Roomba

Introducere

Voi implementa un robot de aspirat. Spre deosebire de un Roomba adevărat, robotul de aspirat are forma unei mașinuțe, va folosi un senzor ultrasonic si un servomotor ca sa evite obstacolele, si va face viraje pe loc, deoarece fiecare roata va fi controlată independent folosind PWM.
Pentru partea de aspirat, voi folosi o sticla ca suport și un motoraș cu ventilator.
Considerând camera dreptunghiulară, robotul va încerca sa o parcurgă pe toată in zig-zag. Se deplasează înainte, pana de de un obstacol (peretele), moment în care investighează direcțiile posibile, folosind senzorul ultrasonic, si alege calea cea mai libera.
Scopul robotului e sa automatizeze curățenia prin casa.

Probleme ce pot apărea:

- Cu cât sunt mai multe obstacole in camera, cu atât robotul va da mai puțin randament, deoarece ii trebuie un algoritm mai inteligent de evitare a obstacolelor si recalcularea traiectoriei.
- Nu se creează suficient vid in recipient pentru a aspira.

Descriere generală

Schema bloc

Hardware Design

Lista de piese

- Arduino UNO R3 (ATmega328p + CH340)
- Shield cu Driver de Motoare L293D
- Motor cu reductor si roata
- Micro Servomotor SG90

- Senzor ultrasonic distanta HC-SR04
- Motor in miniatura 3-6V
- Elice
- Powerbank

Schema electrica

Conexiuni între pini

Shield-ul pentru motoare se atașează direct pe Arduino. În urma consultării datasheet-ului pentru Adafruit Motor Shield V1, acesta folosește următorii pini:

- M1 - D11
- M2 - D3
- M3 - D5 (nu a fost folosit in varianta finala, dar e conectat prin shield)
- M4 - D6 (nu a fost folosit in varianta finala, dar e conectat prin shield)

Pinii menționați mai sus sunt folosiți pentru controlul **PWM** al motoarelor.

Următorii pini sunt folosiți mereu de shield, indiferent dacă sunt conectate motoare sau indiferent de câte sunt conectate, deoarece sunt conectați la latch-ul 74HC595:

- D4
- D7
- D8
- D11

Pinii senzorului ultrasonic au fost legați astfel:

- TRIG (2) - D13 pe Arduino
- ECHO (3) - D2 pe Arduino
- VCC (1) - 5V pe Arduino
- GND (4) - GND pe Arduino

Servomotorul se leagă pe shield la S1, iar conform datasheet-ului:

- Control (3) - D10 pe Arduino
- VCC (2) - 5V pe Arduino
- GND (1) - GND pe Arduino

Bateriile se leagă la **EXT_PWR** de pe shield.

Software Design

Mediu de dezvoltare

- VSCode + PlatformIO

Librarii

- Adafruit Motor Shield Library - AFMotor.h

Librăria **AFMotor.h** expune o interfața pentru a controla motoare prin intermediul shield-ului ales.
Librăria pune la dispoziție clasa **AF_DCMotor**, care primește ca parametru un **număr de motor**, scris pe shield, si oferă funcții pentru controlul motoarelor, precum **run** sau **setSpeed**.
AFMotor.h este un wrapper pentru **PWM**.

- Arduino Servo Library - Servo.h

Librăria **Servo.h** expune o interfața pentru a lucra mai ușor cu servomotoarele.
Librăria pune la dispoziție clasa **Servo**, si o serie de metode prin care servomotorul este configurat si utilizat.
Pentru a specifica pin-ul unde e conectat servomotorul se folosește metoda **attach**.
Pentru a controla servomotorul, avem metoda **write**, care folosește in spate **PWM** si **Timere**.

Structura codului

Robotul Roomba va funcționa pe baza unui FSM, cu următoarele stări:

- **IDLE** - Stare de setup. Robotul este oprit și poziționează senzorul ultrasonic în poziția centrală. Trece apoi în starea de deplasare înainte (GO_FORWARD).
- **GO_FORWARD** - Robotul se deplasează înainte continuu. La intervale regulate (CHECK_INTERVAL), măsoară distanța față de obstacol. Dacă distanța este sub un prag critic (CRITICAL_DISTANCE), trece în starea GO_BACKWARD. Dacă distanța este sub un prag mai mare (DIST_THRESHOLD), trece în starea OBSTACLE_DETECTED. Pragul critic scade șansa robotului de a se bloca in colțuri sau locuri inaccesibile pentru el.
- **GO_BACKWARD** - Robotul se deplasează înapoi pentru o perioadă fixă (BACKWARD_TIME) pentru a se distanța de obstacol. La finalul perioadei, trece în starea OBSTACLE_DETECTED.

- **OBSTACLE_DETECTED** - Robotul oprește motoarele și pregătește scanarea mediului în jur pentru a decide o direcție sigură de deplasare. Trecerea se face către starea SCAN_ALL.
- **SCAN_ALL** - Robotul efectuează o scanare cu servomotorul în mai multe direcții pentru a colecta date despre distanțele în stânga, centru și dreapta. După scanare, trece în starea DECIDE_DIRECTION.
- **DECIDE_DIRECTION** - Pe baza datelor scanate, robotul determină unghiul optim pentru a vira, calculând un offset față de poziția centrală a servomotorului și durata necesară a virajului. În funcție de direcție, se trece în starea TURN_LEFT sau TURN_RIGHT.
- **TURN_LEFT** - Robotul efectuează o virare la stânga pentru durata calculată. După finalizarea virajului, oprește motoarele și revine la deplasarea înainte (GO_FORWARD).
- **TURN_RIGHT** - Robotul efectuează o virare la dreapta pentru durata calculată. După finalizarea virajului, oprește motoarele și revine la deplasarea înainte (GO_FORWARD).

Controlul Motoarelor

Funcțiile pentru controlul motoarelor sunt in headerul **MotorControl.h**.
Am scris niște API-uri pentru a controla mai ușor mișcarea robotului, folosindu-ma de metodele expuse in **AFMotor.h**:

```
void stopMotors();
void forwardMotors();
void backwardMotors();
void turnLeftMotors();
void turnRightMotors();
```

Măsurarea distanței

Funcțiile pentru măsurarea distanței sunt in headerul **Sonar.h**.
Am creat un API pentru a controla senzorul ultrasonic, folosind cunoștințele dobândite in laboratoarele de **GPIO**, **Timere** și **Întreruperi**. Am vrut sa folosesc Timer2 pentru a măsura timpul fără **millis**, dar suprascrierea ISR-ului de Overflow intervine peste librăria ce expune API-ul pentru motoare si apăreau conflicte. Totuși, am folosit întreruperi pentru măsurarea distanței.

```
void initSonar(void);
void initEchoInterrupt(void);
void sendTriggerPulse(void);
uint16_t measureDistance(void);
```

Cum funcționează ?
Se generează o întrerupere pe pinul la care este legat ECHO al senzorului ultrasonic la fiecare schimbare a semnalului.
Când TRIG emite unda, ECHO trece pe HIGH, iar când se întoarce unda, ECHO trece pe LOW. Se retine timestampul pentru fiecare dintre aceste evenimente si se face diferența, pentru a calcula timpul

dus-întors al undei. Știind viteza sunetului, si timpul dus-întors, putem calcula ușor distanța parcursă.

Alegerea unei noi direcții

Pentru alegerea unei noi direcții, FSM-ul din main folosește mai multe API-uri, definite in **ServoControl.h**:

```
void initServoControl(void);
void lookAtRightInstant(int angle);
void lookAtLeftInstant(int angle);
void lookCenter(void);
void initLookPoints(void);
void clearLookPoints(void);
void scanAllLookPoints(void);
int getBestAngle(void);
```

Robotul alege direcția prin rotirea servomotorului la mai multe unghiuri predefinite pentru a scana distanțele din stânga, centru și dreapta. Pe baza acestor măsurători, se selectează unghiul cu cea mai mare distanță liberă, iar robotul se va orienta spre acea direcție pentru a evita obstacolele.

Rezultate Obţinute

- Robotul a fost capabil sa evite obstacole si sa aleagă bine o noua direcție de deplasare.
- Sistemul de vid funcționează parțial – eficiența de aspirare este limitată (aproape inexistentă).
- Controlul motoarelor și al servomotorului este stabil.
- Senzorul ultrasonic detectează corect peretele și obstacolele.

Concluzii

Proiectul demonstrează posibilitatea realizării unui robot de curățenie funcțional, cu un design modular și scalabil. Limitările țin mai ales de lipsa unui algoritm mai sofisticat pentru detecția și ocolirea obstacolelor, precum și de eficiența sistemului de aspirare.

Download

O arhivă (sau mai multe dacă este cazul) cu fişierele obţinute în urma realizării proiectului: surse, scheme, etc. Un fişier README, un ChangeLog, un script de compilare şi copiere automată pe uC crează întotdeauna o impresie bună .

Fişierele se încarcă pe wiki folosind facilitatea **Add Images or other files**. Namespace-ul în care se încarcă fişierele este de tipul **:pm:prj20??:c?** sau **:pm:prj20??:c?:nume_student** (dacă este cazul). **Exemplu:** Dumitru Alin, 331CC → **:pm:prj2009:cc:dumitru_alin**.

Jurnal

Video demonstrativ

https://www.youtube.com/shorts/H-P-53NL3jc

Imagini

Bibliografie/Resurse

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

Resurse Hardware

https://cdn-learn.adafruit.com/downloads/pdf/adafruit-motor-shield.pdf

https://ww1.microchip.com/downloads/en/DeviceDoc/Atmel-7810-Automotive-Microcontrollers-ATmega328P_Datasheet.pdf

https://www.circuito.io/blog/arduino-uno-pinout/

Resurse software

https://github.com/adafruit/Adafruit-Motor-Shield-library/tree/master

Cod sursa

https://github.com/cristiantudor1607/RoombaCar

Export to PDF