Treasure Hunt Robot

Student: Lincă Nicoale-Robert
Grupa: 331 CC

Introducere

Proiectul reprezintă un robot tip mașină, autonom care va identifica si semnaliza automat obiecte de metal aflate pe un traseu predefinit. Pe acest traseu se pot afla obstacole care o vor fi ocolite in mod automat, prin stanga sau prin dreapta, in functie de spatiul disponibil, iar la intalnirea unui zid, care se va afla la final de trraseu, robotul il va identifica si isi va opri executia.

Acest proiect este util pentru că este un prim pas spre învățarea dezolvatării de roboți autonomi care repezintă un interes de actualitate în industrie, spre exemplu un robot autonom care realizează cartografierea unei zone reprezintă un astfel de interes, care prezintă o oarecare similiaritate cu proiectul ales, fiind astfel un bun model de învățare.

Descriere generală

Schema bloc:

Modulul arduino va utiliza sevomotorul pentru a roti senzorul ultrasonic de proximitate pentru a depista obstacolele pentru a asigura menținerea traseului de parcurgere, respectiv a ocoli obstacolele și a reveniri pe traseu. Toți acești factori vor determina comenzile PWM pe care le va transmite driver-ului de motor care controlează motoarele roților ceea ce permite deplasarea corectă și autonomă a mașinii.

Modulul arduino va citi senzorii inductivi pentru a depista găsirea unui obiect de metal, pe care o va notifica prin modulul bluetooth la un monitor serial de pe calculator. Modulul bluetooth va avea rol și ca tool de debugging în dezvoltarea proiectului.

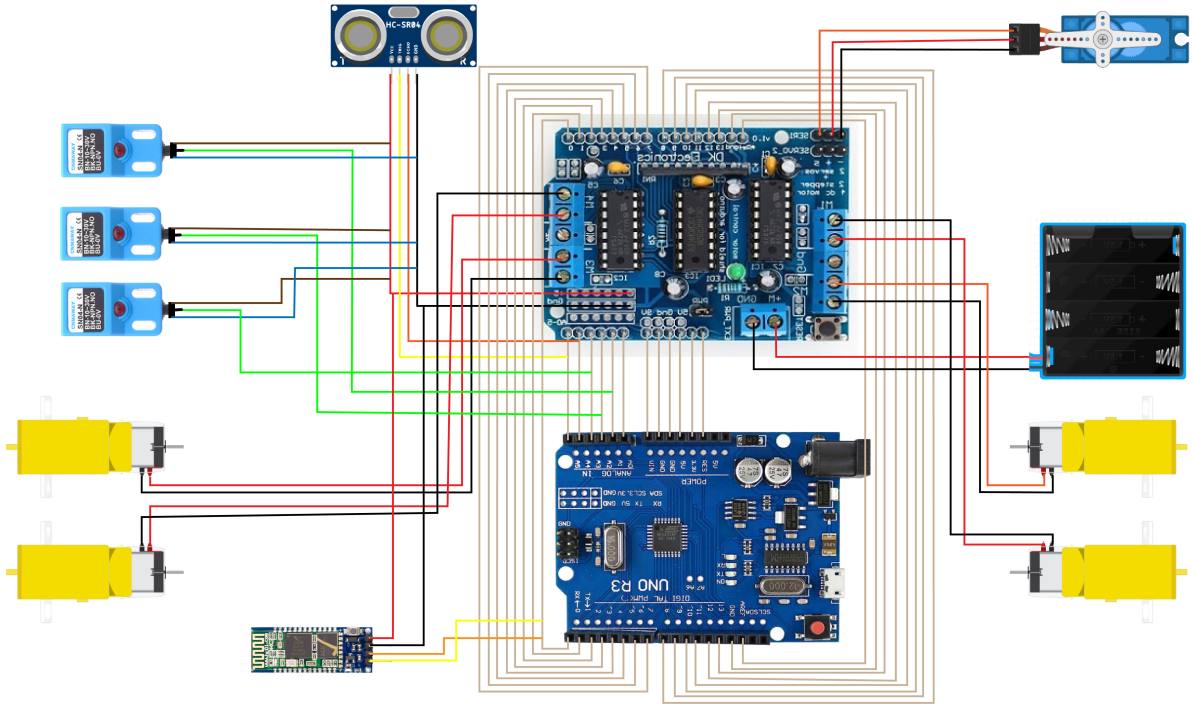
Hardware Design

Listă componete folosite:

- Arduino Uno R3 - ATmega328P

- Modul driver motoare L293D
- Modul bluetooth HC-05
- Motor 5v reductor
- MicroServo motor 9G
- Senzor ultrasonic HC-SR04
- Senzor inductiv SN04-N
- Rezistențe
- Suport baterii
- Șasiu mașină
- Roți

Schemă robot:



Software Design

Codul pentru robot a fost realizat în Arduino 1.8.16, iar pentru controlul motoarelor si a servomotorului am folosit bibliotecile <*AFMotor.h*>, respectiv <*Servo.h*>. In ceea ce priveste funcțiile implementate, acestea pot fi împărțite în următoarele categorii:

A. **Funcții pentru controlul direcției**:

1. *void forward()* → mișcare înainte pe o perioadă nedefinită
2. *void forward2(int timp)* → mișcare înainte timp de *timp* ms
3. *void backward()* → mișcare înapoi pe o perioadă nedefinită
4. *void turnleft()* → rotire pe loc spre stânga pe o perioadă nedefinită
5. *void turnleft2(int timp)* → rotire pe loc spre stânga timp de *timp* ms
6. *void turnright()* → rotire pe loc spre dreapta pe o perioadă nedefinită
7. *void turnright2(int timp)* → rotire pe loc spre dreapta timp de *timp* ms
8. *void Stop()* → oprirea motoarelor

B. **Funcții pentru detecția obstacolelor**:

1. *int leftsee()* → returnează distanța până la cel mai apropiat obiect din stânga-fata în cm
2. *int rightsee()* → returnează distanța până la cel mai apropiat obiect din dreapta-fata în cm
3. *int ultrasonic()* → returnează distanța până la cel mai apropiat obiect detectat de senzorul ultrasonic (pentru mai multe detalii de implementare: https://stackoverflow.com/questions/41501360/getting-distance-in-inches-and-cm-from-ultrasonic-sensor-in-arduino)

C. **Funcții pentru ocolirea obstacolelor**:

1. *void ocolire_stanga()* → rutină de ocolire a obstacolului prin partea stângă print-o parcurgere în formă de trapez
2. *void ocolire_stanga()* → rutină de ocolire a obstacolului prin partea dreaptă print-o parcurgere în formă de trapez

Descriere Algoritm de Funcționare

După ce robotul a trecut de setup-ul senzorilor și a motoarelor, el va face busy-waiting până la primirea caracterului '1' pe interfața serială, prin bluetooth, moment în care va porni algoritmul de parcurgere automată din funcția *loop()*. Algoritmul de funcționare presupune în primul rând citirea în permanență a distanței până la cel mai apropiat obiect de pe traseu, astfel că în momentul apropierii de un obiect, robotul se va opri și va realiza una din rutinele de ocolire, sau va decide dacă ceea ce se află în fața sa este un zid, de la finalul traseului, caz în care își va încheia execuția. Un alt eveniment care are prioritate asupra modului normal de urmărire a traseului sau de evitare a obstacolelor o reprezintă detecția unui obiect de metal, caz în care robotul se va opri și va notifica pe moitorul serial de pe calculator găsirea acestuia, după care își va relua execuția normală a algoritmului de parcurgere a traseului.

Algoitmul de parcurgere a traseului constă in deplasarea în linie dreaptă până ce senzorul ultrasonic depisteză un obstacol la mai puțin de 35cm, moment în care este dată comanda de oprire a motoarelor. După care senzorul va fi poziționat să depisteze distanța la care se află cel mai apropiat obstacol în stânga, respectiv în dreapta obiectului din față, iar dacă aceste distanțe sunt comparabile, diferă cu cel mult 10cm, înseamnă că obiectul din față este un zid, caz în care se încheie execuția programului, altfel robotul va decide să îl ocolească prin stânga/dreapta în funcție de spațiul cel mai mare disponibil (disanța până la cel mai aporpiat obiect din stânga/dreapta cât mai mare), apelând una din cele 2 rutine de ocilire ce realizează un traseu în formă de trapez.

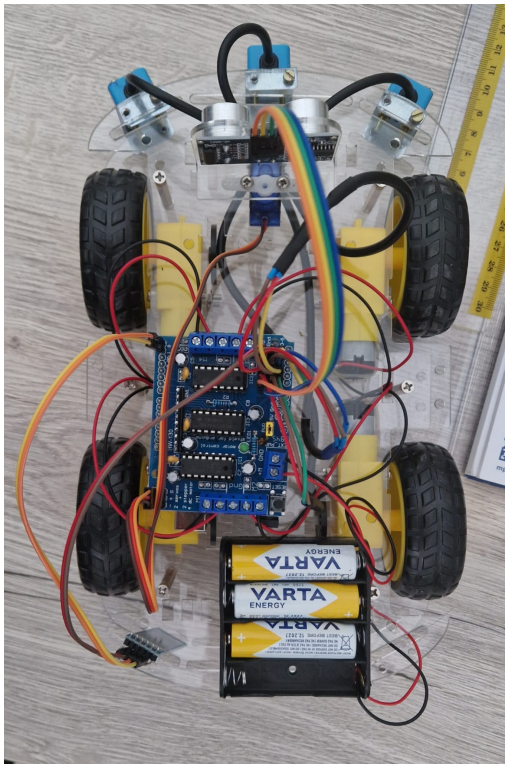
```
void loop() {
  distance = ultrasonic();
  // read metal Sensors
  int val[3] = {0, 0, 0};
  val[0] = analogRead(metalSensor[0]);
  val[1] = analogRead(metalSensor[1]);
  val[2] = analogRead(metalSensor[2]);
  if (val[0] <= metalThreshold || val[1] <= metalThreshold || val[2] <=
metalThreshold) {
    Stop();
    Serial.println("Gasit metal!");
```

```
    forward2(400);
    val[0] = val[1] = val[2] = METAL_MAX;
    delay(wait_metal);
  }

  if (distance < 35) {
    Stop();
    L = leftsee();
    delay(800);
    R = rightsee();
    Serial.println("Obiect detectat!");
    servo.write(spoint);
    if (abs(distance - R) <= wallDistErr && abs(distance - L) <= wallDistErr)
{
      Stop();
      servo.write(spoint);
      Serial.println("Zid detectat!\nProgram ended.");
      delay(500);
      exit(0);
    } else {
      if (L > R) {
        backward();
        delay(300);
        Stop();   delay(wait_time);
        Serial.println("Ocolire stanga.");
        ocolire_stanga();
      } else {
        backward();
        delay(500);
        Stop();   delay(wait_time);
        Serial.println("Ocolire dreapta.");
        ocolire_dreata();
      }
    }
    servo.write(spoint);
  } else {
    forward();
  }
}
```

Rezultate Obţinute

Robot deadline hardware:



Demo funționare: https://www.youtube.com/shorts/Zkjj-E443SU

Concluzii

În implementarea robotului am realizat cât de importanți pot fi factori externi pentru realizarea roboților autonomi,precum și cât de importantă este calitatea componentelor din care este realizat. Acest lucru l-am aflat “the hard way” în timp ce lucram la programul software al robotului unde din cauza problemelor hardware procesul de dezvoltare a fost îngreunat.

Principala problemă de care m-am lovit în momentul în care am asamblat toate piesele și a umrat faza de testare în ansamlbu a fost faptul că motoarele consumă foarte mult, și majoritatea pieselor erau subalimentate, ceea ce m-a făcut să încerc diverse alimentări de la suportul de baterii de 6V, la o baterie de 9V, până la varianta finală în care am decis să folosesc o sursă de tensiune de 12V, modificată de la o sursă de calculator. Însă și după rezolvarea acestei probleme majore tot au mai rămase multe altele precum: aderența slabă a roților, driver-ul de motoare nu distribuie uniform puterea la cele 4 motoare, senzorii inductivi după o perioadă îndelungată de funționare devin mult mai sensibili și dau răspunsuri eronate. Aceste probleme am încercat să le rezolv e cât de mult posibil din software, însă și acesta are limitările sale, fapt pentru care robotul de multe ori poate avea un răspuns eronat.

Aceste lucruri m-au făcut să realizaez cât de importantă este o bună planificare și documentare în alegerea pieselor în faza de incipit a realizării unui astfel de proiect, deoarece o dată apărute astfel de probleme, se poate ajunge la schimbarea radicală a planului de proiectare atât la nivel software, cât și hardware, sau chiar regândirea în întregime a proiectului.

Download

Arhiva cu codul sursa, diagrama robot, schema bloc: pm2023_linca_nicolae_robert.zip

Jurnal

- **30.05.2023** - Finalizare documentație
- **29.05.2023** - Finalizare Software
- **28.05.2023** - Schimbare mod de alimentare robot
- **27.05.2023** - Primă formă Software și schițarea algoritmului robotului
- **20.05.2023** - Finalizare hardware
- **07.05.2023** - Documentația inițială (Introducere, descriere, schema bloc, listă componente)
- **03.05.2023** - Creare pagină wiki
- **30.04.2023** - Comandă piese proiect
- **28.04.2023** - Alegere temă proiect

Bibliografie/Resurse

- https://ocw.cs.pub.ro/courses/pm
- https://www.alldatasheet.com/view.jsp?Searchword=L293d%20Datasheet&gad=1&gclid=CjwKCAjwvdajBhBEEiwAeMh1U1EUUKuStKriJemzf1RBs53vMS1yCWJqSxatoaDHArx2Lmke82saxxoCbI4QAvD_BwE
- https://components101.com/sites/default/files/component_datasheet/HC-05%20Datasheet.pdf
- https://www.the-diy-life.com/arduino-based-obstacle-avoiding-robot-car/
- https://stackoverflow.com/questions/41501360/getting-distance-in-inches-and-cm-from-ultrasonic-sensor-in-arduino
- https://www.amazon.co.uk/Youmile-Inductive-Proximity-Leveling-Bracket/dp/B098B4XFM1
- https://www.youtube.com/watch?v=1n_KjpMfVT0&t=1s&ab_channel=DIYBuilder