ΕΡΓΑΣΙΑ:

Αυτόνομο Όχημα- Ακολούθηση Γραμμής & Αποφυγή Εμποδίων

Ομάδα:Καραγεωργιάδης Αναστάσιος
email:akarageorgiadis@isc.tuc.gr

Πολυτεχνείο Κρήτης
Ενσωματωμένα Συστήματα Μικροεπεξεργαστών
Καθ.Απόστολος Δόλλας
2017

1.|ΕΙΣΑΓΩΓΗ:

1.1|Σκοπός της εργασίας:

Σκόπος αυτής της εργασίας είναι η κατασκευή ενός αύτονομου οχήματος με χρήση του *lego ev3 mindstorm* μικροελεγκτή και του πακέτου *tetrix*, το οποίο θα εκτελεί ακολούθηση γραμμής και αποφυγή εμποδίων. Στόχος αυτής της εργασίας είναι η περεταίρω εξοικείωση με τον μικροελεγκτή, η απόκτηση εμπειρίας και η δημιουργία της βάσης του κώδικα που θα χρειαστεί για τον διαγωνισμό WRO.

1.2|Υλικά Μέρη που χρησιμοποιήθηκαν:

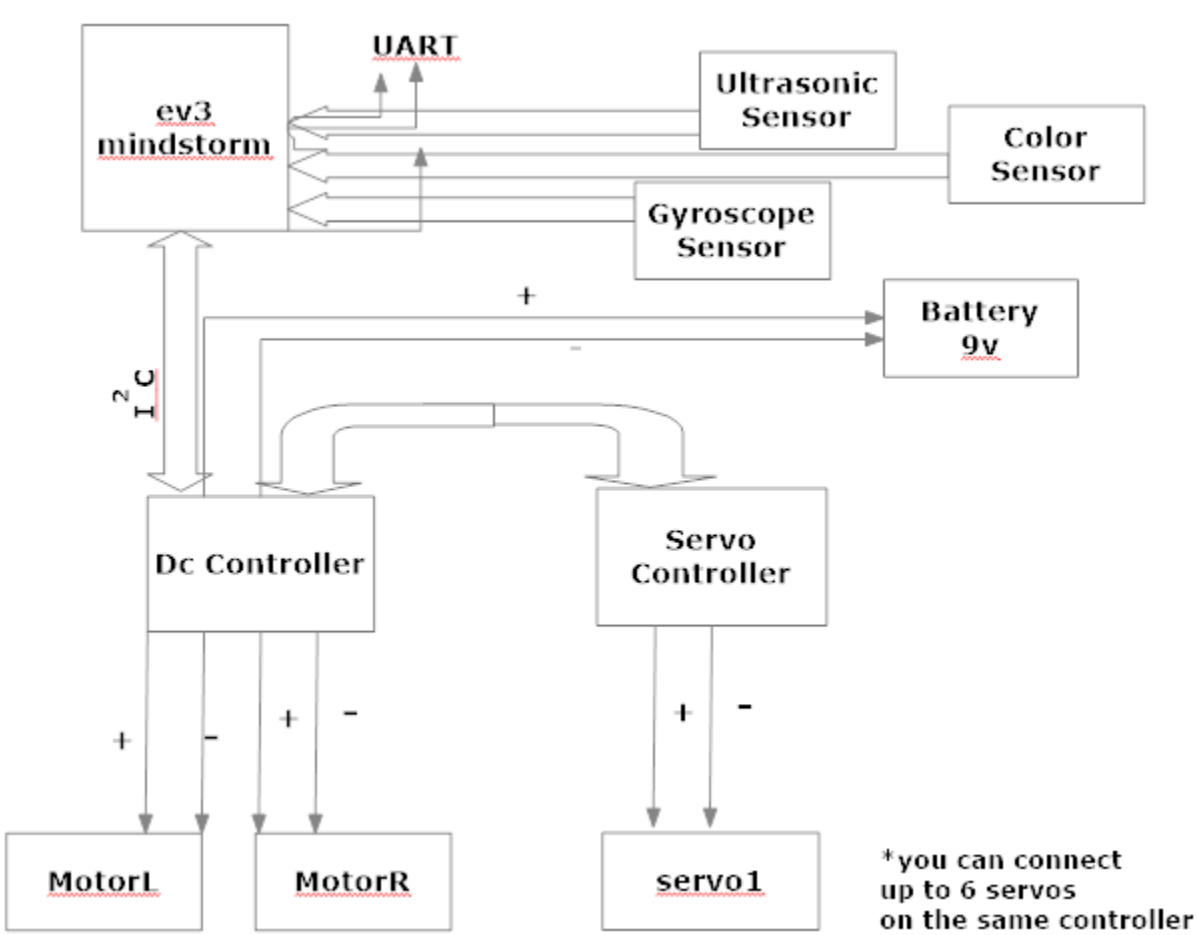
- ➔ Lego Ev3 Mindstorm
- ➔ Lego color sensor
- ➔ Lego ultrasonar sensor
- ➔ Lego gyroscope sensor
- ➔ Hitechnic DC gear motors 12v
- ➔ HSB-480HS servo motor
- ➔ Hitechnic Dc & Servo controllers
- ➔ Metal & mechanical parts from Tetrix

1.3|Λειτουργικό/Λογισμικό :

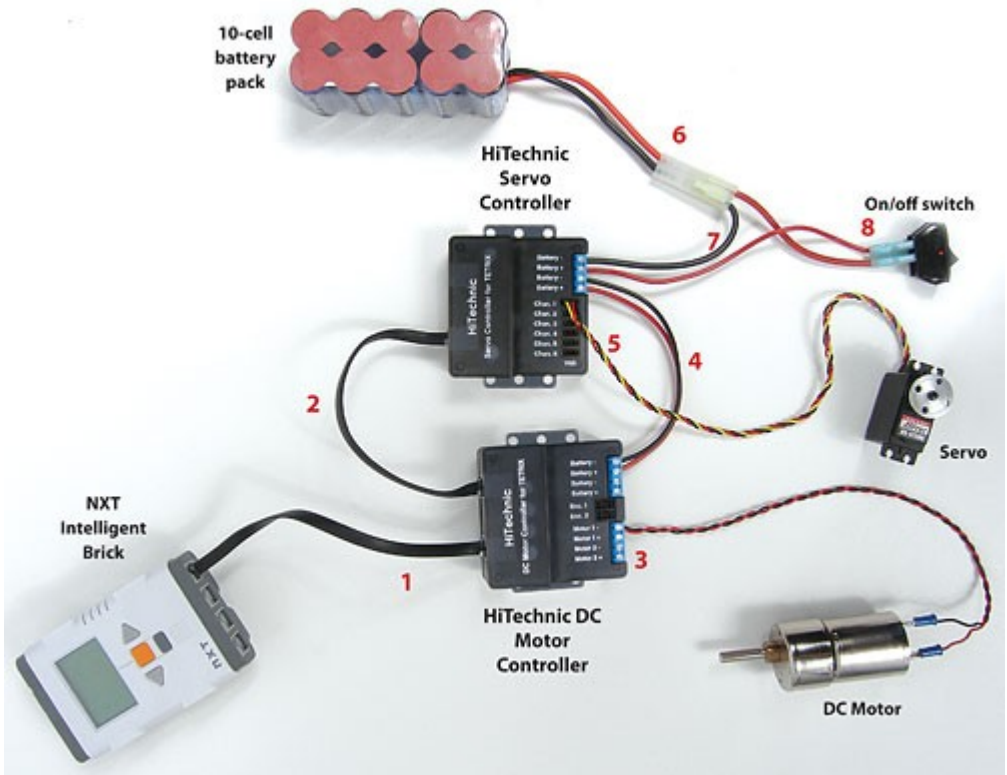
Για τις ανάγκες τις εργασίας δημιουργήκε μία micro-sd κάρτα μνήμης με το λειτουργικό σύστημα *“ev3dev”*, το οποίο είναι μια διανομή linux debian προσαρμοσμένη κατάλληλα για τον μικροελεγκτή. Με τη χρήση αυτού του λειτουργικού συστήματος, η ανάπτυξη της υλοποίησης έγινε σε γλώσσα C· προκειμένου ο κώδικας να είναι μεταφέρσιμος και σε άλλους μικροελεκτές παρόμοιων χαρακτηριστικών (επεξεργαστής ARM9).

2.|ΑΝΑΛΥΣΗ ΕΡΓΑΣΙΑΣ:

2.1.1|Συνδεσμολογία Υλικού:



Σχηματικό Διάγραμμα



Εικόνα2.1

2.1.2|Περιγραφή λειτουργίας υλικού:

Ο μικροελεγκτής μας επικοινωνεί με τις συσκευές του (dc motors,servos κτλπ) μέσω ενός διαύλου τύπου I2C. Ανάμεσα στις συσκευές και τον μικροελεγκτή παρεμβάλεται ένας ή περισσότεροι ελεγκτές,σε σειρά, στους οποίους συνδέονται τα αντίστοιχα μοτερ (βλ.*εικονα2.1*). Έτσι σε πρώτη φάση έχουμε μία διεύθυνση για τον κάθε ενδιάμεσο ελεγκτή("slave") για να μπορούμε να επιλέξουμε με ποιον ελεγκτή επικοινωνούμε. Επέιτα κάθε τέτοιος ελεγκτής έχει μια δεύτερη διεύθυνση,που αντιστοιχεί στις επιμέρους συσκεύες και γράφοντας σειριάκα τα δεδομένα μας σε αυτήν, καταφέρνουμε να ελέγξουμε τα μοτερ μας(*περισσότερα βλ.Παράρτημα 3*).

Όσον αφορά του αισθητήρες αυτοί επικοινωνούν με την χρήση κατάλληλων drivers του λειτουργικού και ουσιαστικά γίνεται *εγγραφή/ανάγνωση* σε συγκεκριμένα αρχεία συσκευών("devices' files").

Ενδεικτικά (κώδικας 2.1.2)

Ψευδοκώδικας για τον χειρισμό μοτερ:

```
ioctl(fileD,i2c_slave,slave_address);
buffer[size]={device_address,..data..};
write(fileD,buffer,size);
```

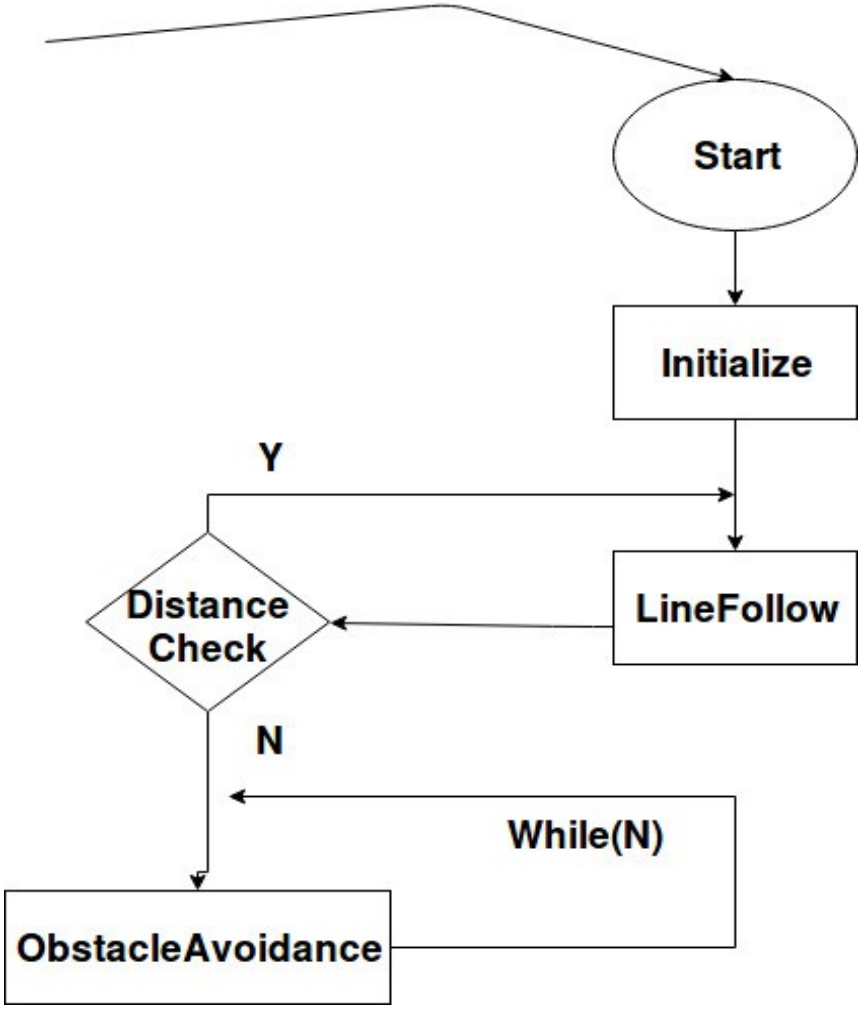
Ψευδοκώδικας για το χειρισμό των αισθητήρων:

```
fileN=open("/sys/class/lego-sensor/sensorN/value0","r");
read_value=readData(fileN);
```

2.1.3|Λειτουργία DC motors:

Η περιστροφή τών dc motors γίνεται γράφοντας στον παραπάνω buffer*(κώδικα2.1.2)* τις ζητούμενες τιμές *PWM(Pulse Width Modulation)* για το κάθενα, με προσοχή στο πρόσιμο αυτών των τιμών για να συγχρονίζονται και να κινούνται στην ίδια κατεύθυνση. Δηλαδή για να περστρέφονται προς τα εμπρός τα μοτερ έπρεπε το *Motor1(δεξί)* να έχει θετική τιμή PWM ενώ το *Motor2(αριστερό)* έπρεπε να έχει αρνητική αλλά αυτή η τιμή ήταν η ίδια κατα απόλυτη τιμή*(κώδικας 2.3)*. Για την κίνηση προς τα πίσω αντιμεταθέτουμε αυτά τα πρόσιμα.

2.2|Λογισμικό-Υλοποίηση:



Σχηματικό Διάγραμμα-Γενική Δομή

2.3|Ακολούθηση Γραμμής:

Το πρόβλημα της ακολούθησης γραμμής είναι ένα κλασσικό πρόβλημα αυτόματου ελέγχου και υπάρχει πληθώρα ενδεικτικών λύσεων και διαγωνισμών αυτού του είδους. Στην δική μας περίπτωση χρησιμοποιήσαμε έναν μόνο αισθητήρα χρώματος με λειτουργία *IR*, για να μετράμε την ένταση του χρώματος της γραμμής και του δαπέδου, όπου κινείται το όχημα μας. Υλοποιώντας έναν ελεγκτή τύπου *PID (Propotional, Integral, Dirivative)* σε επίπεδο λογισμικού, διαμορφώναμε κατάλληλα την ταχύτητα των μοτερ ώστε το όχημα μας να κάνει τις απαραίτητες διορθώσεις και εν τέλει να ακολουθεί τη γραμμή.

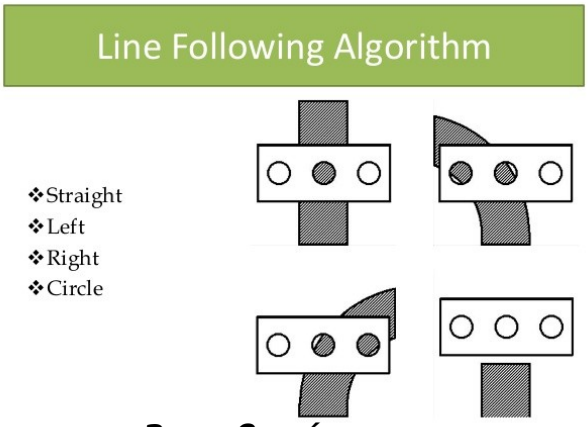
Ο διαχωρισμός των περιπτώσεων για το πότε το όχημα μας βρισκόταν πάνω στην γραμμή ή είχε βγεί αριστερά ή δεξιά,υλοποιήθηκε τοποθετώντας τον αισθητήρα χρώματος οριζόντια*(εικόνα 2.3)*.Έτσι πάνω στην γραμμή είχαμε τιμές έντασης *μεγαλύτερες ή ίσες του κατωφλίου* που θέσαμε,*αριστερά* είχαμε *τιμές μικρότερες* του κατωφλίου αλλά μεγαλύτερες του 35pct, ενώ *δεξιά* είχαμε *πόλυ μικρότερες* τιμές από το κατώφλι, της τάξης του 10pct. Μ'αυτό τον τρόπο καθορίζαμε που βρίσκεται το όχημα μας και το στρέφαμε προς την αντίθετη κατεύθυνση.

Για την εκτέλεση της στοφής διατηρούσαμε την ταχύτητα του μοτερ, της πλευράς που έχουμε ξεφύγει, σταθερή και μειώναμε την ταχύτητα του μοτερ της άλλης πλεύρας κατα έναν παράγοντα *turn_factor* που προέκυπται από τον *PID*,τέτοιο ώστε να καθορίσουμε πόσο μεγάλη ή μικρή θα είναι αυτή η στροφή.

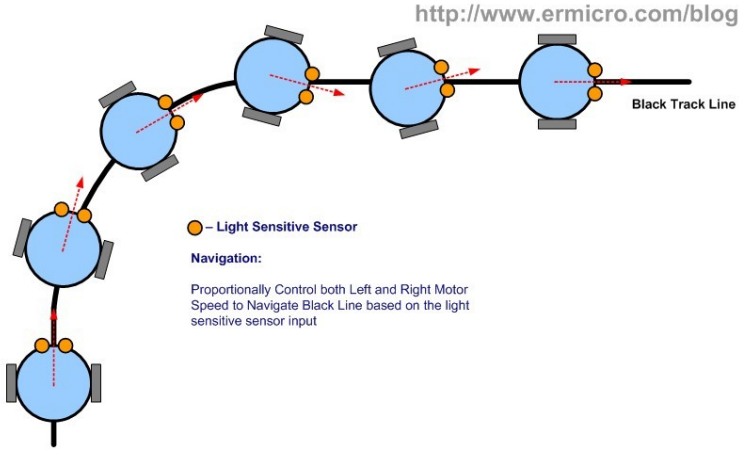
Φορά τοποθέτησης αισθητήρα χρώματος

(εικόνα 2.3)

Εικόνες Υλοποίησης ανάλογα με το πλήθως των αισθητήρων χρώματος:

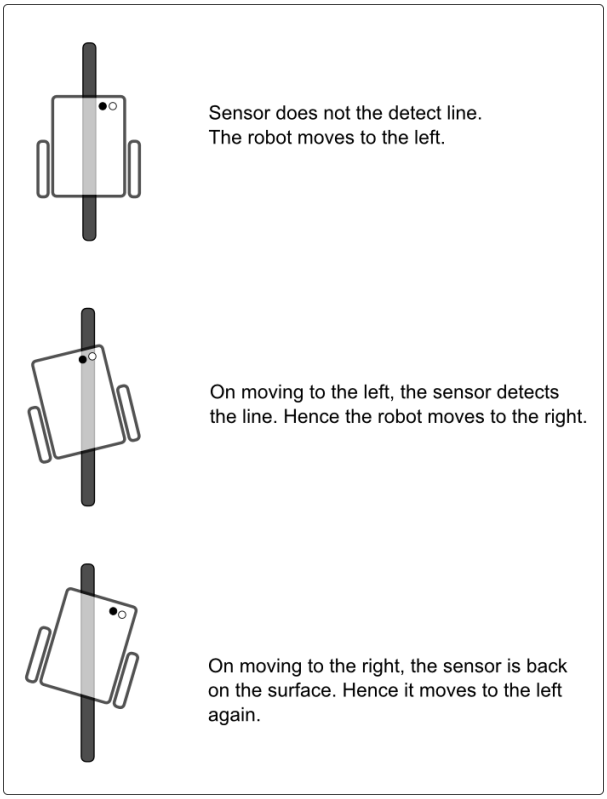


3 αισθητήρες



Line Tracking Navigation Principle on The Line Follower Robot (LFR)

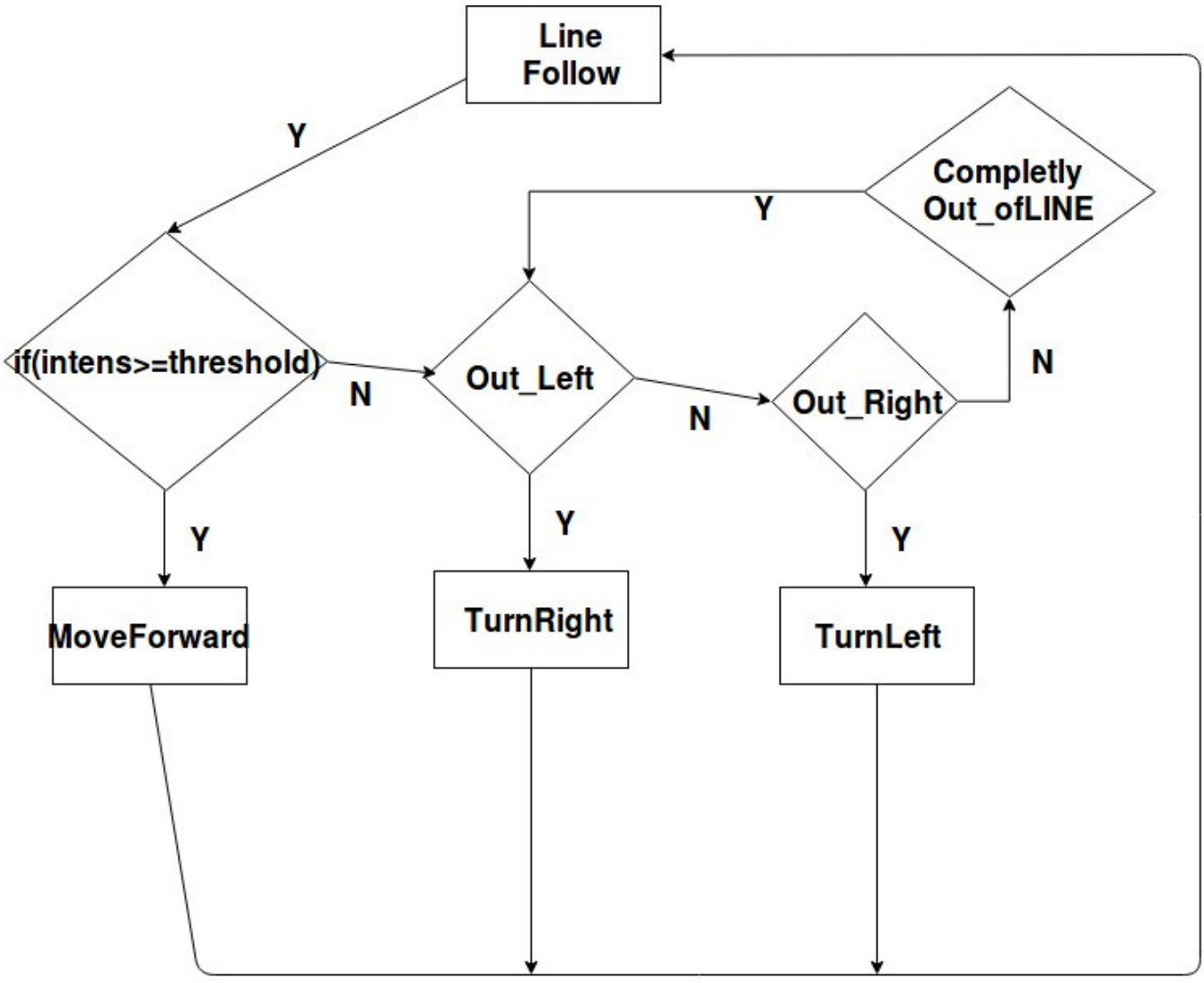
2 αισθητήρες



1 αισθήτηρας

Ψευδοκώδικας(κώδικας 2.3):

```
intensity_threshold=76; //white color line
error=integral=dirivative=last_error=0;
Kp=;
Ki=;
Kd=;
while(){
        intensity_value=take_measurement(); //measure intensityfrom sensor
        error=intensity_treshold-intensity_value;
        integral+=error;
        dirivative=error-last_error;
        last_error=error;
        turn_factor=Kp*error+Ki*integral+Kd*dirivative;
        run(turn_factor*RmotorSpeed,-turn_factor*LmotorSpeed);//move motor
        ...
        if (left_out)
                run(turn_factor*RmotorSpeed,-LmotorSpeed); ….
…..
}
```

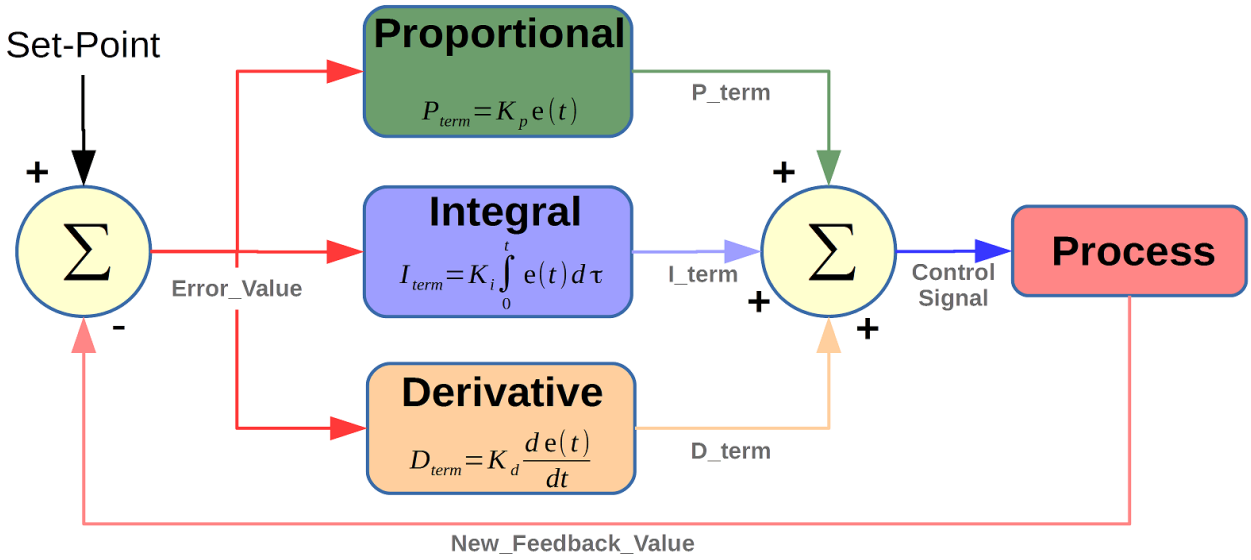


Σχηματικό Διάγραμμα-Ακολούθηση Γραμμής

2.4|Δυσκολίες/Προβλήματα:

Οι δύο μεγάλες δυσκολίες αυτού του προβλήματος μέσα από τις μεθόδους που χρησιμοποιήσαμε ήταν, πρώτον η επιλογή των ιδανικών παραμέτρων Kp, Ki, Kd για τον PID ελεγκτή μας και δεύτερον ο περιορισμός του ενός αισθητήρα χρώματος. Οπώς φαίνεται από τις εικόνες παραπάνω στις περισσότερες μεθόδους ακολούθησης γραμμής γίνεται χρήση *τουλάχιστον* δυο αισθητήρων χρώματος· κάτι που διευκολύνει την στροφή του οχήματός μας σε περιπτώσεις κάθετων διασταυρώσεων (στροφών 90° γενικότερα) σε αντίθεση με την δική μας υλοποίηση οπού χρησιμοποιήσαμε μόνο ένα τέτοιο αισθητήρα.

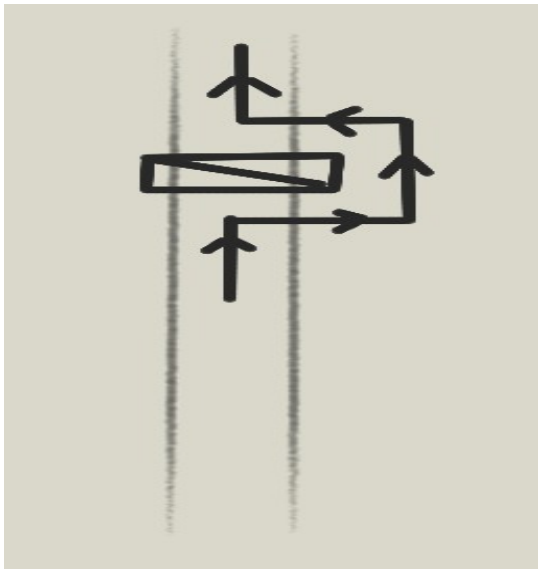
Η δυσκολία του υπολογισμού των παραμέτρων από την άλλη, είναι συνεθισμένη σε τέτοιου είδους προβλήματα και η εύρεση τους στηρίζεται κυρίως στην πολύωρη μοντελοποίηση και τον πειραματισμό για την εκάστοτε περίπτωση.
Για να προσσεγγίσουμε τις τίμες για αύτες τις παραμέτρους, κατασκευάστηκε ένα κομμάτι κώδικα σε C, με το οποίο, παράγοντας κάποιες τυχαίες ακέραιες τιμές, που αντιστοιχούσαν στο εύρος των μετρημένων εντάσεων χρώματος του χώρου μας(*testPID.c, βρίσκεται εντός του github repository*).

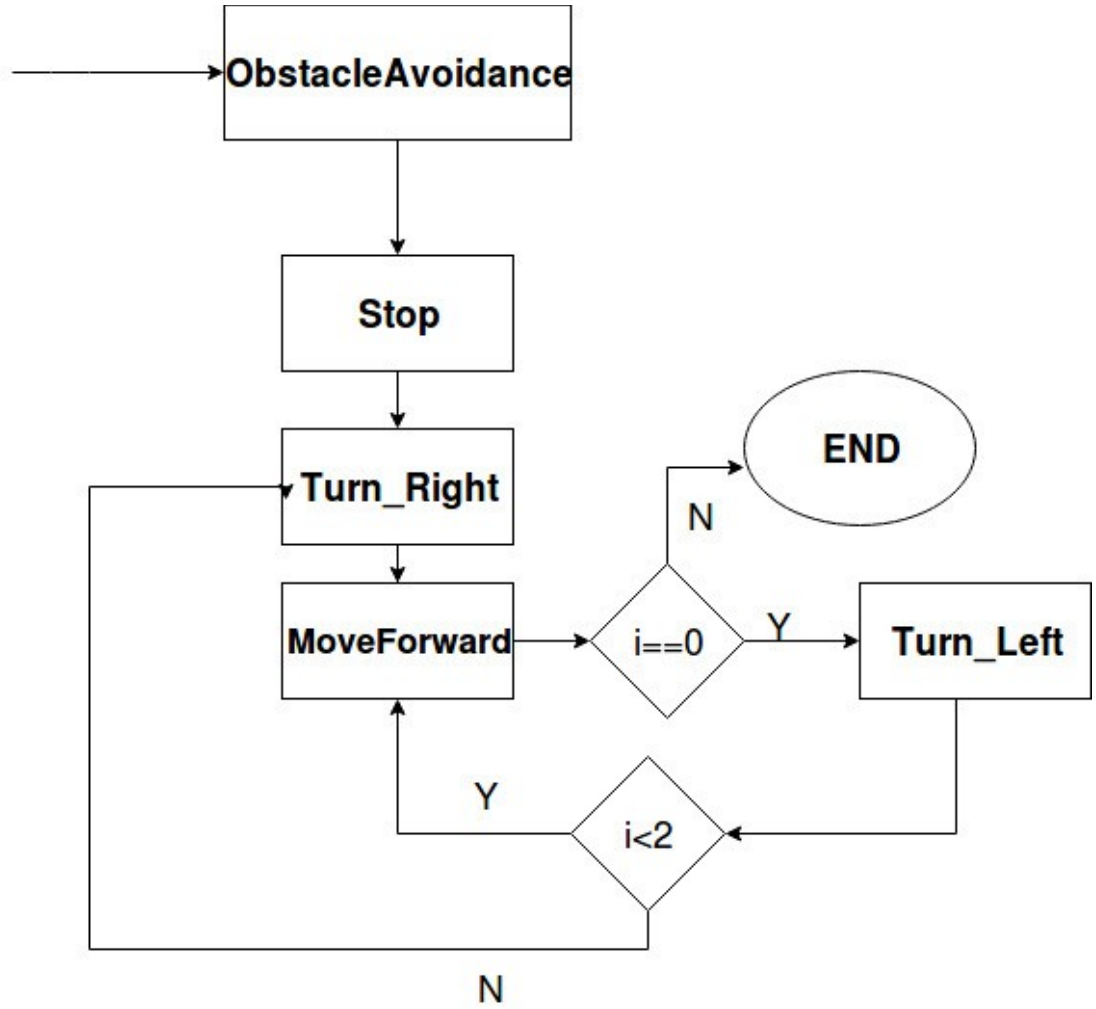


2.4|Αποφυγή Εμποδίων:

Το πρόβλημα της αποφυγής εμποδίων λύθηκε με μια απλή προσέγγιση σύμφωνα με την οποία κάθε φορά που το όχημα μας συναντούσε ένα εμπόδιο, έμπαινε σε μια ρουτίνα και εκτελούσε έναν κύκλο από τα δεξιά και επέστρεφε στην γραμμή. Για το συγκεκριμένο πρόβλημα έγιναν αρκετές απλοποιήσεις ώστε να είναι εφικτό να πραγματοποιηθεί στα χρονικά πλαίσια της εργασίας.

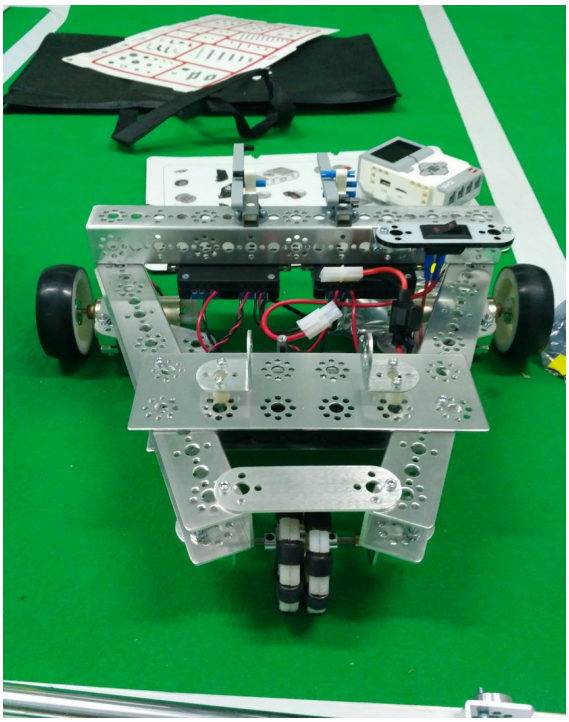
Σχηματική Περιγραφή Υλοποίησης

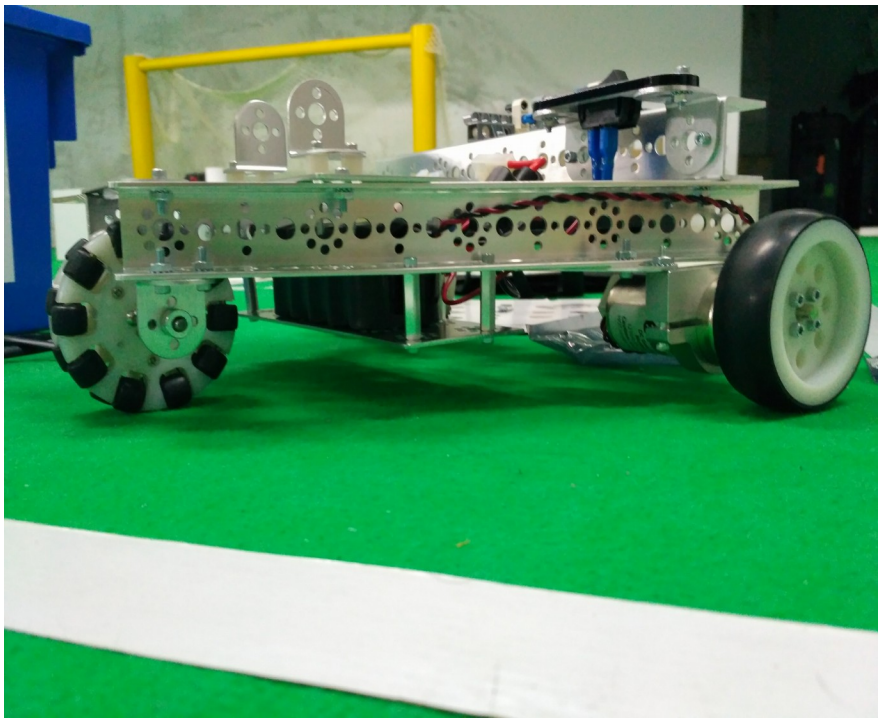


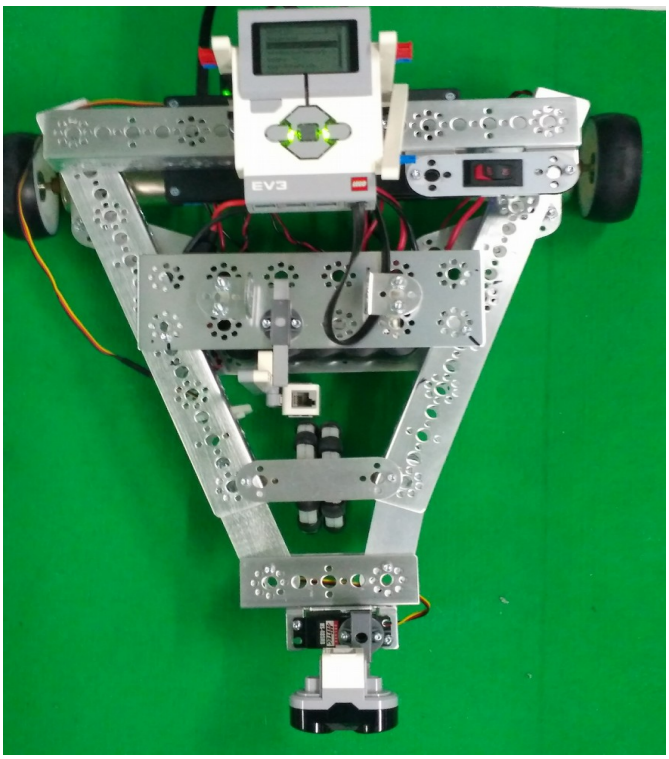


Σχηματικό Διάγραμμα-Αποφυγή Εμποδίων

Εικόνες Οχήματος-Κατασκευής:







ΠΑΡΑΡΤΗΜΑ:

(1)Github Repository of Project:
https://github.com/soylisK/ev3c_LFOA/tree/master

(2)EV3DEV OS documentation & tutorials:
https://www.ev3dev.org

(3)Hitechnic Documentation:
http://www.hitechnic.com/blog/wp-content/uploads/HiTechnic-Motor-Controller-Specification.pdf

http://www.hitechnic.com/blog/wp-content/uploads/HiTechnic-Servo-Controller-Specification.pdf

(4)Tetrix Examples:
http://www.openrtm.org/openrtm/ja/content/tetrix_ev3

Παροχή ev3 mindstorm και tetrix kit εργαστήριο "Κουρήτες"