ΑΝΩΤΑΤΟ ΕΚΠΑΙΔΕΥΤΙΚΟ ΙΔΡΥΜΑ ΠΕΙΡΑΙΑ

ΤΕΧΝΟΛΟΓΙΚΟΥ ΤΟΜΕΑ

Εργαστήριο Μηχατρονικής		
Εργαστηριακή Άσκηση: Προσέγγιση σε Στόχο με Αυτοκινούμενο Όχημα		
Ομάδα:		
Ονοματεπώνυμο	**ΑΜ**	**Εξάμηνο**
Χριστιανίδης Βασίλειος	46289	6ο
Κριστιάν Σεράνι	46865	6ο
Λυτινάκης Αναστάσιος	44286	8ο
Τοπαρόπουλος Μάριος	43950	8ο

Περιεχόμενα

1. Εκφώνηση όπως δίνεται από τον καθηγητή:

2. ΠΡΟΣΕΓΓΙΣΗ ΣΕ ΣΤΟΧΟ

Αντικείμενο

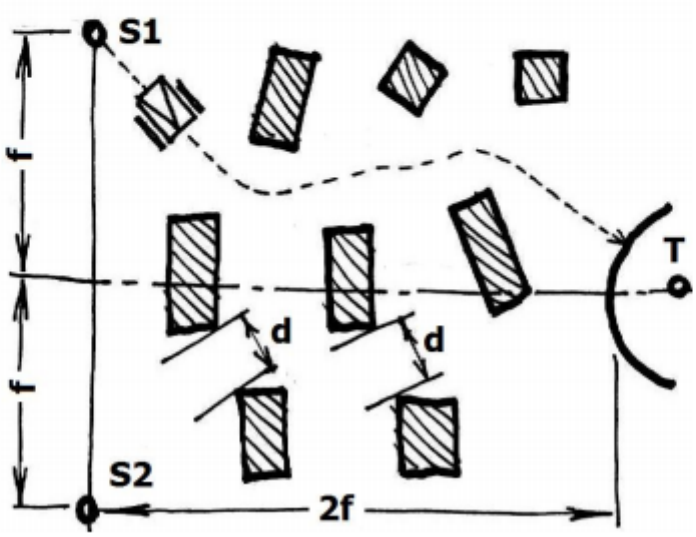
Η εργασία αφορά την κατασκευή και επίδειξη αυτοκινούμενου μικρο-οχήματος, η κίνηση του οποίου ελέγχεται από σύστημα μικροελεγκτή.
Το πεδίο δοκιμής είναι περίπου τετραγωνικής μορφής με πλευρά 10b, όπου b είναι το «πλάτος» του οχήματος, δηλαδή η απόσταση μεταξύ των κέντρων των κινητήριων τροχών.
Κατά την εκκίνηση, το όχημα βρίσκεται σε μια από δύο αφετηρίες S1 ή S2 και είναι προσανατολισμένο προς την αντίθετη αφετηρία (δηλαδή αν τοποθετηθεί στην S1, θα είναι στραμμένο ώστε η κατεύθυνση της κίνησής του να «κοιτάει» την S2).
Το πεδίο δοκιμής περιέχει μια περιοχή στόχου (T), από το κέντρο της οποίας εκπέμπεται ακτινικά υπέρυθρη ακτινοβολία, έτσι ώστε η κατεύθυνση του στόχου να εντοπίζεται από οποιαδήποτε θέση στο εσωτερικό του πεδίου δοκιμής: π.χ. από μια δέσμη από LED τοποθετημένα σε επιλεγμένο ύψος και στραμένα σε διαφορετικές διευθύνσεις.
Επίσης, το πεδίο δοκιμής περιέχει «εμπόδια» ορθογώνιας κάτοψης, τοποθετημένα ώστε να αφήνουν μεταξύ τους διάβαση με πλάτος $d \geq 1.5b$ και, επίσης, να μην εμποδίζουν (διακόπτουν) την ακτινοβολία για τον εντοπισμό του στόχου (στη βασική προδιαγραφή - βλ. πρόσθετη προδιαγραφή κατωτέρω).
Το Σχήμα παρουσιάζει μια ενδεικτική μορφή του πεδίου δοκιμής.

Προδιαγραφή

Μετά την εκκίνηση, το όχημα λειτουργεί αυτόματα (μπορεί να συνδέεται με καλώδιο μόνο για ηλεκτρική τροφοδοσία). Σκοπός της λειτουργίας είναι το όχημα να προσεγγίσει το στόχο, σε χρόνο μικρότερο από εκείνον που απαιτείται για να καλύψει απόσταση 30b. Το Σχήμα απεικονίζει ένα παράδειγμα επιτυχημένης πορείας, όπου το όχημα παρακάμπτει τα εμπόδια και φθάνει στο στόχο T.

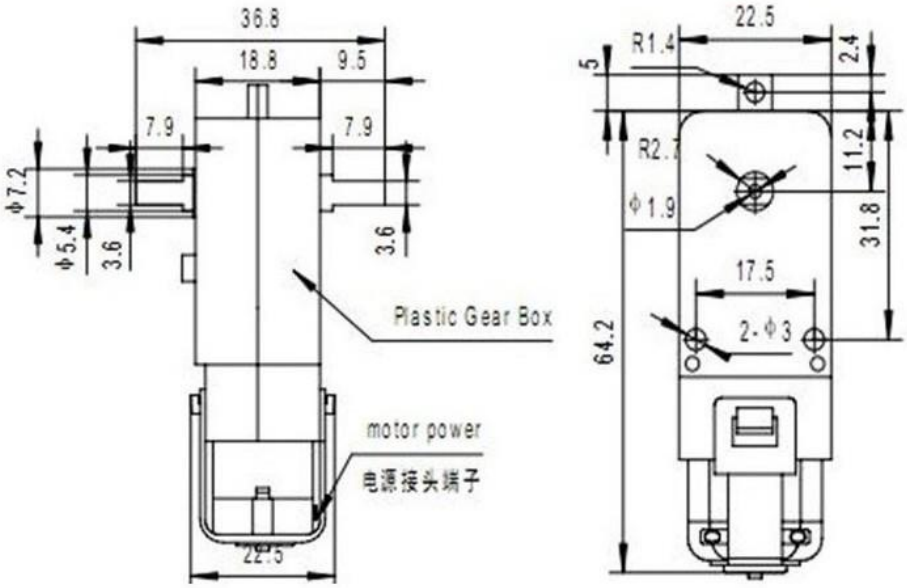
Πρόσθετη προδιαγραφή

Τα εμπόδια είναι στερεά με αρκετό ύψος ώστε τοπικά να «σκιάζουν» την ακτινοβολία του στόχου.



2. Hardware

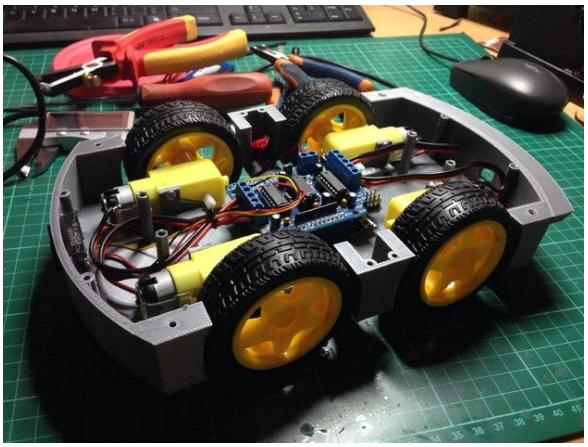
- Για το σασί χρησιμοποιήθηκαν τα 3d printed open source σχέδια από αυτή την ιστοσελίδα: https://www.thingiverse.com/thing:739024/#files
- Για ρόδες, μαζί με 4 dc motors with gears χρησιμοποιήθηκαν:



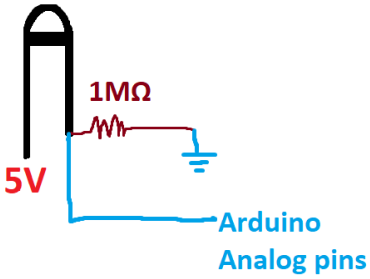
Motors with gears

Wheels

Ετσι δείχνει στο τέλος η κατασκευή.



- Για τους αισθητήρες απόστασης χρησιμοποιήσαμε τρεις HC-SR04, των οποίων το datasheet μπορεί να βρεθεί εύκολα.
- Για τους αισθητήρες του στόχου χρησιμοποιήσαμε τρία 3mm IR receivers, των οποίων το σχηματικό σύνδεσης δεν φαίνεται σε κάποιο datasheet οπότε πειραματιστήκαμε και το βρήκαμε μόνοι μας:

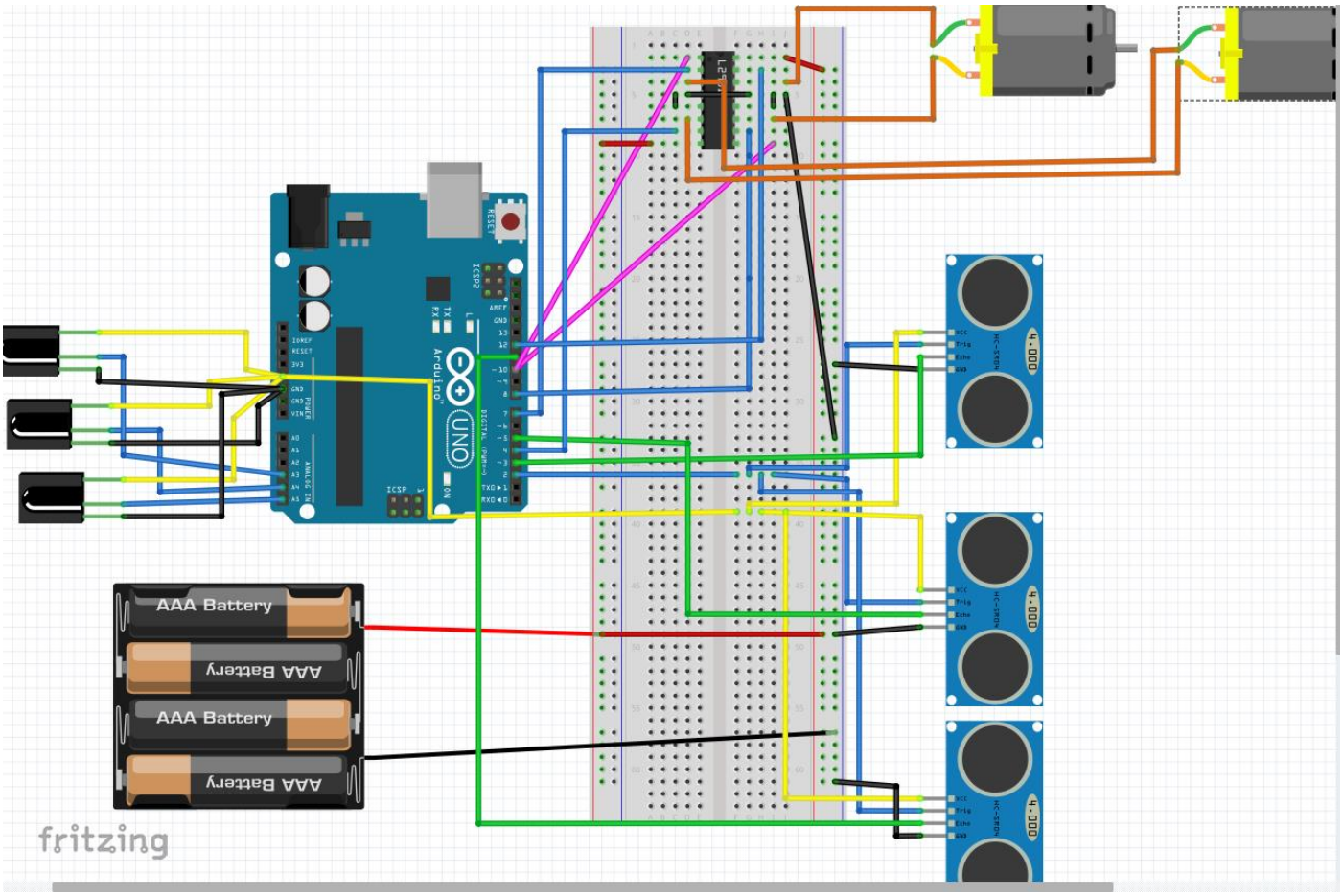


Από ότι καταλαβαίνετε, το χρησιμοποιούμε σαν αντίσταση για να φτιάξουμε διαιρέτη τάσης.

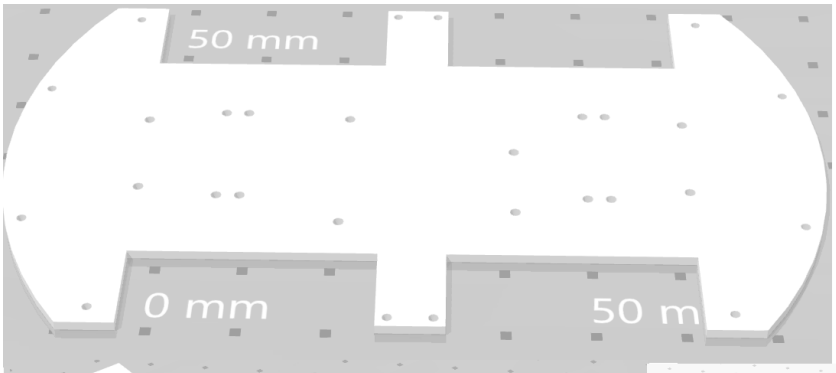
Να σημειωθεί ότι αυτά τα receivers ‘βλέπουν’ σε μια περιορισμένη γωνία μπροστά τους και όχι από οποιαδήποτε κατεύθυνση ‘πέσει το ir φως επάνω τους. Περισσότερες πληροφορίες μπορείτε να δείτε στο datasheet.

- Θα χρειαστεί επίσης ένας Atmega για να φορτωθεί και να λειτουργήσει ο κώδικας που φτιάξαμε.
- Κατά τη συναρμολόγηση προτείνουμε να χρησιμοποιηθούν βίδες με παξιμάδια και κόλλα σιλικόνης μετά το βίδωμα για καλύτερο κράτημα των βιδών πάνω στο σασί.
- Θα χρειαστεί ένα l293d για τον έλεγχο δύο από τις 4 ρόδες, καθώς το Arduino δεν έχει αρκετά pins για να υποστηρίξει όλες τις ρόδες (μαζί με τους αισθητήρες απόστασης και στόχου).
  Επίσης η χρήση και των τεσσάρων motor, θα έκανε τη κίνηση του αυτοκινούμενου οχήματος πιο γρήγορη και ίσως λιγότερη σταθερή και πιο δύσκολα ελέγξιμη. Ένα επιπλέον πλεονέκτημα σε αυτό, είναι ότι μειώνεται η περιπλοκότητα του κώδικα.

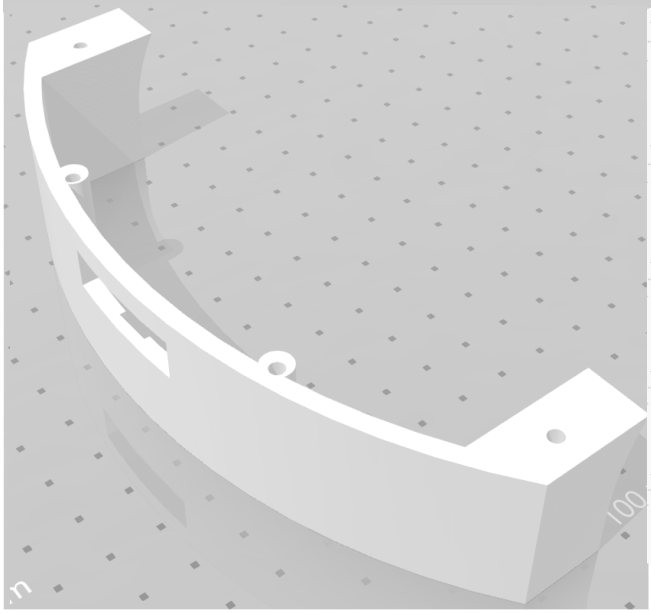
3. Schematic



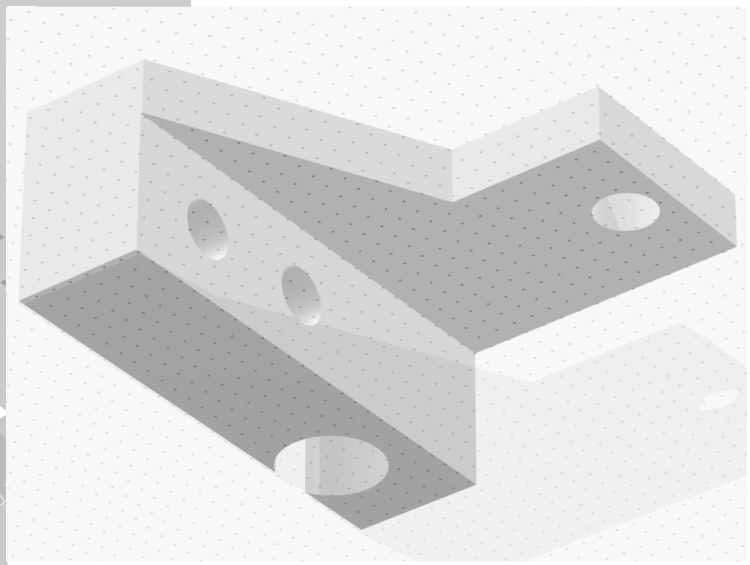
4. Περιληπτικά κομμάτια του σασί



Top/bottom



Front/back



Motor support

5. Περίληψη προγράμματος

Το πρόγραμμα υλοποιεί τη λειτουργία του αυτοκινουμένου οχήματος σε τέσσερα μέρη:

a) την κίνηση των τροχών του
   Εδώ, αξίζει να σημειωθεί ότι δεν χρησιμοποιήθηκαν όλοι οι τροχοί, (4 τροχοί), αλλά μόνο οι δυο πίσω. Για αυτό, το πρόγραμμα περιέχει σχόλια από το αρχικό πρόγραμμα, το οποίο χρησιμοποιούσε και τις 4 ρόδες, μαζί με το motor control ic circuit. Αυτά τα σχόλια μπορούν να διαγραφούν με ασφάλεια.
   Η κύρια συνάρτηση που κινεί τους τροχούς είναι η void Move(int xwheel, int ywheel, bool forward). Η οποία ως πρώτη και δεύτερη μεταβλητή παίρνει τις ρόδες η την ρόδα που θέλουμε να κινήσουμε (ρόδα 3 είναι η πίσω δεξιά, ενώ ρόδα 4 η πίσω αριστερά), ως πρώτη η δεύτερη μεταβλητή και ως τρίτη η forward, η οποία αν είναι true, η/οι αντίστοιχες ρόδες/-α θα κινηθεί μπροστά. Αν είναι false, θα κινηθεί πίσω.

b) την μέτρηση της απόστασης σε εκατοστά των αισθητήρων απόστασης που έχει (μπροστά αριστερά, μπροστά δεξιά και μπροστά ευθεία)
   Εδώ χρησιμοποιείται η συνάρτηση int Distance(char PinNum), η οποία παίρνει ως παράμετρο ,τον αισθητήρα απόστασης ο οποίος θα χρησιμοποιηθεί για να μετρηθεί η απόσταση. (‘r’ για το δεξί αισθητήρα, ‘l’ για τον left αισθητήρα και ‘m’ για τον μεσαίο αισθητήρα. Να σημειωθεί ότι τα triggers των αισθητήρων είναι κοινά. Δεν έχει αρκετά pins το uno για να βάλω ξεχωριστά triggers. Κάθε φορά που θέλω να πάρω μέτρηση μέσω της συνάρτησης για την απόσταση, η συνάρτηση Distance στέλνει παλμό (trigger) σε όλους τους αισθητήρες, όμως αντίστοιχα με το ποιο αισθητήρα θέλουμε να δούμε την έξοδο του (echo),αυτουνού τη τιμή παίρνουμε, παρόλο που τους ενεργοποιούμε όλους με τη trig. Αυτό δεν φαίνεται να επηρεάζει τις μετρήσεις μετά από κάποιους πειραματισμούς που κάναμε.

c) Την μέτρηση της δύναμης του υπεριώδους (infrared) φωτός που λαμβάνει από τους τρεις αισθητήρες infrared φωτός που έχει, ο κάθε ένας αισθητήρας ir είναι κάτω από το αντίστοιχο αισθητήρα απόστασης.
   Η συνάρτηση int Av_sens(char sens, int M=20) παίρνει έναν αισθητήρα ως πρώτη παράμετρο (l,r,m) όπως και με τους αισθητήρες απόστασης. Επίσης παίρνει μια τιμή M σαν παράμετρο, η οποία δείχνει πόσες τιμές θα πάρει σαν δείγμα για να βγάλει το μέσο ορό των μετρήσεων M που πηρέ. Επιστρέφει μια τιμή αναμεσά στο 0-1023

d) Τη βαθμονόμηση της ποσότητας του ir φωτισμού που δέχεται (με τον στόχο αναμένον και με το στόχο σβηστό) για να καθορίσει τη φωτεινότητα του δωμάτιου και τη φωτεινότητα του στόχου σε σχέση με το δωμάτιο.

Τέλος, συνδυάζει όλα τα παραπάνω στη void loop για να κινηθεί στο χώρο και να προσεγγίσει το στόχο.

6. Manual

Σχετικά με τη τελευταία λειτουργία του αυτοκινουμένου οχήματος (λειτουργία 4), για τη χρήση του οχήματος πρέπει να τοποθετείται το όχημα στο δωμάτιο κοιτώντας τη πιο φωτεινή πλευρά του, η οποία καλό είναι να είναι όσο πιο σκοτεινή γίνεται. Με τα ir leds σβηστά, μόλις πάρει την ένδειξη του φωτισμού, θα φανεί στο serial monitor. Υστέρα θα περιμένει περίπου 6 sec, για να ανάψουμε τα ir leds και θα πάρει τη δεύτερη μέτρηση. Ύστερα εφόσον οι δεύτερες μετρήσεις είναι > από τις πρώτες, θα ξεκινήσει να κινείται προς τον στόχο και να αποφεύγει τα εμπόδια

7. Πηγαίος Κώδικας/ Source code

```
//some lines are commented out to disable wheels 1,2 in order to have enugh pins for distance measurement.
int Br_Preset=500;    //set the acceptable 'amount of' brightness in the room.(1024 is the max the sensor can read)
int Obstacle_Dist=20;
//Enable for wheels 1,2
//int En_1_2=2;
//Enable for wheels 3,4
int En_3_4=10;

//FIRST L293D
//int Wheel_1_1=5;//wheel 1_1
//int Wheel_1_2=3;//wheel 1_2

//int Wheel_2_1=6;//wheel 2_1
//int Wheel_2_2=9;//wheel 2_2

//SECOND L293D
int Wheel_3_1=4;//wheel 3_1
int Wheel_3_2=7;//wheel 3_2

int Wheel_4_1=8;//wheel 4_1
int Wheel_4_2=12;//wheel 4_2

//Distance sensors
const int TrigAll= 2;//i will use one pin to trigger them all together
const int EchoLeft=11;
const int EchoMid=5;
const int EchoRight=3;
//Distance variables
long duration;
int distance;

//Distance and last move (right/left) variables
char LastMov='n';
char Last_Ir=0;

int    Cur_Dist_M;
int    Cur_Dist_L;
int    Cur_Dist_R;

//declaring the names where the ir sensor readings are stored, and others for all the functions of ir

//Minimum ir sensor readings (without ir transmitters)
int RightSensMin;
int MidSensMin;
int LeftSensMin;
//============================================================
int i;
//============================================================
//variables that help the function of the function Av_Sens
int SensAdd=0, Sens=0;
//============================================================
//Error for hight brighness
bool BrightError=true;
bool Calibration=false;
//============================================================
//if ir receiver is >= than these, the target will be in sight
int LeftSensMax=0;
int RightSensMax=0;
int MidSensMax=0;
//============================================================
//Sweet spots
int MidSweetSpot,LeftSweetSpot, RightSweetSpot;

//These will change each time they take a different sample:
int MidReading, LeftReading, RightReading;

//function to stop all the wheels at once=============================================================================
void Break(){
//  digitalWrite(Wheel_1_1,0);
//  digitalWrite(Wheel_1_2,0);
//  digitalWrite(Wheel_2_1,0);
//  digitalWrite(Wheel_2_2,0);
  digitalWrite(Wheel_3_1,0);
  digitalWrite(Wheel_3_2,0);
  digitalWrite(Wheel_4_1,0);
  digitalWrite(Wheel_4_2,0);
//  digitalWrite(En_1_2,0);
  digitalWrite(En_3_4,0);
```

```
}

//Function to move the desired two wheels forward or
backward==========================================================================
void Move( int XWheel, int YWheel=0,bool forward=1){
  //first make sure aall wheels are stationary (due to hardware bug);
//  digitalWrite(En_1_2,0);
  digitalWrite(En_3_4,0);

 //enable the wheels we want to move
// if (XWheel==1 || YWheel==2 || XWheel==2 || YWheel==1){
//  digitalWrite(En_1_2,1);
// }

 if(XWheel==3 || XWheel==4 || YWheel==3 || YWheel==4){
    digitalWrite(En_3_4,1);
 }
  //if we pick wheel 1 in any case, make the forward move for wheel 1
   if((XWheel==1) || (YWheel==1)){
//digitalWrite(Wheel_1_1,forward);
//digitalWrite(Wheel_1_2,!forward);
  }
    //if we pick wheel 2 in any case, make the forward move for wheel 2
  if((XWheel==2) || (YWheel==2)){
//digitalWrite(Wheel_2_1,forward);
//digitalWrite(Wheel_2_2,!forward);
  }

  if((XWheel==3) ||(YWheel==3)){
digitalWrite(Wheel_3_1,forward);
digitalWrite(Wheel_3_2,!forward);
  }

    if((XWheel==4) ||(YWheel==4)){
digitalWrite(Wheel_4_1,!forward);
digitalWrite(Wheel_4_2,forward);
  }
}

//=============================================================================================
int Distance(char PinNum){

  digitalWrite(TrigAll,0);
  delayMicroseconds(3);
  //set the trigpin on high for 10 microseconds
  digitalWrite(TrigAll,1);
  delayMicroseconds(12);
  digitalWrite(TrigAll,0);

 if (PinNum=='l'){
duration=pulseIn(EchoLeft,1);
 }
 else if(PinNum=='m'){
duration=pulseIn(EchoMid,1);
 }
 else if(PinNum=='r'){
duration=pulseIn(EchoRight,1);
 }

  //distance calc
return duration*0.017;
  delay(50);
}

//====================Function to avoid obstacles=================================================
void Avoid_Obs(){
    Cur_Dist_M=Distance('m');
    Cur_Dist_L=Distance('l');
    Cur_Dist_R=Distance('r');
if(Cur_Dist_M<=Obstacle_Dist && Cur_Dist_L>Obstacle_Dist && Cur_Dist_R>Obstacle_Dist){
 if (Last_Ir=='l'){
       Serial.println("Moving left due to the last move left");
       do{
      Move(4,0,0);
      Move(3);
       Cur_Dist_M=Distance('m');
       }while(Cur_Dist_M<=Obstacle_Dist);
 }
 else if(Last_Ir=='r'){
        Serial.println("Moving right due to the last move right");
```

```
       do{
        Move(4);
        Move(3,0,0);
        Cur_Dist_M=Distance('m');
       }while(Cur_Dist_M<=Obstacle_Dist);
 }
 Break();
}
if(Cur_Dist_L<=Obstacle_Dist){
    Serial.println("left<=20");
  Break();
   Cur_Dist_L=Distance('l');
  while(Cur_Dist_L<=Obstacle_Dist){
      Move(4);
      Move(3,0,0);
      delay(500);
     Cur_Dist_L=Distance('l');
     Serial.println("left<=20");
  }
        LastMov='r';
  Move(3,4);
  Serial.println("Moving bit forward");
  delay(400);
  Break();
}

if(Cur_Dist_R<=Obstacle_Dist){
  Serial.println("right<=20");
    Break();
 Cur_Dist_R=Distance('r');
  while(Cur_Dist_R<=Obstacle_Dist){
      Move(4,0,0);
      Move(3);
      delay(500);
  Cur_Dist_R=Distance('r');
   Serial.println("right<=20");
  }
    LastMov='l';
    Move(3,4);
    Serial.println("Moving bit forward");
  delay(400);
  Break();
}
Serial.println("End of Avoid_obs()");
}

//============================Av_Sens=============================================================================
=================
//average measurement of a 'c' ir sensor
int Av_Sens(char c, int M=20){
  SensAdd=0;//setting SensAdd to 0 so next time the function is called, it wont use previews measures
  //calibrate for the selected ir sensor
for (i=0; i<=M; i++){ //M measurements for the c sensor
  if(c=='r'){
    Sens=analogRead(A5);
  }
  else if(c=='l'){
    Sens=analogRead(A3);
  }
  else if(c=='m'){
    Sens=analogRead(A4);
  }
delay(10);
SensAdd=SensAdd + Sens;
}
return SensAdd/M;//return the average of the readings
}


//============================Br_Level=============================================================================
====================
//Brightness level, gives BrightError=true or false
void Br_LevelCheck(){
  Serial.println("Dont stand in front of the sensors, turn off the ir transmitter and the lights");
 delay(10000);
  LeftSensMin=Av_Sens('l');
MidSensMin=Av_Sens('m');
RightSensMin=Av_Sens('r');

if(LeftSensMin>Br_Preset || RightSensMin>Br_Preset || MidSensMin>Br_Preset){
```

```
 Serial.println("Room is too bright");
  Serial.print("LeftSensMin= ");
 Serial.println(LeftSensMin);
 Serial.print("MidSensMin= ");
 Serial.println(MidSensMin);
 Serial.print("RightSensMin= ");
 Serial.println(RightSensMin);
 BrightError=true;
}
else {
   Serial.println("Brightness level passed");
  Serial.print("LeftSensMin= ");
 Serial.println(LeftSensMin);
 Serial.print("MidSensMin= ");
 Serial.println(MidSensMin);
 Serial.print("RightSensMin= ");
 Serial.println(RightSensMin);
  BrightError=false;
}
//calibrating
if (BrightError==false){
  Serial.println("Turn on ir transmitter and maintain maximum distance,facing directly the front ir receiver");
  delay(6000);
LeftSensMax=Av_Sens('l');
MidSensMax=Av_Sens('m');
RightSensMax=Av_Sens('r');

if(LeftSensMax>LeftSensMin && MidSensMax>MidSensMin && RightSensMax>RightSensMin){
 Serial.println("Calibrating done");
  Serial.print("LeftSensMax= ");
 Serial.println(LeftSensMax);
 Serial.print("MidSensMax= ");
 Serial.println(MidSensMax);
 Serial.print("RightSensMax= ");
 Serial.println(RightSensMax);
 Calibration=true;
}
else{
  Calibration=false;
  Serial.println("Did not detect any change on light or ir transmitter");
  Serial.print("LeftSensMax= ");
 Serial.println(LeftSensMax);
 Serial.print("MidSensMax= ");
 Serial.println(MidSensMax);
 Serial.print("RightSensMax= ");
 Serial.println(RightSensMax);
  }
}//DONE IF BRIGHTERROR==TRUE

if (Calibration==true){//this means all tests passed, time to set up the sweet spot
  //if the corresponding sensor value is > than the sweet spot, it is within sight
  LeftSweetSpot= (LeftSensMax+ LeftSensMin)/2 ;
   MidSweetSpot= (MidSensMax+ MidSensMin)/2 ;
  RightSweetSpot= (RightSensMax+ RightSensMin)/2 ;

  Serial.println("Sweet spots: ");
  Serial.print("LeftSweetSpot= ");
  Serial.println(LeftSweetSpot);
  Serial.print("MidSweetSpot= ");
  Serial.println(MidSweetSpot);
  Serial.print("RightSweetSpot= ");
  Serial.println(RightSweetSpot);
}
}

//=========================setup=============================================================================
========
void setup() {
  Serial.begin(9600);
pinMode(TrigAll,OUTPUT);
pinMode(EchoRight,INPUT);
pinMode(EchoLeft,INPUT);
pinMode(EchoMid,INPUT);

  //all the wheels
//pinMode( Wheel_1_1,OUTPUT);
//pinMode( Wheel_1_2,OUTPUT);
//pinMode( Wheel_2_1,OUTPUT);
//pinMode( Wheel_2_2,OUTPUT);
```

```
pinMode( Wheel_3_1,OUTPUT);
pinMode( Wheel_3_2,OUTPUT);
pinMode( Wheel_4_1,OUTPUT);
pinMode( Wheel_4_2,OUTPUT);

//pinMode(En_1_2, OUTPUT);
pinMode(En_3_4, OUTPUT);

//digitalWrite(En_1_2,0);
digitalWrite(En_3_4,0);
delay(100);

//check the brightness level start:
Br_LevelCheck();

}


//=======================loop=============================================================================================
=======
void loop() {

if (Calibration==true){

  do{
    MidReading=Av_Sens('m',1);
    LeftReading= Av_Sens('l',1);
    RightReading=Av_Sens('r',1);
    Serial.print("MidReading= ");
    Serial.println(MidReading);

    Serial.print("LeftReading= ");
    Serial.println(LeftReading);

    Serial.print("RightReading= ");
    Serial.println(RightReading);

  if (MidReading>MidSweetSpot){
    Move(3,4);
    Avoid_Obs();
    Serial.println("Moving forward/avoiding obstacles");
  }
  else if(RightReading>RightSweetSpot && RightReading>LeftReading){
    Last_Ir='r';
    Break();
    do{
      Serial.println("Moving right due to sweet spot");
      Move(3,0,0);
      Move(4);
      MidReading= Av_Sens('m',1);
      LeftReading= Av_Sens('l',1);
    }while(MidReading<=MidSweetSpot || LeftReading<=LeftSweetSpot);
  }
  else if(LeftReading> LeftSweetSpot && LeftReading>=LeftSweetSpot){
    Last_Ir='l';
    Break();
    do{
    Serial.println("Moving left due to sweet spot");
      Move(3);
      Move(4,0,0);
      MidReading= Av_Sens('m',1);
      RightReading=Av_Sens('r',1);
    }while(MidReading<=MidSweetSpot || RightReading<=RightSweetSpot);
  }
}  while(1);//END OF DO WHILE(1)

}//END OF IF CALIBRATION==TRUE

}//END OF LOOP
```

8. Πηγές και Βιβλιογραφία

- https://www.arduino.cc
- https://www.opateipir.gr
- https://auto.teipir.gr
- https://www.thingiverse.com/