνδεδεμένη στις εισόδους +12V και GND της πλακέτας. Στην είσοδο GND της πλακέτας είναι συνδεδεμένο και ένα καλώδιο που την συνδέει με τον ακροδέκτη 9 του μικροελεγκτή έτσι ώστε να έχουν ένα κοινό αρνητικό (ρεύμα). Τέλος, σε μια από τις 4 θύρες USB του μικροελεγκτή είναι συνδεδεμένος ο δέκτης για την ασύρματη επικοινωνία με το τηλεχειριστήριο τύπου Xbox. Στις εικόνες 3.10 έως 3.12 φαίνεται το μικρο-όχημα με όλα τα συνδεδεμένα εξαρτήματα.



Εικόνα 3. 10: Μικρο-όχημα, μπροστά όψη

Εικόνα 3. 11: Μικρο-όχημα, πίσω όψη



Εικόνα 3. 12: Μικρο-όχημα, πάνω όψη

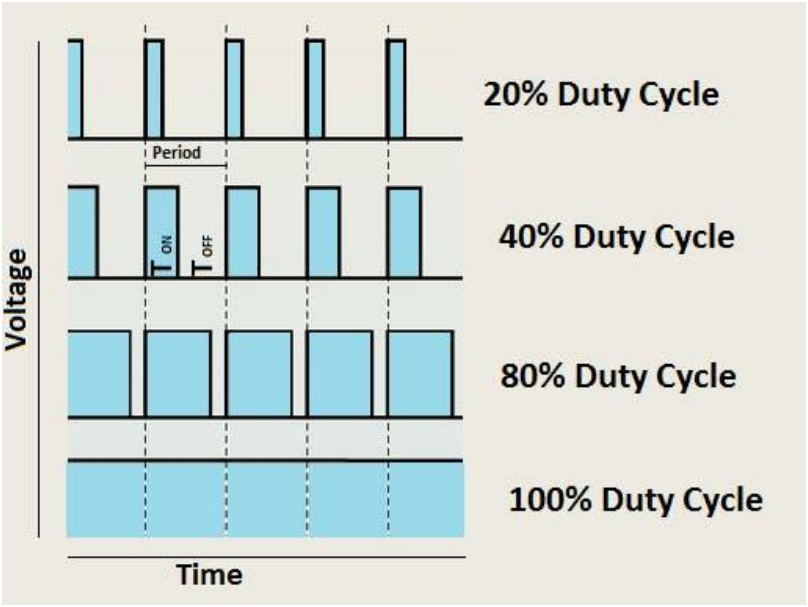
3.4 Περιβάλλον και έλεγχος μικροελεγκτή

Ο μικροελεγκτής τρέχει το λειτουργικό σύστημα Raspberry Pi OS Lite, το οποίο είναι μια έκδοση του λειτουργικού συστήματος linux και δεν έχει γραφικό περιβάλλον για τον χρήστη. Έτσι, όλος ο έλεγχος του συστήματος γίνεται μέσω της κονσόλας και απομακρυσμένα από έναν άλλο υπολογιστή συνδεδεμένο στο ίδιο τοπικό δίκτυο. Η σύνδεση στην απομακρυσμένη κονσόλα γίνεται μέσω SSH (Secure Shell protocol). Με αυτό τον τρόπο ο μικροελεγκτής μπορεί να είναι συνδεδεμένος στο ασύρματο δίκτυο και να έχουμε πλήρη έλεγχο του λειτουργικού από άλλον υπολογιστή χωρίς την χρήση κάποιου καλωδίου που να περιόριζε την κίνηση του οχήματος. Επίσης, για την πιο εύκολη μετακίνηση εικόνων και αρχείων μεταξύ μικροελεγκτή και υπολογιστή μπορεί να χρησιμοποιηθεί σύνδεση μέσω FTP (File Transfer Protocol). Τέλος όλος ο κώδικας που τρέχει στον μικροελεγκτή είναι γραμμένος στην γλώσσα προγραμματισμού Python.

3.5 Έλεγχος μικρο-οχήματος

Ένα από τα βασικά μέρη της εργασίας είναι ο έλεγχος του μικρο-οχήματος. Πρέπει να υπάρχει ένας εύκολος τρόπος για τον χρήστη να οδηγήσει το όχημα σε μια διαδρομή, τέτοιος ώστε να επιτρέπει στον μικροελεγκτή να καταγράφει εικόνες και παράλληλα να γνωρίζει ποια είναι η κατάσταση ελέγχου του οχήματος. Δηλαδή να ξέρει σε κάθε χρονική στιγμή την θέση του τιμονιού και του γκαζιού. Ένας τρόπος ελέγχου του οχήματος ο οποίος μπορεί να γίνει ασύρματα, με ελάχιστη καθυστέρηση και μας δίνει αρκετή ακρίβεια στον έλεγχο είναι μέσω ενός τηλεχειριστηρίου παιχνιδομηχανής. Το τηλεχειριστήριο μπορεί να επικοινωνήσει ασύρματα με τον μικροελεγκτή. Με την χρήση μιας βιβλιοθήκης της Python μπορούμε να επεξεργαστούμε το πάτημα κάθε κουμπιού ως "event". Έτσι έχουμε την δυνατότητα να αντιστοιχίσουμε το πάτημα ενός κουμπιού με μια εντολή στο πρόγραμμα.

Πιο συγκεκριμένα ο έλεγχος του τιμονιού και του γκαζιού γίνεται μέσω παλμών PWM (Pulse Width Modulation), όπου αλλάζοντας τον κύκλο ισχύος (duty cycle) μπορούμε να ελέγξουμε την κλίση του τιμονιού και την ταχύτητα του μοτέρ.



Εικόνα 3. 13: Duty Cycle - PWM

4.ΑΛΓΟΡΙΘΜΟΣ ΜΗΧΑΝΙΚΗΣ ΜΑΘΗΣΗΣ

4.1 Συλλογή δεδομένων

Για την υλοποίηση του αλγορίθμου μηχανικής μάθησης απαιτούνται δεδομένα πάνω στα οποία θα εκπαιδευτεί το μοντέλο. Τα δεδομένα αυτά αποτελούν το βασικότερο κομμάτι του αλγορίθμου καθώς θα καθορίσουν το τελικό αποτέλεσμα και αν δεν είναι "καθαρά", δηλαδή αν περιέχουν περιττές πληροφορίες ή πληροφορίες οι οποίες θα μπερδέψουν το μοντέλο, θα κάνουν την ανάπτυξη του μοντέλου αρκετά δύσκολη. Στα πλαίσια της εργασίας τα δεδομένα που συλλέγουμε είναι οι εικόνες που καταγράφει ο μικροελεγκτής με τις αντίστοιχες εντολές ελέγχου κατά την διάρκεια της καταγραφής εικόνων εκπαίδευσης.

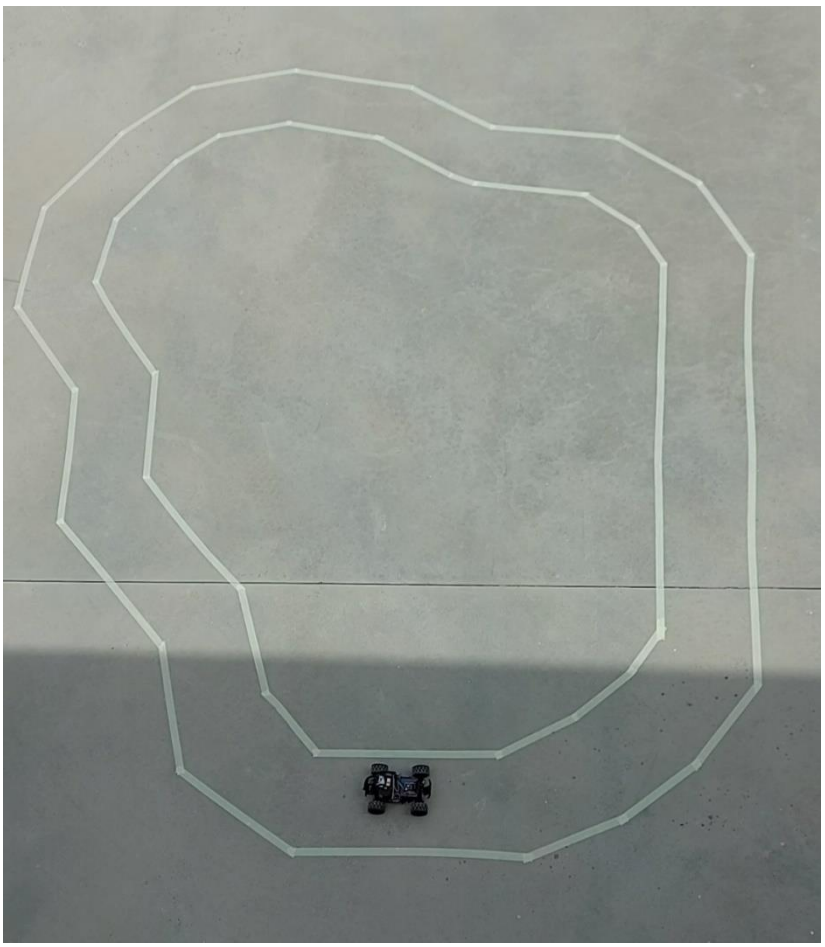
Η καταγραφή των εικόνων εκπαίδευσης γίνεται με την βοήθεια του χρήστη. Ο χρήστης πλοηγεί το όχημα σε μια τεχνητή πίστα και ο μικροελεγκτής καταγράφει μια σειρά από εικόνες. Με μερικές τεχνικές βελτιστοποίησης και με χαμηλή ανάλυση ο μικροελεγκτής μπορεί να καταγράφει με ευκολία 40 εικόνες ανά δευτερόλεπτο. Οι εντολές που αντιστοιχούν σε κάθε εικόνα αποθηκεύονται ως το όνομα του αρχείου της εικόνας και περιγράφουν το ποσοστό γκαζιού και την κλίση του τιμονιού. Αυτός ο τρόπος καταγραφής κάνει το επόμενο βήμα της ταξινόμησης σε κατηγορίες και καθαρισμού των εικόνων πιο εύκολο.

4.2 Καθαρισμός δεδομένων

Από τις εικόνες μπορούμε να δούμε ότι το καρέ περιέχει το πάνω μέρος από τις ρόδες του οχήματος και το ένα τέταρτο της εικόνας αποτελείται από τον ορίζοντα. Ο ορίζοντας δεν περιέχει κάποια χρήσιμη πληροφορία για τον αλγόριθμο και επιβραδύνει την απόδοση του. Οι ρόδες όμως εκτός από άχρηστη πληροφορία είναι και ένα πολύ μικρό κομμάτι της εικόνας το οποίο μπορεί να κάνει το μοντέλο πλασματικά πολύ αποδοτικό, αλλά στην πράξη ανίκανο να κάνει σωστές προβλέψεις. Αυτό μπορεί να γίνει γιατί, όταν οι ρόδες είναι στραμμένες προς τα δεξιά η εικόνα ανήκει στην κατηγορία δεξιά και αντίστοιχα όταν είναι αριστερά ή στο κέντρο, έτσι ο αλγόριθμος θα διδαχθεί ότι όταν οι ρόδες είναι δεξιά η εικόνα ανήκει στην κατηγορία δεξιά και ούτω καθεξής. Αυτό ισχύει πάντα για τις εικόνες που θα δει ο αλγόριθμος στην διαδικασία της εκπαίδευσης και της αξιολόγησης με αποτέλεσμα να έχει υψηλή αποδοτικότητα. Στην πραγματικότητα, όμως, όταν το μοντέλο κληθεί να προβλέψει την κίνηση του οχήματος με βάση μια εικόνα, οι ρόδες θα είναι πάντα στο κέντρο και ο αλγόριθμος θα προβλέπει πάντα ότι η εικόνα αντιστοιχεί στην κατηγορία κέντρο, κάτι το οποίο δεν είναι αληθές. Για να αποφύγουμε αυτό το πρόβλημα και να αυξήσουμε την αποδοτικότητα του αλγορίθμου μπορούμε απλά να περικόψουμε τις περιττές πληροφορίες από την εικόνα. Στην περίπτωση αυτή ο μικροελεγκτής έχει την υπολογιστική ικανότητα να κάνει περικοπή της εικόνας πριν την αποθηκεύσει χωρίς να επηρεαστεί η ταχύτητα λήψης των καρέ. Η περικοπή σε πραγματικό χρόνο είναι η καλύτερη λύση καθώς εξασφαλίζει ότι οι εικόνες στις οποίες θα εκπαιδευτεί το μοντέλο θα είναι ίδιες με αυτές τις οποίες θα κληθεί να κατηγοριοποιήσει στην πράξη.

Εικόνα 4. 1: Εικόνα από την κάμερα χωρίς περικοπή

Εικόνα 4. 2: Εικόνα από την κάμερα με περικοπή



Εικόνα 4. 3 Πανοραμική όψη πίστας

4.3 Ταξινόμηση δεδομένων

Για να ταξινομήσουμε τα δεδομένα σε κατηγορίες πρέπει πρώτα να αποφασίσουμε ποιες θα είναι οι κατηγορίες που θα ζητήσουμε από τον αλγόριθμο να προβλέψει. Μετά από αρκετές δοκιμές και λαμβάνοντας υπόψη τους περιορισμούς και τις δυνατότητες του συστήματος, οι κατηγορίες που αποφέρουν τα καλύτερα αποτελέσματα είναι: **Αριστερά**, **Κέντρο**, **Δεξιά** και **Εκτός πίστας.** Έτσι λοιπόν από την διαδικασία της συλλογής εικόνων έχουμε δύο βασικές κατηγορίες εικόνων, οι εικόνες που καταγράφηκαν εντός πίστας και εκτός πίστας. Για να τις ταξινομήσουμε σε φακέλους αρχικά πρέπει να τις μεταφέρουμε σε έναν υπολογιστή με την χρήση του πρωτοκόλλου FTP. Στην συνέχεια μέσω κώδικα είναι δυνατή η ταξινόμηση σε φακέλους που αναπαριστούν τις κλάσεις. Η ταξινόμηση γίνεται με βάση το όνομα της εικόνας, το οποίο περιγράφει την κατάσταση ελέγχου του οχήματος και πιο συγκεκριμένα την κατεύθυνση του τιμονιού. Η τέταρτη κλάση **Εκτός Πίστας** απλά περιέχει εικόνες οι οποίες καταγράφηκαν εκτός πίστας και δεν έχει σημασία η κατεύθυνση ή το γκάζι που αντιστοιχεί στην εικόνα.

4.4 Σχεδιασμός αλγορίθμου μηχανικής μάθησης

Ο καλύτερος τρόπος για κατηγοριοποίηση εικόνων με αλγόριθμο μηχανικής μάθησης είναι η χρήση συνελικτικών νευρωνικών δικτύων (Convolutional Neural Networks), όπως αναφέρθηκε και στην δεύτερη ενότητα. Για να σχεδιάσουμε τον αλγόριθμο πρέπει να δούμε τα δεδομένα που έχουμε, τι θα ζητήσουμε από τον

αλγόριθμο και να λάβουμε υπόψη τους περιορισμούς του συστήματος. Αρχικά τα δεδομένα είναι οι εικόνες που καταγράφονται από τον μικροελεγκτή. Μετά από δοκιμές το καλύτερο μέγεθος εικόνας φαίνεται να είναι 160 επί 128 pixel γιατί αυτή η ανάλυση χρησιμοποιεί όλη το οπτικό πεδίο του φακού χωρίς να χάνονται χρήσιμες πληροφορίες λόγω της χαμηλής ανάλυσης. Μετά την περικοπή και έναν μετασχηματισμό η εικόνα έχει μέγεθος 85 επί 85. Τα δεδομένα που θα δέχεται ο αλγόριθμος ,είναι εικόνες οι οποίες αναπτύσσονται σε πίνακα τριών διαστάσεων 85 επί 85 επί 3. Η τρίτη διάσταση αφορά τα 3 διαφορετικά χρώματα που αποτελούν την εικόνα. Έπειτα έχουμε το τι ζητάμε από τον αλγόριθμο, όπου στην προκειμένη περίπτωση του ζητάμε να κατηγοριοποιήσει την εικόνα σε μία από τις τέσσερις κατηγορίες. Τέλος, είναι οι περιορισμοί όπου έχουν να κάνουν με την υπολογιστική δυνατότητα του μικροελεγκτή, πρέπει δηλαδή ο αλγόριθμος να μην έχει πολύ μεγάλη υπολογιστική πολυπλοκότητα έτσι ώστε να μπορεί να επεξεργάζεται τουλάχιστον 30 εικόνες ανά δευτερόλεπτο ακόμα και όταν τρέχει στον μικροελεγκτή.

Το κομμάτι του σχεδιασμού και της αξιολόγησης του αλγορίθμου είναι ένα από τα πιο χρονοβόρα κομμάτια όλης της διαδικασίας υλοποίησης καθώς υπάρχουν πάρα πολλές υπερπαράμετροι (hyperparameters) που καθορίζουν το αποτέλεσμα. Με κάθε αλλαγή κάποιας παραμέτρου, ο αλγόριθμος πρέπει να εκπαιδευτεί από την αρχή και να αξιολογηθεί, μια διαδικασία που μπορεί να πάρει από ένα λεπτό έως και ώρες ανάλογα την ικανότητα κάθε υπολογιστή. Ο αλγόριθμος που προέκυψε μετά από αρκετές δοκιμές αποτελείται από 3 συνελικτικά επίπεδα τα οποία έχουν από 6, 8 και 14 φίλτρα το κάθε ένα. Το κάθε φίλτρο έχει μέγεθος 3 επί 3 και η συνάντηση ενεργοποίησης που χρησιμοποιείται σε κάθε επίπεδο είναι η Λειτουργία Ενεργοποίησης Ανορθωτή (ReLU). Μέσα από κάθε επίπεδο γίνεται ομαδοποίηση γειτόνων με μέγεθος 2 επί 2 όπου επικρατεί η μέγιστη τιμή γείτονα. Μετά οι πίνακες που προκύπτουν από αυτά τα επίπεδα μετατρέπονται σε ένα διάνυσμα το οποίο τροφοδοτείται στο επόμενο επίπεδο. Τα τελευταία 3 επίπεδα του αλγορίθμου αποτελούνται από πυκνά συνδεδεμένους κόμβους, όπου το κάθε επίπεδο έχει 40, 40 και 4 κόμβους αντίστοιχα. Σε αυτά τα επίπεδα χρησιμοποιείται η ίδια συνάρτηση ενεργοποίησης με τα συνελικτικά επίπεδα εκτός από το τελευταίο, το οποίο χρησιμοποιεί την Κανονικοποιημένη Εκθετική Συνάρτηση (Softmax), έτσι ώστε το αποτέλεσμα των τεσσάρων κόμβων να αθροίζεται πάντα σε ένα. Η τιμή του κάθε κόμβου είναι η πιθανότητα η εικόνα να ανήκει στην κατηγορία που αναπαριστά ο κόμβος, τα αποτελέσματα των τεσσάρων κόμβων αθροίζουν πάντα στην μονάδα.

Ο αλγόριθμος βελτιστοποίησης που χρησιμοποιείται στο δίκτυο είναι ο Adam και ο αλγόριθμος απώλειας είναι ο Sparse Categorical Cross Entropy.
Η υλοποίηση του αλγορίθμου έγινε με την χρήση των βιβλιοθηκών Tensorflow και Keras για Python. Αυτές οι βιβλιοθήκες είναι από τις πιο δημοφιλείς για κατασκευή και υλοποίηση αλγορίθμων μηχανικής μάθησης.

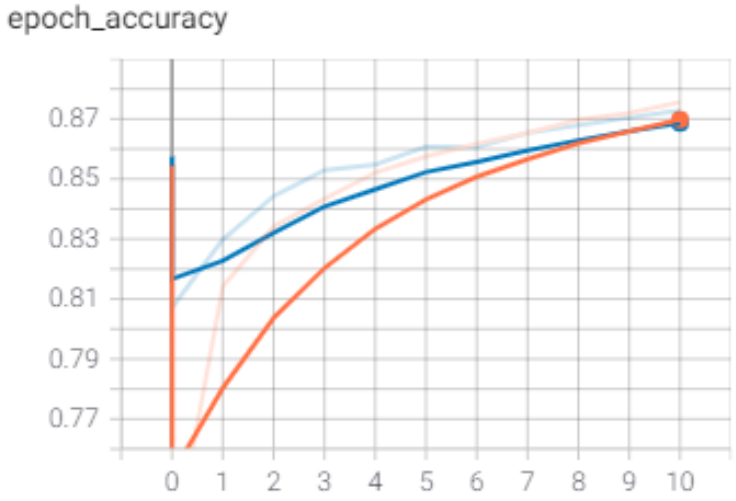
```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
rescaling (Rescaling)        (None, 85, 85, 3)         0
_________________________________________________________________
random_contrast (RandomContr (None, 85, 85, 3)         0
_________________________________________________________________
conv2d (Conv2D)              (None, 83, 83, 6)         168
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 41, 41, 6)         0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 39, 39, 8)         440
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 19, 19, 8)         0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 17, 17, 14)        1022
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 8, 8, 14)          0
_________________________________________________________________
flatten (Flatten)            (None, 896)               0
_________________________________________________________________
dropout (Dropout)            (None, 896)               0
_________________________________________________________________
dense (Dense)                (None, 40)                35880
_________________________________________________________________
dense_1 (Dense)              (None, 40)                1640
_________________________________________________________________
dense_2 (Dense)              (None, 4)                 164
=================================================================
Total params: 39,314
Trainable params: 39,314
Non-trainable params: 0
_________________________________________________________________
```

Εικόνα 4. 4 Περιγραφή αρχιτεκτονικής του μοντέλου

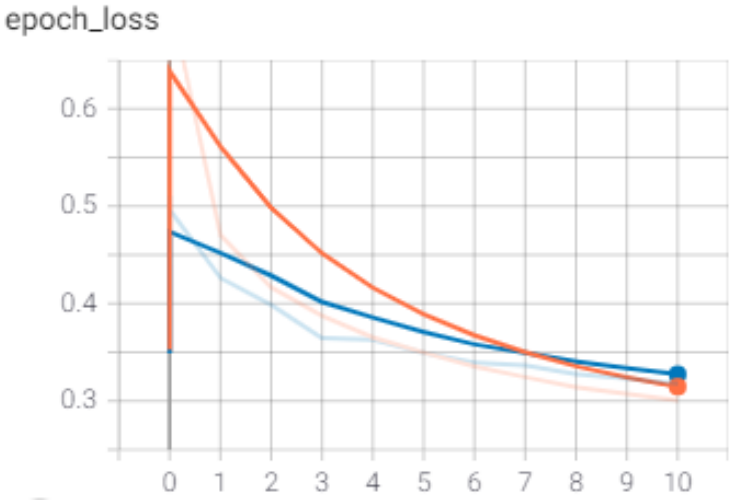
4.5 Εκπαίδευση και αξιολόγηση αλγορίθμου μηχανικής μάθησης

Οι βιβλιοθήκες που αναφέρθηκαν στην προηγούμενη ενότητα διευκολύνουν και το κομμάτι της εκπαίδευσης του αλγορίθμου. Υπάρχουν συναρτήσεις όπου μας επιτρέπουν να δημιουργήσουμε σύνολα δεδομένων (datasets) από φακέλους με αρχεία. Στην προκειμένη περίπτωση οι φάκελοι είναι τέσσερις και αντιπροσωπεύουν τις τέσσερις κατηγορίες που χωρίζονται οι εικόνες. Κάθε φάκελος περιέχει εικόνες που αντιστοιχούν στην κατηγορία του φακέλου. Μέσω της συνάρτησης της βιβλιοθήκης παράγονται σύνολα δεδομένων όπου η κάθε εικόνα έχει μια ετικέτα (την κατηγορία της) και χωρίζονται σε ομάδες 10 εικόνων. Ο αλγόριθμος απώλειας που ταιριάζει στον συγκεκριμένο νευρωνικό δίκτυο είναι ο Sparse Categorical Cross Entropy. Αυτός ο αλγόριθμος επιτρέπει την μέτρηση της απώλειας σε μοντέλα με κατηγοριοποίηση πολλών κλάσεων.

Τα παρακάτω γραφήματα παρουσιάζουν την εξέλιξη των τιμών κατά την διάρκεια της εκπαίδευσης. Ο κάθετος άξονας αναπαριστά την τιμή της μεταβλητής και ο οριζόντιος το βήμα στο οποίο μετρήθηκε η τιμή. Με μπλε χρώμα αναπαρίστανται οι μετρήσεις πάνω στο σύνολο δεδομένων επαλήθευσης και με πορτοκάλι για το σύνολο εκπαίδευσης. Όπως φαίνεται από τα γραφήματα αυτά ο αλγόριθμος παρουσιάζει σταδιακή και συνεχή βελτίωση της ακρίβειας σε όλα τα βήματα. Μετά το δέκατο βήμα φαίνεται ότι η ακρίβεια εκπαίδευσης ξεπερνά την ακρίβεια επαλήθευσης, κάτι το οποίο είναι σημάδι υπερβολικής προσαρμογής.



Εικόνα 4. 5 Γράφημα ακρίβειας μοντέλου κατά την διάρκεια της εκπαίδευσης



Εικόνα 4.6 Γράφημα απώλειας μοντέλου κατά την διάρκεια της εκπαίδευσης

4.6 Βελτιστοποίηση αλγορίθμου – μοντέλου

Είναι βασικό να λάβουμε υπόψη τις περιορισμένες δυνατότητες του μικροελεγκτή. Παρόλο που έχει αρκετά καλά χαρακτηριστικά για να τρέξει ένα λειτουργικό σύστημα που έχει τις περισσότερες λειτουργίες οποιουδήποτε κοινού υπολογιστή, εξακολουθεί να μην έχει την δύναμη ενός ισχυρού υπολογιστή με κάρτα γραφικών. Αν δοκιμάσουμε να τρέξουμε το μοντέλο όπως είναι για την πρόβλεψη των κατηγοριών εικόνων και την πλοήγηση του οχήματος θα δούμε ότι δεν μπορεί να αποδώσει καλά. Ενώ οι κλάσεις που προβλέπει θα είναι σωστές, οι κινήσεις του οχήματος θα είναι λανθασμένες. Αυτό συμβαίνει γιατί ο μικροελεγκτής με το παρόν μοντέλο μπορεί να προβλέψει 3 με 4 φωτογραφίες ανά δευτερόλεπτο, έτσι όταν ο αλγόριθμος θα προβλέψει την κλάση δεξιά μετά από την κλάση κέντρο το όχημα θα έχει βγει από την πίστα μέχρι να γίνει η πρόβλεψη δεξιά. Είναι λοιπόν φανερό ότι πρέπει να γίνει κάποια βελτιστοποίηση.

Υπάρχουν δύο τρόποι που ίσως δώσουν την δυνατότητα στον μικροελεγκτή να προβλέπει και να επεξεργάζεται περισσότερες εικόνες ανά δευτερόλεπτο. Ο ένας είναι να απλοποιήσουμε το μοντέλο. Αν και υπάρχουν περιθώρια απλοποίησης, στην πράξη θα δούμε ότι ο μικροελεγκτής θα μπορεί να επεξεργάζεται 10 εικόνες ανά δευτερόλεπτο με ένα απλοποιημένο μοντέλο αλλά η ακρίβεια του μοντέλου θα είναι αρκετά χαμηλότερη. Έτσι καταλήγουμε στο ίδιο αποτέλεσμα που είχαμε χωρίς την βελτιστοποίηση. Ο δεύτερος τρόπος ο οποίος αποδείχθηκε και ο πιο αποτελεσματικός είναι η μετατροπή του μοντέλου με την χρήση της βιβλιοθήκης Tensorflow Light. Η βιβλιοθήκη αυτή μας δίνει την δυνατότητα να συμπιέσουμε και να απλοποιήσουμε το μοντέλο χάνοντας ελάχιστη ακρίβεια στις προβλέψεις. Η κύρια μετατροπή που κάνει αυτή η βιβλιοθήκη είναι ότι αλλάζει τους τύπους μεταβλητών του μοντέλου από Float32 σε Float8, έτσι μειώνεται το μέγεθος κάθε μεταβλητής στο ένα τέταρτο κάνοντας την δουλειά του μικροελεγκτή πιο εύκολη. Η θεωρία φαίνεται να αποδίδει και στην πράξη καθώς με το συμπιεσμένο μοντέλο ο μικροελεγκτής είναι ικανός να επεξεργαστεί πάνω από 60 εικόνες το δευτερόλεπτο χωρίς να απλοποιήσουμε το μοντέλο και να χάσουμε ακρίβεια.

4.7 Αυτόνομη πλοήγηση οχήματος

Για την αυτόνομη πλοήγηση του οχήματος είναι βασικό να αντιστοιχηθούν οι προβλέψεις του αλγορίθμου με την κίνηση του οχήματος. Οι προβλέψεις είναι οι τέσσερις κατηγορίες, δεξιά, αριστερά, κέντρο και εκτός πίστας. Αφού το όχημα θα κάνει γύρους στην πίστα επ’ αόριστον και δεν χρειάζεται να σταματήσει σε κάποιο σημείο αυτόνομα, μπορούμε να θέσουμε το μοτέρ σε χαμηλή ταχύτητα όταν η πρόβλεψη είναι δεξιά ή αριστερά, σε κανονική ταχύτητα όταν είναι κέντρο και να το σταματήσουμε όταν η πρόβλεψη είναι εκτός πίστας. Αν χρησιμοποιήσουμε υψηλές ταχύτητες είναι επικίνδυνο να βγει το όχημα εκτός πίστας λόγω καθυστερημένης ή λανθασμένης πρόβλεψης. Για το τιμόνι, λογικό είναι ότι όταν η πρόβλεψη είναι κέντρο ή εκτός πίστας θα θέσουμε το τιμόνι στην κεντρική θέση, όταν η πρόβλεψη είναι αριστερά θα το θέσουμε με αριστερή κλίση και αντίστοιχα όταν είναι δεξιά.

Επιπλέον κατά την διάρκεια της αυτόνομης οδήγησης είναι απαραίτητο να μπορούμε να επικοινωνούμε και να ελέγχουμε το όχημα μέσω του τηλεχειριστηρίου. Ο πιο αποδοτικός τρόπος για να το πετύχουμε αυτό είναι να τρέξουμε την διαδικασία της αυτόνομης οδήγησης σε ένα ξεχωριστό thread του επεξεργαστή. Έτσι η διαδικασία της αυτόνομης οδήγησης και του χειρισμού μέσω του τηλεχειριστηρίου θα τρέχουν παράλληλα και θα μπορούμε να έχουμε τον έλεγχο του οχήματος. Ο κύριος λόγος της ανάγκης του ελέγχου είναι το κουμπί πανικού

όπου σταματάει το όχημα αν κάτι πάει στραβά, καθώς οποιοσδήποτε άλλος έλεγχος του οχήματος μέσω του τηλεχειριστηρίου καταπιέζεται από την διαδικασία της αυτόνομης οδήγησης.

5.ΕΠΙΛΟΓΟΣ ΚΑΙ ΣΥΜΠΕΡΑΣΜΑΤΑ

5.1 Αποτελέσματα και παρατηρήσεις

Μέσω της υλοποίησης και μελέτης της εργασίας αυτής όχι μόνο επιτεύχθηκε ο στόχος, δηλαδή η κατασκευή ενός αυτόνομου μικρό-οχήματος, αλλά έγιναν και αρκετές παρατηρήσεις. Μια από τις πιο ενδιαφέρουσες παρατηρήσεις ήταν το ότι, αν κατά την διάρκεια της εκπαίδευσης γινόταν λάθος πλοήγηση, δηλαδή ο χρήστης έπαιρνε κάποια στροφή ανοιχτά ή κλειστά, και αυτά τα δεδομένα συμπεριλαμβάνονταν στην εκπαίδευση του, τότε κατά την διάρκεια της αυτόνομης πλοήγησης ο αλγόριθμος θα έκανε το ίδιο λάθος στην ίδια στροφή. Επομένως, φαίνεται ότι κάθε εικόνα μετράει, καθώς διακόσιες εικόνες που περιείχαν αυτό το λάθος από τις σαράντα χιλιάδες που χρησιμοποιήθηκαν για την εκπαίδευση του αλγορίθμου κατάφεραν να επηρεάσουν το τελικό αποτέλεσμα. Επίσης, ένα πολύ σημαντικό συμπέρασμα είναι το ποσό σημαντική είναι η απόδοση ενός αλγορίθμου και το υπολογιστικό κόστος του. Προσθέτοντας ένα επίπεδο με δέκα νευρώνες στο δίκτυο ή γενικότερα κάνοντας κάποια αύξηση στις τιμές των υπέρ παραμέτρων δεν φαίνεται σημαντικό από θέμα κόστους, όταν όμως αυτό πρέπει να επαναληφθεί 30 φορές ανά δευτερόλεπτο φαίνεται πόσο σημαντικό είναι το κόστος του. Ο αλγόριθμος, χωρίς κάποια αλλαγή, είναι ικανός να επεξεργάζεται 30 εικόνες ανά δευτερόλεπτο ενώ με κάποια από τις αλλαγές αυτές η απόδοση πέφτει στις 25 εικόνες ανά δευτερόλεπτο.

Ένα συμπέρασμα που προκύπτει από την εργασία είναι ότι η αυτόνομη οδήγηση με την σημερινή τεχνολογία και τεχνογνωσία είναι κάτι εφικτό, όχι όμως κάτι εύκολο. Υπάρχουν εκατομμύρια σενάρια στα οποία θα πρέπει να αντιδράσει ένας αλγόριθμος, όπου πάντοτε το κύριο μέλημα του θα πρέπει να είναι η ασφάλεια των ανθρώπων. Δεν μπορούμε, λοιπόν, να υλοποιήσουμε ένα αυτόνομο σύστημα το οποίο θα είναι διαθέσιμο για το κοινό αν δεν είμαστε εκατό τοις εκατό σίγουροι ότι θα πάρει την σωστή απόφαση σε κάθε σενάριο.

5.2 Δυσκολίες, τρόποι αντιμετώπισης και βελτίωσης

Κατά την διαδικασία της υλοποίησης υπήρξαν αρκετές δυσκολίες. Μια από τις κυριότερες δυσκολίες αφορά στον έλεγχο του οχήματος και γενικότερα τον έλεγχο του μοτέρ κατά την διάρκεια της αυτόνομης πλοήγησης. Με τα εξαρτήματα που χρησιμοποιήθηκαν το μοτέρ δεν ανταποκρινόταν σε κύκλο ισχύος μικρότερο από 60% και στο διάστημα 60% με 70% η ταχύτητα του μοτέρ ήταν ανάλογη με την δύναμη της μπαταρίας. Έτσι, το μοτέρ έχει μια σχετικά απρόβλεπτη ανταπόκριση στην ελάχιστη δυνατή ταχύτητα, χαρακτηριστικό που δυσκολεύει τον έλεγχο. Αυτό οφείλεται κυρίως στην πλακέτα ελέγχου μοτέρ. Μια διαφορετική πλακέτα τύπου ESC (Electronic Speed Control) θα ήταν καταλληλότερη για αυτή την χρήση. Μια άλλη πιθανή λύση είναι η χρήση ενός επιταχυνσιόμετρου (accelerometer) το οποίο θα μας έδινε την δυνατότητα να βλέπουμε την ταχύτητα που είχε το όχημα κατά την διάρκεια της εκπαίδευσης και να προσπαθήσουμε να την ταιριάξουμε κατά την διαδικασία της αυτόνομης πλοήγησης. Με αυτό τον τρόπο δεν θα εξαρτάται το

σύστημα από την απόδοση του μοτέρ αλλά θα επιταχύνει ή θα επιβραδύνει μέχρι να ταιριάξει την επιθυμητή ταχύτητα.

Η δεύτερη βασική δυσκολία παρουσιάστηκε κατά την διάρκεια της εκπαίδευσης και της αυτόνομης πλοήγησης. Επειδή η πίστα βρισκόταν σε εξωτερικό χώρο η θέση του ήλιου και οι σκιές από τα δέντρα άλλαζαν την εμφάνιση της πίστας κατά την διάρκεια της ημέρας. Παρατηρήθηκε λοιπόν ότι το όχημα ήταν ικανό να πλοηγηθεί αυτόνομα για αρκετούς γύρους στην πίστα τις πρωινές ώρες ενώ τις απογευματινές ολοκλήρωνε έναν γύρο με δυσκολία. Αυτό βέβαια είναι εύκολο να διορθωθεί καθώς αν εκπαιδευτεί ο αλγόριθμος σε δεδομένα από διάφορες ώρες της ημέρας θα είναι ικανός να πλοηγηθεί στην πίστα ανεξάρτητα των συνθηκών όπως και έγινε. Όταν ο αλγόριθμος είχε δεδομένα από ποικίλες ώρες δεν εμφάνιζε αυτή την δυσκολία στην πλοήγηση. Μια άλλη λύση σε αυτό το πρόβλημα θα ήταν η επαύξηση των δεδομένων, μια διαδικασία όπου παίρνουμε την κάθε εικόνα και δημιουργούμε καινούργιες αλλάζοντας τυχαία τον θόρυβο, τον κορεσμό, την φωτεινότητα και άλλες ρυθμίσεις που αφορούν τις τιμές της εικόνας.

5.3 Επίλογος

Είναι εντυπωσιακό το πώς τόσα διαφορετικά εξαρτήματα, αλγόριθμοι και προγράμματα μπορούν να συνδυαστούν και να αποτελέσουν ένα αυτόνομο μικρο-όχημα. Αλγόριθμοι που παρουσιάστηκαν το 1960 μας δίνουν σήμερα την δυνατότητα να λύσουμε προβλήματα με έναν τελείως διαφορετικό τρόπο και να παρουσιάσουμε καινοτόμες λύσεις σε σημαντικά προβλήματα. Η μηχανική μάθηση, λοιπόν, είναι ένα ισχυρό εργαλείο το οποίο μπορεί να επιλύσει πολλά από αυτά με ευκολία.

ΠΑΡΑΡΤΗΜΑ 1 - ΠΗΓΑΙΟΣ ΚΩΔΙΚΑΣ

1.ΠΗΓΑΙΟΣ ΚΩΔΙΚΑΣ ΜΙΚΡΟΕΛΕΓΚΤΗ

1.1 main.py

Το αρχείο αυτό είναι η είσοδος στο πρόγραμμα που τρέχει στον μικροελεγκτή. Εδώ γίνεται η αρχικοποίηση από όλες τις επιμέρους κλάσεις και η διαχείριση των εισόδων του τηλεχειριστηρίου.

```
from inputs import get_gamepad
from Car import Car
from Camera import Cam
from Driver import Driver
from Dashboard import Dashboard
import concurrent.futures
import time
import sys

def main():

    PiDashboard = Dashboard()
    PiDashboard.addInfo("Started!")
    PiCar = Car(7, 15, 13, 11, PiDashboard)
    turned_on = True
    recording = False
    auto_pilot = False
    speed_limit = False
    try:
        #Get the controller inputs
        while turned_on:
            events = get_gamepad()
            for event in events:
                if event.code == "BTN_WEST" and event.state==0:
                    #Button pressed to calibrate the steering center angle
                    PiCar.calibrateSteer()
                elif event.code == "ABS_RZ":
                    #Button pressed to move the Car forwards
                    PiCar.setMotorSpeed(event.state,1)
                elif event.code == "ABS_Z":
                    #Button pressed to move the Car in reverse
                    PiCar.setMotorSpeed(event.state,0)
                elif event.code == "BTN_TR" and event.state==1:
                    #Button pressed to set speed limit
                    if speed_limit:
                        PiCar.setSpeedLimit(1)
                    else:
                        PiCar.setSpeedLimit(0.70)
```

```
                    speed_limit = not speed_limit
                elif event.code == "ABS_X":
                    #Button pressed to change the steering angle
                    PiCar.setSteer(event.state)
                elif event.code == "BTN_EAST" and event.state==1:
                    #Button pressed to start/stop recording
                    PiCar.setSteer(PiCar.center_steering+1,True)
                    time.sleep(1)
                    if not recording:
                        recording = True
                        PiDashboard.mode = "Camera"
                        PiCam = Cam(PiDashboard) #Open and start camera
                        #Open a thread to run the camera and keep controll of
 the car
                        executor = concurrent.futures.ThreadPoolExecutor(max_
workers=1)
                        e = executor.submit(PiCam.startRecording,PiCar)
                        executor.shutdown(wait=False)
                    else:
                        PiDashboard.addInfo("Stopping")
                        recording = False
                        PiCam.stop() #Stop the camera
                        time.sleep(1)
                        e.cancel()
                        PiDashboard.mode = "User"
                    PiCar.setSteer(PiCar.center_steering,True)
                elif event.code == "BTN_SOUTH" and event.state==1:
                    #Button pressed to delete last seconds of Recording
                    if recording:
                        PiDashboard.addInfo("Deleting frames...")
                        PiCam.deleteLastSec()

                elif event.code == "BTN_NORTH" and event.state==1:
                    #Button pressed to start/stop auto pilot
                    PiCar.setSteer(PiCar.center_steering-1,True)
                    time.sleep(1)
                    if not auto_pilot:
                        PiDriver = Driver(PiCar,PiDashboard)
                        auto_pilot = True
                        executor = concurrent.futures.ThreadPoolExecutor(max_
workers=1)
                        e = executor.submit(PiDriver.autoPilot3)
                        executor.shutdown(wait=False)
                    else:
                        PiDashboard.addInfo("Stopping auto pilot")
                        auto_pilot = False
                        PiDriver.auto_pilot = False
                        time.sleep(2)
                        e.cancel()
                    PiCar.setSteer(PiCar.center_steering,True)
```

```
                elif event.code == "BTN_SELECT" and event.state==1:
                    #Emergency motor stop-lock
                    PiCar.motor_lock = not PiCar.motor_lock
                    PiDashboard.addInfo("Car motor lock: {}".format(PiCar.mot
or_lock))
                elif event.code == "BTN_START" and event.state==0:
                    #Button pressed to turn off the car
                    PiDriver.PiCar.turnOffCar()
                    turned_on = False

    except:
        print("Unexpected error:", sys.exc_info())

    finally:
        #Check if the car is turned on
        if turned_on:
            PiDashboard.addInfo("The car wasn't turned off, turning off now")
            PiCar.turnOffCar()
        if recording:
            PiDashboard.addInfo("The camera wasn't stopped, stopping now")
            PiCam.stop()
            time.sleep(1)
            e.cancel()
        if auto_pilot:
            PiDashboard.addInfo("The auto pilot wasn't stopped,stopping now")
            PiDriver.auto_pilot = False
            time.sleep(1)
            e.cancel()
        PiDashboard.addInfo("Exiting")
        exit()


if __name__ == "__main__":
    main()
```

1.2 Camera.py

Σε αυτό το αρχείο υπάρχει η κλάση «Κάμερα» (Cam). Αυτή η κλάση διαχειρίζεται την κάμερα και περιέχει διαδικασίες που αφορούν την κάμερα όπως η καταγραφή εικόνων.

```
import os
import time
import glob
import numpy as np
from datetime import datetime
from PIL import Image
from picamera import PiCamera
from picamera.array import PiRGBArray
```

```
from Car import Car
from Dashboard import Dashboard

class Cam:
    def __init__(self,dashboard):
        #Initialize all the camera parameters and start the camera
        self.camera = PiCamera()
        self.camera.resolution = (160,128) #(width,height)
        self.camera.framerate = 40 # Frames per second
        self.rawCapture = PiRGBArray(self.camera,size=self.camera.resolution)
        self.stream = self.camera.capture_continuous(self.rawCapture, format=
"rgb", use_video_port=True)
       #Video port allows for better framerate but worst quality

        #Variables to controll the camera from another thread
        self.recording = True
        self.frame = None
        self.deleting = False

        self.Dashboard = dashboard
        self.Dashboard.addInfo("The camera is starting, say cheese!")
        time.sleep(2)

    def __saveImg__(self,frame,i,status):
        #Save the frame to an image file
        frame = frame[25:110] #cut the horizon and wheels
        img = Image.fromarray(frame)
        location = '{}/img{}_{}_{}_{}_.jpg'.format(self.savedir,i,status[0
],status[1],status[2],status[3])
        img.save(location)

    def startRecording(self,car):
        #Start to capture frames
        self.frame_num = 0
        self.car = car
        dt = datetime.now().strftime("%d-%m-%Y-%H-%M-%S")
        self.savedir = 'imgs/{}'.format(dt)
        os.mkdir(self.savedir)
        start = time.time()
        for frame in self.stream:
            self.frame = frame.array
            self.__saveImg__(self.frame,self.frame_num,self.car.getStatus())
            self.rawCapture.truncate(0)
            self.frame_num+=1
            if not self.recording:
                break

        finish = time.time()
        self.Dashboard.addInfo("Captured {} frames at {:.2f}fps".format(self.
frame_num, self.frame_num / (finish - start)))
```

```
    def deleteLastSec(self,sec = 5):
        #Delete last x seconds from the recording
        self.deleting = True
        frames_to_del = sec*self.camera.framerate
        #seconds*frames/second = frames
        frames_to_del = int(frames_to_del)
        fn = self.frame_num
        for i in range(frames_to_del):
            num = fn - i
            like = "{}/img{}*".format(self.savedir,num)
            for filename in glob.glob(like):
                os.remove(filename)
        self.frame_num = fn-frames_to_del if fn-frames_to_del>0 else 0
        self.Dashboard.addInfo('Deleted the last {}sec / {} frames'.format(se
c,frames_to_del))
        self.deleting = False

    def getFrame(self,cut_frame=True):
        #Get a single frame from the camera
        frame = next(self.stream)
        frame = frame.array
        self.rawCapture.truncate(0)
        if cut_frame:
            frame = frame[25:110]
            #print(frame.shape)
        return frame

    def stop(self):
        #Stop the camera and release resources
        self.Dashboard.addInfo("Stopping the camera")
        if self.recording:
            self.recording = False
            time.sleep(1)
        self.stream.close()
        self.rawCapture.close()
        self.camera.close()
```

1.3 Car.py

Σε αυτό το αρχείο υπάρχει η κλάση «Αυτοκίνητο» (Car). Αυτή η κλάση διαχειρίζεται όλες τις λειτουργίες για την κίνηση του μικρο-οχήματος όπως τον έλεγχο της ταχύτητας του μοτέρ και την κλίση του τιμονιού.

```
from inputs import get_gamepad
from Dashboard import Dashboard
import RPi.GPIO as GPIO
import json
import time
```

```
class Car:
    def __init__(self, servo_pin, enb_pin, in3_pin, in4_pin, dashboard):
        self.servo_pin = servo_pin
        self.enb_pin = enb_pin
        self.in3_pin = in3_pin
        self.in4_pin = in4_pin
        self.speed_limit = 1
        self.speed = 0
        self.angle = 0
        self.diraction = 'N'
        self.steer_side = 'M'
        self.motor_lock = False
        self.Dashboard = dashboard
        #Setting up the GPIO pins
        self.gpioSetup()
        #Reading the calibrated angle from settings.json
        self.readSteer()
        #Centering the steering
        self.setSteer(self.center_steering,True)
        self.Dashboard.addInfo("The Car is ready to use")

    def __dashboard__(self):
        self.Dashboard.diraction = self.diraction
        self.Dashboard.speed = self.speed
        self.Dashboard.angle = self.steer_duty
        self.Dashboard.display()

    def gpioSetup(self):
        GPIO.setmode(GPIO.BOARD)
        #Servo
        GPIO.setup(self.servo_pin,GPIO.OUT)
        self.servo = GPIO.PWM(self.servo_pin,50)
        #Motor
        GPIO.setup(self.enb_pin,GPIO.OUT)
        GPIO.setup(self.in3_pin,GPIO.OUT)
        GPIO.setup(self.in4_pin,GPIO.OUT)
        self.motor = GPIO.PWM(self.enb_pin,1000)
        #Starting Motor and Servo
        self.servo.start(0)
        self.motor.start(0)

    def readSteer(self):
        with open('settings.json') as f:
            data = json.load(f)
        self.center_steering = data['angle']
        f.close()
        self.Dashboard.addInfo("Car settings have been loaded")

    def setSteer(self,value,duty=False):
        if not duty:
```

```
            #Normalizing the controller value to [0,1]
            value = value/32770
            angle = round((value*-1.2)+self.center_steering,2)
        else:
            angle = value
        self.steer_duty = angle
        self.servo.ChangeDutyCycle(self.steer_duty)

        # %of steering angle
        self.angle = round((abs(angle-self.center_steering)/1.2)*100)
        if angle > self.center_steering:
            self.steer_side = 'L'
        elif angle < self.center_steering:
            self.steer_side = 'R'
        else:
            self.steer_side = 'M'

        self.__dashboard__()

    def calibrateSteer(self):
        #Find and save the center position of the wheels
        self.setSteer(8,True)
        time.sleep(0.5)
        self.setSteer(self.center_steering,True)
        angle = self.center_steering
        self.Dashboard.mode = 'Calibration'
        self.Dashboard.addInfo("Calibration started")
        while 1:
            events = get_gamepad()
            for event in events:
                if event.code == "ABS_HAT0X" and event.state!=0:
                    n = event.state
                    if n == -1:
                        if angle+0.05 <= 9:
                            angle = angle+0.05
                        else:
                            self.Dashboard.addInfo("Error! Too far")
                    elif n == 1:
                        if angle-0.05 >= 5.5:
                            angle = angle-0.05
                        else:
                            self.Dashboard.addInfo("Error! Too far")
                    self.Dashboard.calibration_value = round(angle,2)
                    self.setSteer(round(angle,2),True)
                if event.code == "BTN_SOUTH":
                    with open('settings.json', "w") as f:
                        json.dump({"angle": round(angle,2)}, f)
                    self.Dashboard.mode = 'User'
                    self.Dashboard.addInfo("Saved center steering value: {}".
format(round(angle,2)))
```

```
                f.close()
                self.center_steering = round(angle,2)
                return

    def setMotorSpeed(self,value,direction=1,duty=False):
        if not duty:
            #Adjustment to offset the limitation of L298N
            min_duty = 35
            #Normalizing the controller value to [0,1]
            val = value/255
            val = val*self.speed_limit
            speed = round((val*(100-min_duty))+min_duty)
            if value<2:
                #if the controller input is low stop the motor
                speed = 0
        else:
            speed = value
        if self.motor_lock:
            speed = 0
        #If speed is 0 then break the motor
        if speed == 0:
            #L298N overheats when breaking motor with both pin in LOW
            #GPIO.output(self.in3_pin,GPIO.LOW)
            #GPIO.output(self.in4_pin,GPIO.LOW)
            self.motor.ChangeDutyCycle(0)
            self.diraction = 'N'
        else:
            #Forwards
            if direction:
                GPIO.output(self.in3_pin,GPIO.LOW)
                GPIO.output(self.in4_pin,GPIO.HIGH)
                self.diraction = 'F'
            #Backwards
            else:
                GPIO.output(self.in3_pin,GPIO.HIGH)
                GPIO.output(self.in4_pin,GPIO.LOW)
                self.diraction = 'B'
            self.motor.ChangeDutyCycle(speed)
            # %of throttle speed

        self.speed = speed
        self.__dashboard__()

    def setSpeedLimit(self,value):
        #%of available power
        if value<=1:
            self.speed_limit = value
            self.Dashboard.addInfo("Speed limit set to: {}".format(value))
        else:
            self.Dashboard.addInfo("Error! Speed limit accepts values [0,1]")
```

```
    def getStatus(self):
        status = [self.diraction,self.speed,self.steer_side,self.angle]
        return status

    def turnOffCar(self):
        #visual feedback for turning off
        self.setSteer(self.center_steering-1,True)
        time.sleep(0.5)
        self.setSteer(self.center_steering,True)
        time.sleep(0.5)
        self.setSteer(self.center_steering+1,True)
        time.sleep(0.5)
        self.setSteer(self.center_steering,True)
        self.Dashboard.addInfo("Turning off the car")
        time.sleep(0.5)
        self.setSteer(0,True)
        self.setMotorSpeed(0,True)
        self.motor.stop()
        self.servo.stop()
        GPIO.cleanup()
        self.Dashboard.addInfo("Car turned off")
```

1.4 Dashboard.py

Σε αυτό το αρχείο υπάρχει η κλάση «Ταμπλό» (Dashboard). Αυτή η κλάση διαχειρίζεται την εμφάνιση όλων των πληροφορίων από το πρόγραμμα. Η κύρια λειτουργία της είναι να διατηρεί και να εμφανίζει στην κονσόλα τις 10 τελευταίες κρίσιμες – σημαντικές πληροφορίες του προγράμματος.

```
from os import system
import time


class Dashboard:
    def __init__(self):
        self.mode = 'User'
        self.diraction = 'N'
        self.speed = 0
        self.angle = 6
        self.calibration_value = 6
        self.auto_pilot_msg = ''
        self.info_list = []
        self.display()

    def addInfo(self, info):
        list_len = 10
        if len(self.info_list)>list_len:
            self.info_list.pop(0)
```

```
        self.info_list.append(info)
        self.display()

    def display(self):
        #Poor performance when using system clear
        #Increases latency
        #system('clear')
        str_to_print = "MODE: {} \nSPEED: {}-
{} ANGLE: {}".format(self.mode,self.diraction,self.speed,self.angle)
        print(str_to_print)
        if self.mode == 'Calibration':
            print("Center steering value: {}".format(self.calibration_value))
        if self.mode == 'Auto Pilot':
            print(self.auto_pilot_msg)
        else:
            print("*****Info*****")
            for info in self.info_list:
                print(info)
```

1.5 Driver.py

Σε αυτό το αρχείο υπάρχει η κλάση «Οδηγός» (Driver). Αυτή η κλάση διαχειρίζεται την αυτόνομη πλοήγηση του οχήματος. Προβλέπει την κλάση στην οποία ανήκουν οι εικόνες και ρυθμίζει ανάλογα την ταχύτητα και το τιμόνι.

```
from Car import Car
from Dashboard import Dashboard
from Camera import Cam
from tflite_runtime.interpreter import Interpreter
import time
import cv2
import numpy as np


class Driver:
    def __init__(self,Car,dashboard):
        #Setting up the car
        self.PiCar = Car
        self.Dashboard = dashboard
        self.auto_pilot = True

    #Initialize Auto Pilot
    def __start__(self):
        self.Dashboard.mode = "Auto Pilot"
```

```
        self.Dashboard.addInfo("Auto pilot starting")
        self.PiCam = Cam(self.Dashboard)
        self.class_names = ['L', 'M', 'N', 'R']
        #Load the TFLite model
        self.interpreter = Interpreter(model_path='type3.tflite')
        self.interpreter.allocate_tensors()
        self.input_details = self.interpreter.get_input_details()
        self.output_details = self.interpreter.get_output_details()

    #Predict a frame
    def __predictDiraction__(self):
        #Get a frame from the camera
        frame = self.PiCam.getFrame()
        #Prepare the image array for the model
        frame = cv2.resize(frame,dsize=(85,85), interpolation = cv2.INTER_CUB
IC)
        img_array = np.expand_dims(frame, 0) # Create a batch
        img_array = np.float32(img_array)
        #Get the predictions
        self.interpreter.set_tensor(self.input_details[0]['index'],img_array)
        self.interpreter.invoke()
        self.score = self.interpreter.get_tensor(self.output_details[0]['inde
x'])
        #Get the predicted diraction and steer
        self.go = self.class_names[np.argmax(self.score)]
        self.Dashboard.auto_pilot_msg = "This image most likely belongs to {}
 with a {:.2f} percent confidence".format(self.class_names[np.argmax(self.sco
re)], 100 * np.max(self.score))

    def __stop__(self):
        self.PiCar.setMotorSpeed(0,1,duty=True)
        self.PiCar.setSteer(a,duty=True)
        self.PiCam.stop()
        self.Dashboard.addInfo("Auto pilot stopped")

    def autoPilot(self):
        self.__start__()
        start = time.time()
        i = 0
        center_steering = self.PiCar.center_steering
        while self.auto_pilot:
            self.__predictDiraction__()
            confidence = np.amax(self.score)
            #Throttle based on confidence when turning
            throttle = 70 - (10*confidence)
            if self.go == 'M':
                self.PiCar.setSteer(center_steering,duty=True)
                throttle = 75
            elif self.go == 'L':
                #Set the steering based on the confidence
```

```
                angle = round(1.2*confidence,2)
                self.PiCar.setSteer(center_steering+angle,duty=True)
            elif self.go == 'R':
                #Set the steering based on the confidence
                angle = round(1.2*confidence,2)
                self.PiCar.setSteer(center_steering-angle,duty=True)
            else:
                #Out of track
                throttle = 0
                slow_down = 0
            throttle = round(throttle)
            self.PiCar.setMotorSpeed(throttle,1,duty=True)
            i += 1
        finish = time.time()
        self.Dashboard.mode = "User"
        self.Dashboard.addInfo("Captured {} frames at {:.2f}fps".format(i,i /
 (finish - start)))
        self.__stop__()
```

2.ΠΗΓΑΙΟΣ ΚΩΔΙΚΑΣ ΓΙΑ ΤΟ ΜΟΝΤΕΛΟ

2.1 Image-preprocessing.py

Ο κώδικας σε αυτό το αρχείο παίρνει όλες τις εικόνες μέσα από έναν φάκελο και τους υποφακέλους του, έπειτα με βάση το όνομα της εικόνας ξεχωρίζει και ταξινομεί τις εικόνες σε φακέλους. Το όνομα της εικόνας περιγράφει την κατάσταση ελέγχου του οχήματος την στιγμή της λήψης της εικόνας. Οι φάκελοι αντιστοιχούν στην κλάση οπού ανήκει η κάθε εικόνα. Επίσης πρέπει να προσθέσουμε έναν φάκελο με το όνομα «N» μέσα στον φάκελο ready οπού εκεί θα βάλουμε φωτογραφίες που καταγραφήκαν εκτός πίστας.

```
import os
import random
from PIL import Image
dirName = '...Path_to_images'

# Iterate over all the entries
i = 0
def getListOfFiles(dirName):
    global i
    # create a list of file and sub directories
    # names in the given directory
    listOfFile = os.listdir(dirName)
    allFiles = list()
    for entry in listOfFile:
        # Create full path
        fullPath = os.path.join(dirName, entry)
        # If entry is a directory then get the list of files in the directory
```

```
        if os.path.isdir(fullPath):
            allFiles = allFiles + getListOfFiles(fullPath)
        else:
            im1 = Image.open(fullPath)
            #img name example: img234_N_0_R_100_.jpg
            #1:diraction, 2:speed, 3:side, 4:angle
            l,diraction,speed,side,angle,extension = entry.split("_")
            speed ,angle = int(speed), int(angle)
            if speed+angle > 0:
                im1.save('...Path_to_save_images/training/ready{}/{}-
{}'.format(side,i,entry))
            i+=1
            allFiles.append(entry)

    return allFiles

dirs = ['L','M','R']
try:
    for d in dirs:
        os.mkdir('training/ready/{}'.format(d))
finally:
    listOfFiles = getListOfFiles(dirName)
```

2.2 model.ipynb

Ο κώδικας σε αυτό το αρχείο έχει να κάνει με το την αρχικοποίηση, εκπαίδευση και μετατροπή του μοντέλου μηχανικής μάθησης που τρέχει στον μικροελεγκτή.

```
import numpy as np
import os
import PIL
import PIL.Image
import tensorflow as tf
import pathlib
from tensorflow.keras import layers
from tensorflow import keras

#Get the images
data_dir = pathlib.Path('...Path_to_imgs/training/ready')
image_count = len(list(data_dir.glob('*/*.jpg')))

#Make training and validation datasets
batch_size = 10
img_height = 85
img_width = 85

train_ds = tf.keras.preprocessing.image_dataset_from_directory(
    data_dir,
    validation_split=0.3,
```

```
    subset="training",
    seed=123,
    interpolation="bicubic",
    image_size=(img_height, img_width),
    batch_size=batch_size)

val_ds = tf.keras.preprocessing.image_dataset_from_directory(
    data_dir,
    validation_split=0.3,
    subset="validation",
    seed=123,
    interpolation="bicubic",
    image_size=(img_height, img_width),
    batch_size=batch_size)

class_names = train_ds.class_names

#Initialize and make the model
model = tf.keras.Sequential([
    layers.experimental.preprocessing.Rescaling(1./255,input_shape=(img_heigh
t, img_width, 3)),
    layers.experimental.preprocessing.RandomContrast(0.1),
    layers.Conv2D(6, (3,3),strides=1, activation='relu',use_bias=True),
    layers.MaxPooling2D(2,2),
    layers.Conv2D(8, (3,3), activation='relu',use_bias=True),
    layers.MaxPooling2D(2,2),
    layers.Conv2D(14, (3,3), activation='relu',use_bias=True),
    layers.MaxPooling2D(2,2),
    layers.Flatten(),
    layers.Dropout(0.1),
    layers.Dense(40, activation='relu'),
    layers.Dense(40, activation='relu'),
    layers.Dense(4, activation='softmax')
])
optimizer = tf.keras.optimizers.Adam(learning_rate = 0.0003)
model.compile(
  optimizer = optimizer,
  loss = 'sparse_categorical_crossentropy',
  metrics=['accuracy'])

model.summary()

#Make logs for TensorBoard
path = '...Path_to/logs'
tensorboard_callback = tf.keras.callbacks.TensorBoard(log_dir=path, histogram
_freq=1)

#Make model checkpoints
#Save the model at the best performing epoch
checkpoint_filepath = '...Path_to/checkpoint'
```

```
model_checkpoint_callback = tf.keras.callbacks.ModelCheckpoint(
    filepath=checkpoint_filepath,
    save_weights_only=False,
    monitor='val_loss',
    mode='min',
    save_best_only=True)

#Train the model
history = model.fit(
  train_ds,
  validation_data=val_ds,
  batch_size=500,
  epochs=11,
  callbacks=[model_checkpoint_callback,tensorboard_callback]
)

#Load and start TensorBoard
%load_ext tensorboard
%tensorboard --logdir logs

#Load the best model from checkpoint
model = tf.keras.models.load_model('checkpoint')

#Convert model to TFLite model
converter = tf.lite.TFLiteConverter.from_keras_model(model)
converter.optimizations = [tf.lite.Optimize.DEFAULT]
tflite_model = converter.convert()
open("type3.tflite", "wb").write(tflite_model)
```

ΒΙΒΛΙΟΓΡΑΦΙΚΕΣ ΑΝΑΦΟΡΕΣ

[1] Gentle Dive into Math Behind Convolutional Neural Networks https://towardsdatascience.com/gentle-dive-into-math-behind-convolutional-neural-networks-79a07dd44cf9

[2] Understanding the math behind Neural Networks https://medium.com/dataseries/understanding-the-maths-behind-neural-networks-108a4ad4d4db

[3] Machine learning course syllabus https://www.coursera.org/learn/machine-learning?#syllabus

[4] How Do Convolutional Layers Work in Deep Learning Neural Networks? https://machinelearningmastery.com/convolutional-layers-for-deep-learning-neural-networks/

[5] Tensorflow2 API Documentation https://www.tensorflow.org/api_docs/python/tf/all_symbols

[6] Raspberry Pi Documentation https://www.raspberrypi.org/documentation/