Maze solving car

Introducere

Proiectul constă în dezvoltarea unei mașinuțe ce poate rezolva un labirint folosind un Arduino Uno. Această mașinuță este capabilă să navigheze printr-un labirint folosind senzori și să găsească o cale către ieșire.

Ideea proiectului a pornit de la un interviu tehnic unde am fost întrebată cum se rezolvă un labirint (ca întrebare think outside the box). Ulterior, am văzut roboței maze-solver care participau la concursuri și m-am gândit să încerc și eu să realizez unul.

Ca utilitate, proiectul încurajează dezvoltarea creativității prin găsirea unor mijloace ingenioase de rezolvare a labirintului, însă poate reprezenta și o modalitate de divertisment.

Descriere generală

Proiectul implică controlarea unei mașinuțe care este echipată cu senzori de infraroșu și motoare. Mașinuța este apoi programată să rezolve un labirint autonom, folosindu-se de acești senzori și motoare pentru a se deplasa în labirint și a ajunge la destinație.

Hardware Design

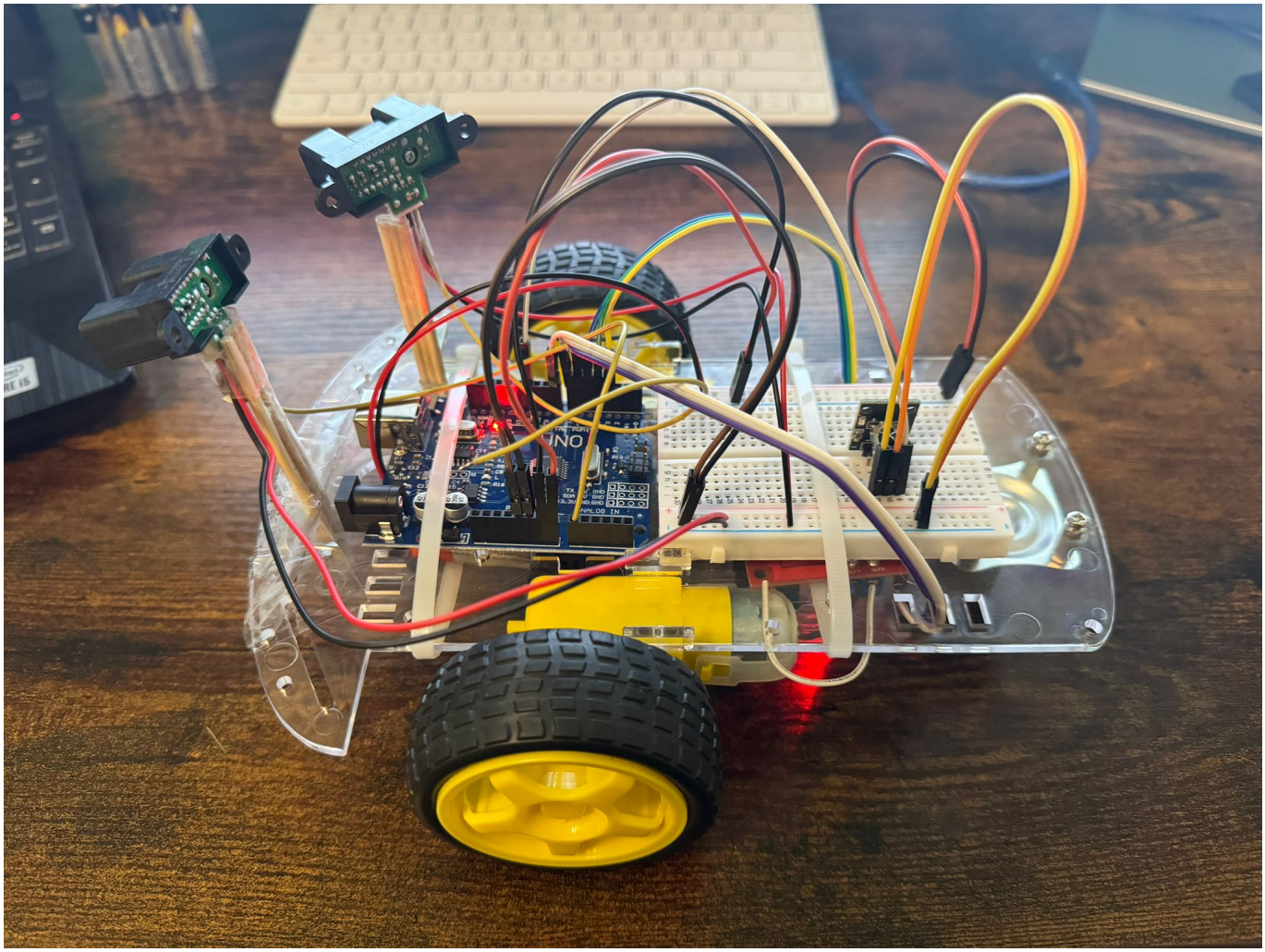
Piese:

- Arduino UNO
- 2 x Motor DC gearmotors
- 2 x Roata diametru 65mm
- 1 x Roata mobila
- 2 x Senzor analogic cu infrarosu pentru distanta SHARP
- 1 x Șasiu
- 1 x Driver motoare L298N (punte H dublă)
- 1 x led galben
- 1 x rezistenta 220

Board:

Circuit:

Design:



Software Design

Mediu de Dezvoltare

- Arduino IDE

Librării și surse 3rd-party

- SharpIR: utilizată pentru interfațarea senzorilor de distanță Sharp IR prin ADC.

Algoritmi și structuri implementate

Controlul motoarelor:

- Controlul direcției și vitezei motoarelor prin PWM (Pulse Width Modulation) și pini de direcție.
- Funcții pentru mișcarea înainte (moveForward), înapoi (moveBackward) și oprirea motoarelor (stopMotors).
- Măsurarea distanței: Utilizarea senzorilor Sharp IR pentru a măsura distanța față de obstacolele din față și din dreapta robotului.
- Stabilirea unor praguri minime și maxime pentru distanțele detectate. Folosirea corectiilor pentru a elimina snake-like movement si a avea o traiectorie mai dreapta.

Algoritmul de control:

- Proportional Control (P): Utilizat pentru ajustarea vitezei motoarelor în funcție de eroarea dintre distanța dorită și distanța măsurată.
- Detecția obstacolelor: Dacă un obstacol este detectat în față, robotul se oprește și efectuează o serie de manevre pentru a evita obstacolul.
- Menținerea distanței față de perete: Ajustarea poziției robotului pentru a menține o distanță constantă față de peretele din dreapta.
- Interupere externă: Utilizarea unei întreruperi externe pentru a detecta schimbările unui pin specific (motorDirection2A) și pentru a trimite un semnal pe un alt pin (signalPin).

Etapa 3: Surse și funcții implementate

Inițializarea motoarelor:

Se seteaza pinii driverului pe output si se ataseaza intreruperea la un pin de directie (odata cu modificarea lui se va aprinde ulterior un led)

```
void setup() {
  pinMode(motorSpeedPinA, OUTPUT);
  pinMode(motorDirection1A, OUTPUT);
  pinMode(motorDirection2A, OUTPUT);

  pinMode(motorSpeedPinB, OUTPUT);
  pinMode(motorDirection1B, OUTPUT);
  pinMode(motorDirection2B, OUTPUT);

  analogWrite(motorSpeedPinA, 0);
  digitalWrite(motorDirection1A, HIGH);
  digitalWrite(motorDirection2A, LOW);

  analogWrite(motorSpeedPinB, 0);
  digitalWrite(motorDirection1B, LOW);
  digitalWrite(motorDirection2B, HIGH);
```

```
  attachInterrupt(digitalPinToInterrupt(motorDirection2A), handlePinChange,
CHANGE);

  Serial.begin(9600);
}
```

Controlul mișcării:

Am facut functiile de moveForward si moveBackward, fara turnLeft si turnRight pentru ca abordarea cu un indice de corectie ma restrange la a avea 2 parametri, deci pentru left si right unul din parametri va fi 0.

```
void moveForward(int leftSpeed, int rightSpeed) {
  digitalWrite(motorDirection1A, HIGH);
  digitalWrite(motorDirection2A, LOW);
  digitalWrite(motorDirection1B, LOW);
  digitalWrite(motorDirection2B, HIGH);

  analogWrite(motorSpeedPinA, leftSpeed);
  analogWrite(motorSpeedPinB, rightSpeed);
}

void moveBackward() {
  digitalWrite(motorDirection1A, LOW);
  digitalWrite(motorDirection2A, HIGH);
  digitalWrite(motorDirection1B, HIGH);
  digitalWrite(motorDirection2B, LOW);

  analogWrite(motorSpeedPinA, baseSpeed);
  analogWrite(motorSpeedPinB, baseSpeed);
}

void stopMotors() {
  analogWrite(motorSpeedPinA, 0);
  analogWrite(motorSpeedPinB, 0);
}
```

Gestionarea întreruperii:

Functia seteaza un flag, o variabila volatile bool care se va verifica constant in functia de loop.

```
void handlePinChange() {
  pin2Changed = true;
}
```

Rezultate Obţinute

O masina care iese din labirint.

Demo: https://www.youtube.com/watch?v=fPwL9fN5p2s&t=14s

Concluzii

Foarte foarte mult de munca, mai ales pe baterii de 9V unde PWM sub 100 nu mergea nici cand era bateria plina. Bateriile se consuma foarte rapid, deci pentru oricine mai foloseste acest driver recomand numai baterii de 12V speciale. Am mai avut niste problemute minore, initial am uitat sa conectez GND-ul de la driver la arduino si nu intelegeam de ce nu merg motoarele :)))), dar s-au rezolvat rapid.

Overall, a fost un proiect din care chiar am invatat multe chestii utile si sper ca a iesit bine.

Download

Arhiva:

proj_pm.zip

Jurnal

Puteți avea și o secțiune de jurnal în care să poată urmări asistentul de proiect progresul proiectului.

Bibliografie/Resurse

Listă cu documente, datasheet-uri, resurse Internet folosite, eventual grupate pe **Resurse Software** şi **Resurse Hardware**.

Export to PDF