Robotic arm

Introducere

Proiectul meu este un brat robotic pe care il construiesc folosind servo-motoare, un LCD, LED-uri si un microcontroller. Poate sa ridice si sa mute obiecte mici, iar scopul lui e sa ma ajute sa invat cum functioneaza partea practica din robotica – de la mecanica si electronica pana la programare.

Am pornit de la ideea de a face ceva care se misca, ceva mai interactiv decat proiectele clasice. Mi-am dorit sa invat mai bine cum se controleaza miscarea unor piese reale si cum se imbina toate componentele intr-un sistem functional.

Cred ca e un proiect util pentru mine, pentru ca ma ajuta sa inteleg mai bine tot ce tine de automatizari si control. Poate fi de folos si altora care vor sa inceapa cu ceva simplu, dar educativ, fara sa investeasca mult in materiale sau echipamente complicate.

Descriere generală

Arduino-ul e centrul de control al intregului sistem. De la el pleaca comenzile catre cele patru servo-motoare care misca bratul robotic in diferite directii, in functie de cum e actionat joystick-ul. Joystick-ul e practic “mana” utilizatorului, iar bratul raspunde in timp real la miscarile facute.

Pe ecranul LCD sunt afisate informatii legate de starea bratului – cum ar fi pozitia sau comenzile primite. LED-urile au rol de indicatori vizuali: unul se aprinde cand clestele de prindere este inchis, iar altul cand este deschis, ca sa fie clar dintr-o privire ce face bratul. Toate componentele sunt alimentate printr-un modul de alimentare dedicat, care asigura tensiunea potrivita pentru functionarea intregului ansamblu.

Hardware Design

Lista de piese hardware pentru EEZYbotARM

- Arduino Uno
- 4 x Servo motoare
- LED-uri
- 2 x Joystick

Software Design

Mediu de dezvoltare

Am folosit **Arduino IDE** pentru toata partea de cod. E cel mai simplu mediu pentru Arduino si are tot ce trebuie - editor, compilator, si posibilitatea sa urc codul direct pe placa. Plus ca e gratis si destul de usor de folosit.

Librării şi surse 3rd-party

Codul se bazeaza pe doua librarii:

- **Servo.h** - libraria standard care vine cu Arduino IDE pentru controlul servo-urilor. Face toata treaba grea cu semnalul PWM
- **MsTimer2.h** - o librarie externa pe care am gasit-o pentru timer-e mai precise. Am avut nevoie de ea pentru rutina automata ca sa nu blocheze restul codului

Algoritmi şi structuri implementate

Controlul servo-urilor cu joystick: Am facut o mapare simpla intre valorile de la joystick (care vin intre 0-1023) si pozitiile servo-urilor. Am pus si zone moarte ca sa nu se miste bratul din orice atingere mica a joystick-ului. Fiecare axa controleaza cate un servo, cu limite ca sa nu fortez motoarele.

Sistem cu intreruperi pentru butoane: Am folosit intreruperi hardware ca sa detectez cand apas butoanele. Am pus si un fel de debouncing simplu - verific ca au trecut cel putin 200ms intre apasari ca sa nu se activeze de mai multe ori la o singura apasare.

Rutina automata: Aici am facut o masina de stari care executa o secventa fixa de miscari. Fiecare pas are timpul lui si se trece automat la urmatorul. Rutina simuleaza ridicarea unui obiect de pe masa, rotirea bazei si punerea obiectului in alta parte.

Filtrare pentru joystick-ul bazei: Pentru joystick-ul care controleaza rotatia am implementat o medie pe ultimele 5 citiri. Asta elimina zgomotul si face miscarea mai fluida. Servo-ul pentru baza e de tip rotatie continua, deci functioneaza diferit - nu merge la pozitii fixe ci se roteste continuu.

LED-uri indicatoare: LED-urile imi arata ce se intampla: LED-ul rosu se aprinde cand rutina automata ruleaza, LED-ul verde e aprins in mod normal, iar cand clestele e inchis LED-ul rosu clipeste.

Surse şi funcţii implementate

Link git: https://github.com/Robert2003/RoboticArm.git

Am organizat codul in cateva functii principale:

- `controlArmMovement()` - citeste joystick-ul si misca servo-urile pentru forward/backward si up/down
- `controlBaseRotation()` - controleaza rotatia bazei cu filtrare pentru miscari mai line
- `executeRoutine()` - ruleaza secventa automata la apasarea joystick-ului
- `updateLEDs()` - schimba starea LED-urilor in functie de ce face bratul
- Functii ISR pentru intreruperi care seteaza flag-uri pentru butoane

Toata logica e organizata ca sa nu blocheze executia - folosesc flag-uri si timer-e ca sa pot face mai multe lucruri simultan fara sa se incurce.

Bucati de cod la care mi-a placut sa lucrez:

Deadzone pentry joystick, altfel se miscau incontinuu servo-urile:

```
if (abs(baseAverage - centerValue) > baseDeadzone) {
  if (baseAverage > centerValue + baseDeadzone) {
    baseRotationServo.write(100);
  } else if (baseAverage < centerValue - baseDeadzone) {
    baseRotationServo.write(80);
  }
} else {
  baseRotationServo.write(continuousStopPos);
}
```

Control manual cu joystick pe 2 axe. Am ales increment de 2, pentru ca daca era mai mic se misca sacadat, daca era mai mare, se misca prea repede.

```
if (abs(joystickX - joystickCenter) > deadzone) {
  if (joystickX > joystickCenter + deadzone && forwardBackwardPos < 180) {
    forwardBackwardPos += 2;
    forwardBackwardServo.write(forwardBackwardPos);
  } else if (joystickX < joystickCenter - deadzone && forwardBackwardPos > 0)
{
    forwardBackwardPos -= 2;
    forwardBackwardServo.write(forwardBackwardPos);
  }
```

```
}
```

Codul de la LED-uri. Cand se executa rutina automata, LED-ul rosu palpaie. Cand gheara e inchisa, LED-ul rosu sta aprins. LED-ul verde arata ca bratul poate fi operat, adica atunci cand nu e in rutina automata.

```
if (routineActive) {
  digitalWrite(redLED, HIGH);
  digitalWrite(greenLED, LOW);
} else {
  digitalWrite(greenLED, HIGH);
  if (!clawOpen) {
    digitalWrite(redLED, (millis() / 500) % 2);
  } else {
    digitalWrite(redLED, LOW);
  }
}
```

Rezultate Obţinute


AWS
SAM
500g


New Line
9600 baud
500g

Concluzii

Download

Jurnal

Bibliografie/Resurse

Export to PDF