Mașinuță cu ghidaj inteligent

Autor: Dima Elena-Madalina

Grupa: 332CA

Introducere

* Proiectul reprezintă o mașină ce are 2 moduri de funcționare:

- *Control automat* - în acest mod, mașinuța identifică obstacolele și le ocolește, mai exact la întâlnirea unui obstacol, robotul se deplasează un pic în spate si alege o altă direcție de deplasare; de asemenea mașina are grijă și să nu cadă de pe suprafața pe care se alfă, oprindu-se în momentul în care detectează că se află pe margine.

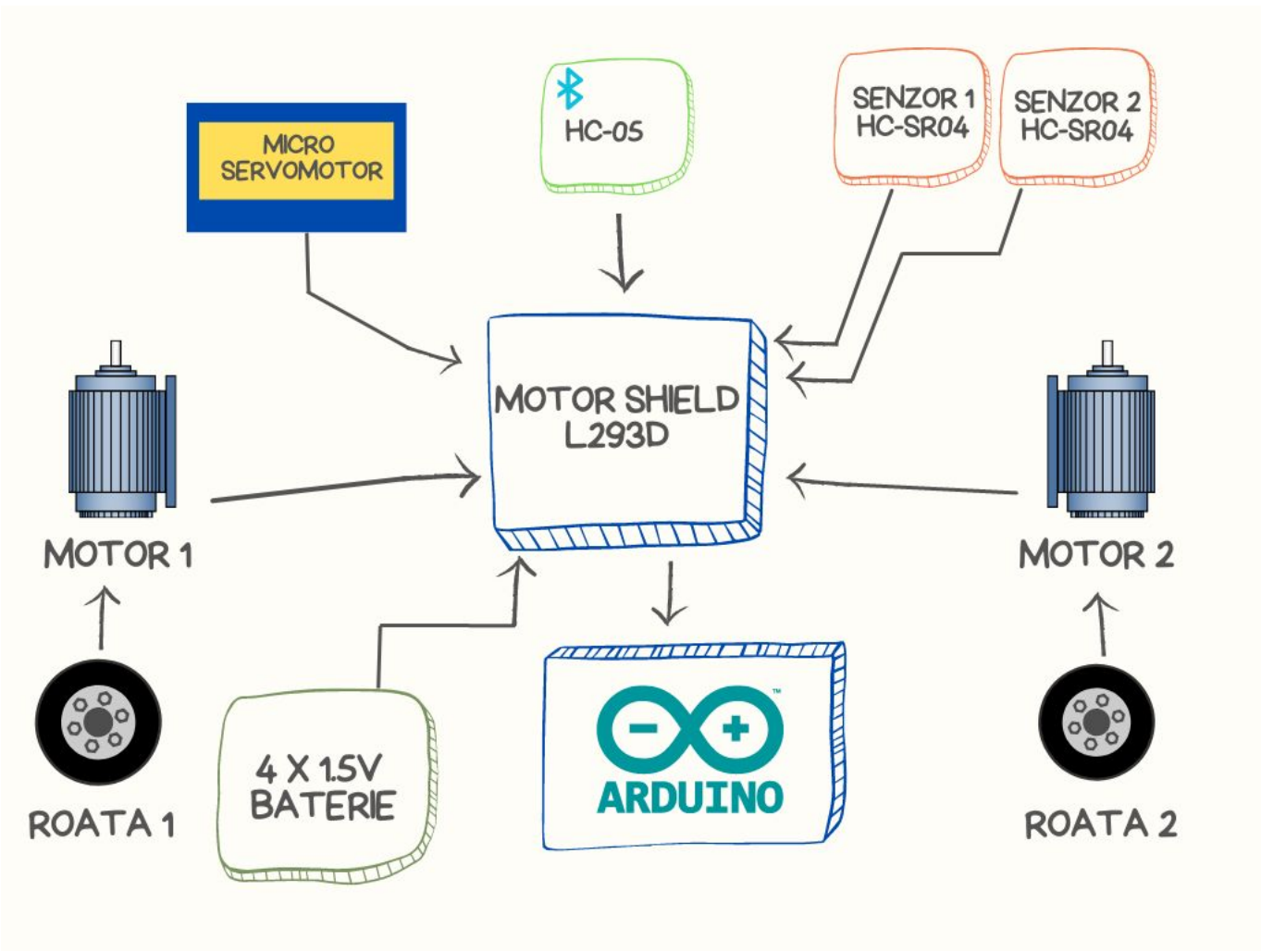
- *Control manual* - în acest mod, robotul poate fi controlat de către un șofer, folosind un telefon mobil pe post de telecomană

* **Scopul proiectului** este unul recreativ, funcționând ca o jucărie, dar poate reprezenta un prototiv pentru dezvoltarea unor proiecte mai complexe cum ar fi aspiratoare-robot, dispozitive de colectarea a datelor din spații greu accesibile, sisteme auto de detecție a diverselor pericole ce pot apărea

* Ideea inițială cuprindea un simplu robotel ce ar fi fost controlat folosind telefonul și care ar fi avut atașat în partea frontală o lavetă pentru a șterge praful de pe diverse suprafețe, în general, rafturi.

* Din punct de vedere personal, **utilitatea proiectului** este reprezentată de aprofundarea și consolidarea cunoștiințelor de pm și nu numai dobândite, alături de construirea unui dispozitiv ce poate fi ulterior adaptat pentru diverse alte activități.

Descriere generală



Pentru un control mai facil al motoarelor și al caracteristicilor acestora s-a ales folosirea motor-shield-ului L293D care se atașează plăcuței Arduino. Astfel, conexiunile la porturile plăcuței se vor face tot prin intermediul motor shield-ului.

Motorul din dreapta robotelului a fost conectat la porturile M2, iar motorul stâng la porturile M3.

Pentru identificarea obstacolelor și verificarea poziției pe suprafața pe care se deplasează se folosesc senzori de distanță ultrasonici HC-SR04. Pentru depistarea obstacolelor, un senzor este plasat în partea superioară a robotelului, atașat de un micro servomotor care permite analiza celorlalte direcții de deplasare atunci când se identifică un obstacol în față. Pentru a verifica faptul că există o suprafață sub roți, unul dintre senzorii de distanță va fi orientat spre podea. În cazul ambilor senzori porturile Gnd și vcc sunt conectate la aceleași porturi ale motor shield-ului, în timp ce echo și trig au fost conectați la pinii analogici ai plăcuței.

Pentru recepționarea comenzilor trimise manual se folosește modulul Bluetooth HC-05 care are pinul RXD conectat la TXD-ul Arduino-ului, iar pinul TXD conecta la RDX.

Pentru o identificare mai ușoară a firelor și a utilității lor s-a adoptat următorul cod de culoare:

- fir verde → GND
- fir alb → VCC

Alimentarea se face folosind 4 baterii de 1,5V, plasate într-un suport de baterii conectat la motor-shield la care se adaugă o baterie de 9V pentru alimentarea plăcuței Arduino.

Hardware Design

Listă piese:

- 1 X Arduino Uno
- 1 X Motor Shield L293D
- 1 X Suport baterii
- 1 X Suport baterii
- 1 X Micro servomotor SG90
- 3 X Senzori ultrasonici HC-SR04
- 2 X Roti
- 2 X Motoare
- 1 X Modul bluetooth HC-05
- Fire

Schemă electrică

Software Design

===== Platforme utilizate=====

- mediu de dezvoltare: Arduino IDE
- librării şi surse 3rd-party:
  - Adafruit Motor Shield pentru controlul motoarelor,
  - NewPing.h pentru controlul senzorilor de distanță
  - Servo.h pentru control servomotorului

Descrierea codului

Includerea bibliotecilor necesare

```
#include <Servo.h>
#include <Servo.h>
#include <NewPing.h>
#include <AFMotor.h>
```

Pentru a spori lizibilitatea codului și a lucra mai ușor cu pinii și constantele folosite s-au definit următoarele macro-uri:

```
#define TRIG_PIN_OBS A0
#define ECHO_PIN_OBS A3

#define TRIG_PIN_S A2
#define ECHO_PIN_S A5
#define MAX_DIST 400
#define MAX_SPEED 255
```

Senzorii, motoarele și servomotorul au fost declarate global folosindu-se funcțiile specifice de bibliotecă. Prin sonarObs se face referire la senzorul de distanță plasat deasupra robotului care măsoară distanța față de obiectele din față, fiind folosit pentru ocolirea obstacolelor. SonarS desemnează al doilea senzor de distanță utilizat, cel cu care se verifică dacă mașinuța încă se află pe o suprafață.

```
NewPing sonarObs(TRIG_PIN_OBS, ECHO_PIN_OBS, MAX_DIST);
NewPing sonarS(TRIG_PIN_S, ECHO_PIN_S, MAX_DIST);

AF_DCMotor motorRight(2,  MOTOR12_8KHZ);
AF_DCMotor motorLeft(3,  MOTOR12_8KHZ);
Servo servo;
```

Mesajele primite de către roboțel sunt de tip caracter, fiecare literă declanșând o anumită acțiune. Lista comenzilor disponibile este prezentată mai jos. Astfel, variabila *dataIn* reține comanda primită și inițial are valoare 'S', asociată cu oprirea motoarelor. Una dintre funcționalitățile oferite este setarea vitezei de deplasare, aceasta fiind reținută în variabila *speed*, care inițial are valoarea maximă. Variabila *dist* reprezintă distanța până la cel mai apropiat obiect din față, în timp ce *distS* reprezintă distanță până la suprafață pe care se deplasează mașina.

```
char dataIn = 'S';
int speed = 255;
int dist = 100;
int distS = 0;
```

Pentru fiecare tip de deplasare al mașinuței s-a implementat câte o funcție după cum urmează:

```
// stop
void turnOffMotors() {
  motorRight.run(RELEASE);
  motorLeft.run(RELEASE);
}

// deplasare în față
void moveForward() {
    motorRight.setSpeed(speed);
    motorLeft.setSpeed(speed);
    motorRight.run(FORWARD);
    motorLeft.run(FORWARD);
}

// deplasare in spate
```

```
void moveBackward() {
    motorRight.setSpeed(speed);
    motorLeft.setSpeed(speed);
    motorRight.run(BACKWARD);
    motorLeft.run(BACKWARD);
}

// întoarcere la stânga
void turnLeft() {
  motorRight.setSpeed(speed);
  motorLeft.setSpeed(speed);
  motorRight.run(FORWARD);
  motorLeft.run(BACKWARD);
}
// întoarcere la dreapta
void turnRight() {
  motorRight.setSpeed(speed);
  motorLeft.setSpeed(speed);
  motorRight.run(BACKWARD);
  motorLeft.run(FORWARD);
}

// deplasare la stanga
void moveLeft() {
  motorRight.setSpeed(speed);
  motorLeft.setSpeed(speed);
  motorRight.run(FORWARD);
  motorLeft.run(BACKWARD);
  delay(500);
  motorRight.setSpeed(speed);
  motorLeft.setSpeed(speed);
  motorRight.run(FORWARD);
  motorLeft.run(FORWARD);
}

// deplasare la dreapta
void moveRight() {
  motorRight.setSpeed(speed);
  motorLeft.setSpeed(speed);
  motorRight.run(BACKWARD);
  motorLeft.run(FORWARD);
  delay(500);
  motorRight.setSpeed(speed);
  motorLeft.setSpeed(speed);
  motorRight.run(FORWARD);
  motorLeft.run(FORWARD);
}

// deplasare în direcția Nord - Est
void moveRightForward() {
  motorRight.setSpeed(speed / 2);
```

```
  motorLeft.setSpeed(speed);
  motorRight.run(FORWARD);
  motorLeft.run(FORWARD);
}

// deplasare în direcția Sud - Est
void moveRightBackward() {
  motorRight.setSpeed(speed / 2);
  motorLeft.setSpeed(speed);
  motorRight.run(BACKWARD);
  motorLeft.run(BACKWARD);
}

// deplasare în direcția Nord - Vest
void moveLeftForward() {
  motorRight.setSpeed(speed);
  motorLeft.setSpeed(speed / 2);
  motorRight.run(FORWARD);
  motorLeft.run(FORWARD);
}

// deplasare în direcția Sud - Vest
void moveLeftBackward() {
  motorRight.setSpeed(speed);
  motorLeft.setSpeed(speed / 2);
  motorRight.run(BACKWARD);
  motorLeft.run(BACKWARD);
}
```

Pentru virajele la stânga, respectiv la dreapta, roata dinspre viraj se mișcă în spate, iar cealaltă roată se mișcă în față, în timp ce deplasările pe diagonale sunt realizate prin setarea de viteze diferite pentru cele două motoare.

Inițializarea servomotorului cuprinde 2 etape: legarea la pinul 10 prin funcția de bibliotecă *attach* și setarea unghiului la 90º, unghi la care senzorul de distanță are orientarea corespunzătoare.

```
void initServo() {
  servo.attach(10);
  servo.write(90);
}
```

Inițializările a fost incluse în funcția *setup*:

```
void setup() {
  Serial.begin(9600);
  initServo();
  turnOffMotors();
}
```

Logica principală

Pentru implementarea modului manual de control al robotului, comenzile primite prin modulul Bluetooth sunt analizate și se apelează funcția corespunzătoare comenzii primite.

Literele care definesc comenzile au fost alese astfel încât să corespundă celor folosite în aplicația mobilă descrisă în link pentru a se putea realiza astfel și controlul mașinii folosind telefonul drept telecomandă. Aplicația permite și controlul robotelului prin mișcarea telefonului, folosindu-se de accelerometru

Aplicație control mașină

```
void loop() {
  // put your main code here, to run repeatedly:
  char command = getBluetoothInput();
  Serial.println(command);

  switch (command) {
    case 'F':
      moveForward();
      command = getBluetoothInput();
      break;

    case 'B':;
      moveBackward();
      command = getBluetoothInput();
      break;

    case 'L':
      turnLeft();
      command = getBluetoothInput();
      break;

    case 'R':
      turnRight();
      command = getBluetoothInput();
      break;

    case 'I':
      moveRightForward();
      command = getBluetoothInput();
      break;

    case 'J':
      moveRightBackward;
      command = getBluetoothInput();
      break;


    case 'G':
      moveLeftForward();
```

```
      command = getBluetoothInput();
      break;

    case 'H':
      moveLeftBackward();
      command = getBluetoothInput();
      break;

    case 'S':
      turnOffMotors();
      command = getBluetoothInput();
      break;

    case 'W':
      servo.write(90);
      command = getBluetoothInput();
      break;

    case '1':
      servo.write(45);
      command = getBluetoothInput();
      break;

    case '2':
      servo.write(0);
      command = getBluetoothInput();
      break;

    case '3':
      servo.write(135);
      command = getBluetoothInput();
      break;

    case '4':
      servo.write(180);
      command = getBluetoothInput();
      break;

    case '5':
      speed = 150;
      command = getBluetoothInput();
      break;

      case '6':
      speed = 175;
      command = getBluetoothInput();
      break;

      case '7':
      speed = 200;
```

```
      command = getBluetoothInput();
      break;

      case '8':
      speed = 225;
      command = getBluetoothInput();
      break;

      case '9':
      speed = 250;
      command = getBluetoothInput();
      break;

      case 'V':
        avoid();
        command = getBluetoothInput();
        break;
  }
}
```

Comenzi disponibile

- *F* → deplasare înainte
- *B* → deplasare înapoi
- *L* → viraj stânga
- *R* → viraj dreapta
- *I* → deplasare Nord - Est
- *J* → deplasare Sud - Est
- *G* → deplasare Nord - Vest
- *H* → deplasare Sud - Vest
- *S* → oprire
- *W* → aducerea servo-ului în poziția inițială (90º)
- *1* → aducerea servo-ului la 45º (deplasare senzor spre dreapta)
- *2* → aducerea servo-ului la 0º (deplasare senzor spre dreapta)
- *3* → aducerea servo-ului la 135º (deplasare senzor spre stânga)
- *4* → aducerea servo-ului la 180º (deplasare senzor spre stânga)
- *5* → setarea vitezei la 150
- *6* → setarea vitezei la 175
- *7* → setarea vitezei la 200
- *8* → setarea vitezei la 225
- *9* → setarea vitezei la 250
- *V* → declanșarea modului automat

Implementarea modului automat

Prima dată se efectuează verificarea faptului că robotul încă se află pe o suprafață prin citirea distanței dintre senzorul de distanță S și primul obiect găsit dedesut. O distanță mai mare de 7cm semnalizează faptul că în față suprafață se termină, așa că robotul se oprește, se deplasează înapoi

pentru a evita căderea la ocolirea marginii și se întoarce spre stânga.

După ce s-a validat faptul că deplasarea în față nu poate cauza căderea de pe masă, se verifică dacă există obstacole în apropiere. Pentru aceasta se citește distanța până la primul obiect din față și când această distanță este mai mică de 25 cm se consideră că obiectul identificat este o amininţare și trebuie ocolit. Direcția în care se continuă deplasarea este aleasă pe baza distanței maxime. Mai precis, senzorul de distanță Obs se direcționează spre dreapta prin aducerea servomotorului la 0º, determină distanța până la cel mai apropiat obstacol și se aduce servomotorul în starea inițială(la 90º). Asemănător se determină și distanță din stânga, cu diferență că, de această dată servo-ul este adus la 180º. Calcularea acestor distațe este realizată prin funcțiile:

```
int getRightDist() {
  servo.write(0);
  delay(500);
  delay(70);
  int dist = sonarObs.ping_cm();
  delay(100);
  servo.write(90);
  return dist;
}

int getLeftDist() {
  servo.write(180);
  delay(500);
  delay(70);
  int dist = sonarObs.ping_cm();
  delay(100);
  servo.write(90);
  return dist;
}
```

Astfel, deplasarea se continuă în direcția obstacolului cel mai depărtat. Dacă nu s-a identificat un obstacol în apropiere și nici nu există pericolul de cădere, mașinuța se deplasează înainte.

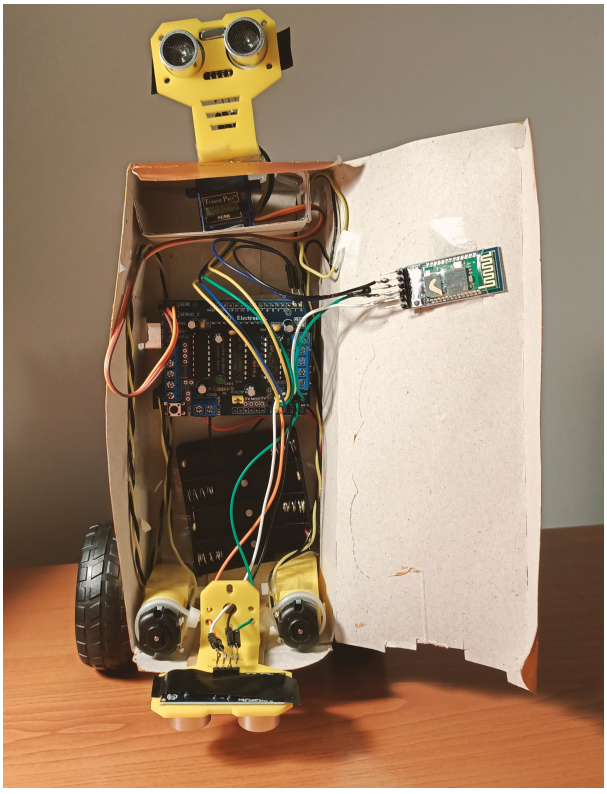
Logica completă a acestui mod de funcționare este implementată astfel:

```
void avoid() {
  distS = sonarS.ping_cm();
      delay(40);
      if (distS >= 7) {
        turnOffMotors();
        delay(300);
        moveBackward();
        delay(50);
        turnLeft();
        delay(100);
      } else {
        int rDist = 0;
        int lDist = 0;
        dist = sonarObs.ping_cm();
```

```
        delay(40);
        if (dist <= 25) {
          // obstacle is too close and needs to be avoided
          Serial.println("obstacle is too close and needs to be avoided");

          turnOffMotors();
          delay(100);

          moveBackward();
          delay(300);

          turnOffMotors();
          delay(200);

          rDist = getRightDist();
          delay(200);
          lDist = getLeftDist();
          delay(200);

          if (rDist >= lDist) {
            moveRight();
            turnOffMotors();
          } else {
            moveLeft();
            turnOffMotors();
          }
        } else {
          Serial.println("No obstacles on my way");
          delay(15);
          moveForward();
        }
      }
      dist = sonarObs.ping_cm();
}
```

Rezultate Obţinute

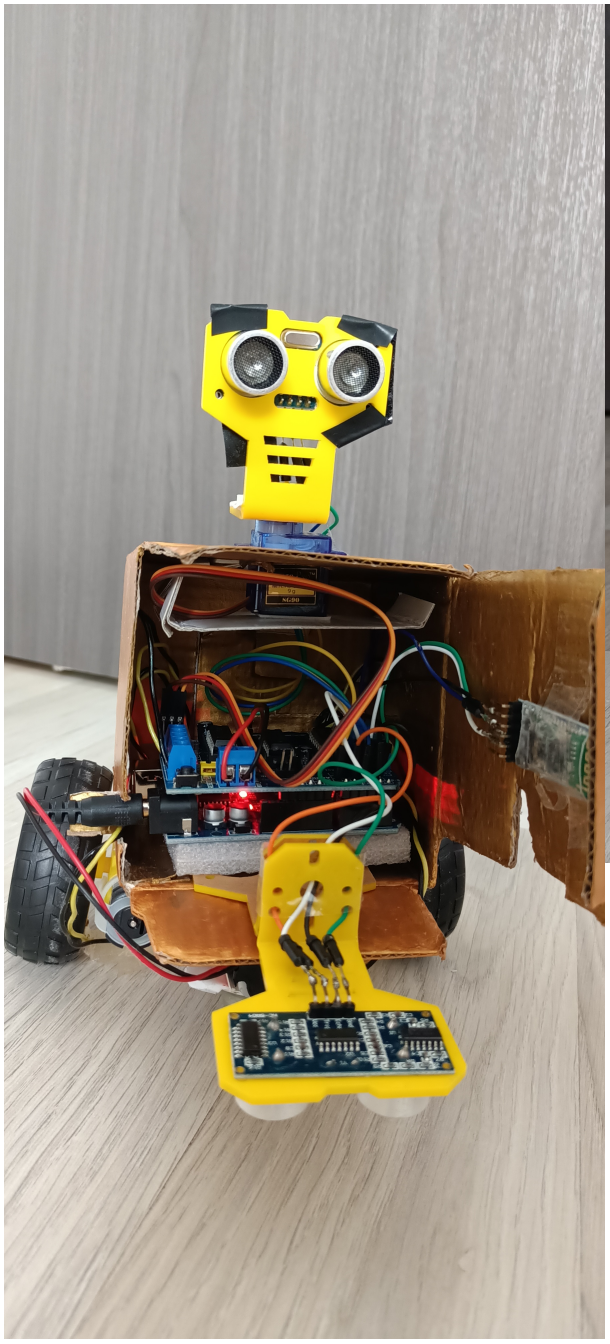
Design inițial:

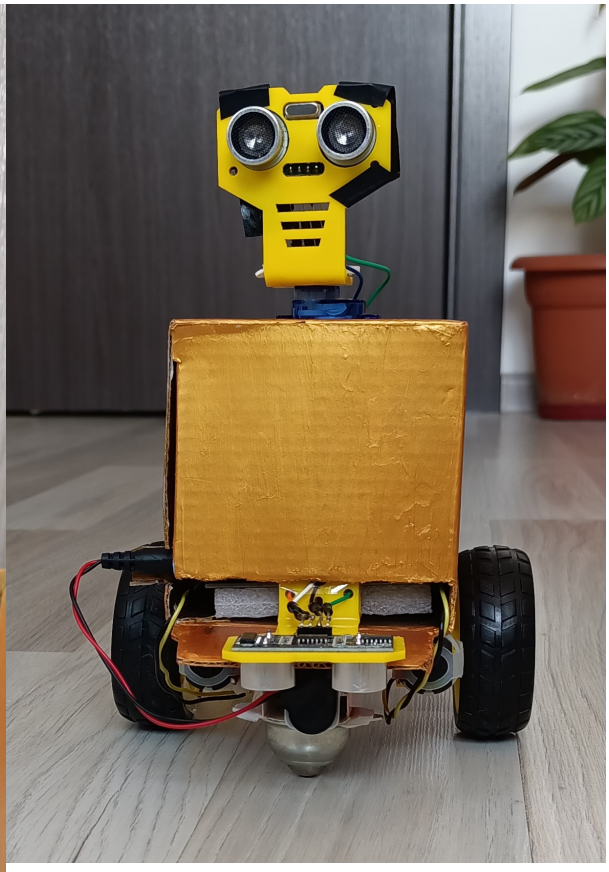


Inițial s-a optat pentru o carcasă încăpătoare pentru a nu înghesui componentele și a se încerca o organizare a cablurilor cât mai neîncâlcită, dar, în urma testării s-a dovedit că acest design are probleme cu echilibrul, la plecarea de pe loc, datorită inerției, robotul căzând pe spate. De asemenea, această carcasă, fiind dintr-un carton destul de subțire nu oferea stabilitate și motoarele nu puteau fi fixate corespunzător, aspect ce afecta direcția de deplasare.

De aceea s-a optat pentru schimbarea carcasei cu una mai scundă și mai lată ce oferă o stabiltate mai bună, construită dintr-un carton mai tare ce permite fixarea componentelor. Deși spațiul este mai mic, nu s-a renunțat la păstrarea firelor neîncâlcite, încercându-se o organizarea cât mai ordonată a componentelor.

Design final





Testare

Primul test - control manual

Test 2 - cotrol automat - evitare cădere

Robotelul evită căderea doar când marginea suprafeței este înaintea lui, pentru acoperirea celorlalte cazuri fiind nevoie de senzori de distanță pe fiecare latură care să funcționeze similar cu cel folosit în acest proiect.

Test 3 - control automate evitare obstacole mai scunde ca senzorul

Acest test a scos la iveală faptul că robotelul ignoră obstacolele care nu intră în aria lui vizuală. O versiune mai avansată ar putea folosi fie mai mulți senzori, plasați la diferite înălțimi, fie un singur senzor care să își poată modifica unghiul sub care privește.

Test 4 - control automat - evitare obstacole

Test 5 - control automate unghi ascuțit

În cadrul acestui test se poate observa faptul că șansele ca un obstacol să fie identificat scad atunci când unghiul dintre acesta și senzorul de distanță este ascuțit. Această problemă ar putea fi rezolvată fie prin adăugarea mai multor senzori care să monitorizeze simultan mai multe direcții, fie prin mai multe verificări efectuate cu un singur senzor prin poziționarea sa în mai multe unghiuri înainte de a se lua decizia dacă există sau nu un obstacol.

Concluzii

1. Ca în cadrul oricărei lucrări, proiectarea este una dintre etapele cele mai importante, în cazul în care este făcută cu atenție, având ca rezultat o desfășurare cu mai puține erori a proiectului, alături de evitarea unor costuri suplientare atât materiale, cât și temporale.

2. Robotelul este departe de o formă finală, putând fi îmbunătățit în foarte multe feluri (adăugarea mai multor senzori pentru identificarea corectă a obstacolelor și evitarea tuturor cazurilor în care poate cădea de pe suprafață, adăugarea mai multor elemente de design, leduri, buzzer pentru redarea unor sunete diferite în funcție de modul de funcționare, semnalizarea blocării)

Download

Codul folosit Arhivă cod

Jurnal

- 26.04: - Alegere proiect
- 28.04: - 30.04 - Achiziționare componente
- 02.05: - Lipit fire motoare
- 05.05: - Schemă electrică
- 06.05: - Documentație inițială
- 12.05: - 14. 05 - Lipit fire componente + asamblare
- 20.05: - Completări documenație: detaliere descriere generală

- 21.05:
  1. Testat deplasare robot
  2. Schimbat șasiu și carcasă din motive de echilibru
- 22.05 - Testare senzor de distanță - erori
- 23.05 - Depanare erori senzor distanță
- 27.05 - Scriere cod pentru verificare suprafață și evitare obstacole
- 28.05
  1. Verificare funcționalitate robot
  2. Definitivare documentație

Bibliografie/Resurse

Resurse hardware:

https://lastminuteengineers.com/l293d-motor-driver-shield-arduino-tutorial/
https://howtomechatronics.com/tutorials/arduino/arduino-and-hc-05-bluetooth-module-tutorial/
https://howtomechatronics.com/tutorials/arduino/ultrasonic-sensor-hc-sr04/
https://playground.arduino.cc/Code/NewPing/#Example
https://components101.com/sites/default/files/component_datasheet/HC-05%20Datasheet.pdf
https://5.imimg.com/data5/PX/UK/MY-1833510/l293d-based-arduino-motor-shield.pdf
https://datasheetspdf.com/pdf-file/1351717/Microchip/ATmega328P/1

Tutoriale:

https://www.instructables.com/Make-Your-First-Arduino-Robot-the-Best-Tutorial/
https://www.instructables.com/How-to-Make-Smart-Obstacle-Avoiding-Robot-Using-Ar/
https://www.viralsciencecreativity.com/post/arduino-simple-obstacle-avoiding-robo
https://srituhobby.com/how-to-make-an-obstacle-avoiding-robot-with-three-ultrasonic-sensors/

Export to PDF