Senzor de distanță cu barieră automată

Nume studenti: Andrei Danut Petcu

Chihalau Gabriel-Eduard

Adrese e-mail: gabriel-eduard.chihalau@student.tuiasi.ro

andrei-danut.petcu@student.tuiasi.ro

Rezumat și motivația alegerii temei

Am ales acest proiect deoarece reprezintă un exemplu practic şi uşor de implementat din domeniul sistemelor înglobate (embedded), care combină detecţia de obstacole folosind un senzor ultrasonic cu acţiunea unui servomotor. Este o simulare simplificată a unei bariere automate care reacţionează la apropierea unui vehicul. Astfel, se poate demonstra uşor funcţionarea unor concepte fundamentale: generarea unui semnal PWM pentru servo, măsurarea duratei unui puls pentru a estima distanţa şi logica decizională de control.

Analiza - design - implementare

- Analiza:

Scopul proiectului este să detecteze un obiect aflat la o distanţă mai mică de 10 cm folosind un senzor ultrasonic HC-SR04, iar apoi să ridice o barieră printr-un servomotor pentru 3 secunde, după care o coboară automat.

- Design:
  Sistemul include:
  • Raspberry Pi Pico
  • Senzor ultrasonic HC-SR04
  • Servomotor
  • Breadboard și fire de conexiune
  Conexiuni:
  • TRIG -> GP17
  • ECHO -> GP18
  • Servo -> GP16

- Implementare:
  Codul este scris în MicroPython. Ciclul principal măsoară continuu distanța. Dacă este detectat un obstacol la mai puțin de 10 cm, bariera se ridică și apoi se coboară după ce obstacolul dispare.

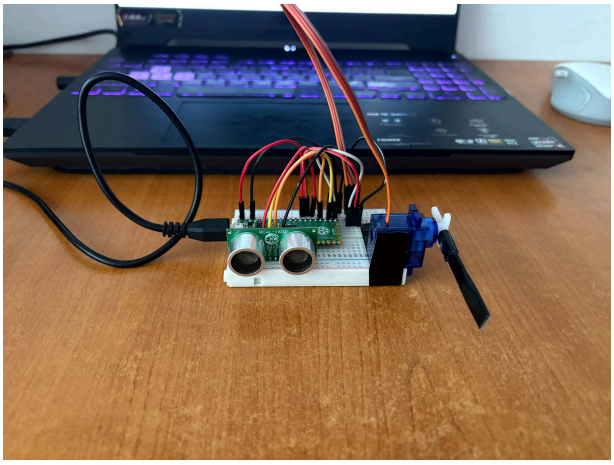
Codul sursă (dda.py)

```
from machine import Pin, PWM, time_pulse_us
from time import sleep

servo = PWM(Pin(16))
servo.freq(50)

trig = Pin(17, Pin.OUT)
echo = Pin(18, Pin.IN)

def opreste():
    servo.duty_u16(4800)
    sleep(0.1)
    servo.deinit()

def misca_90_grade():
    servo.duty_u16(3000)
    sleep(0.165)
    opreste()

def revino_la_start():
    global servo
    servo = PWM(Pin(16))
    servo.freq(50)
    servo.duty_u16(7000)
    sleep(0.16)
    servo.duty_u16(4800)
    sleep(0.1)
```

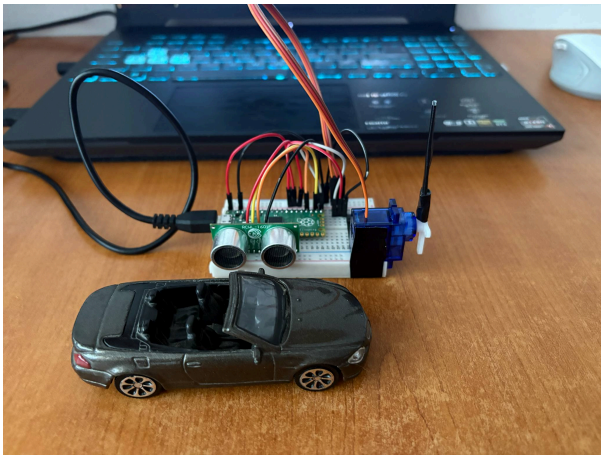
```
    servo.deinit()

def masoara_distantă():
    trig.low()
    sleep(0.002)
    trig.high()
    sleep(0.00001)
    trig.low()

    durata = time_pulse_us(echo, 1, 30000)
    distanta_cm = (durata / 2) / 29.1
    return distanta_cm

while True:
    dist = masoara_distantă()
    print("Distanta:", dist, "cm")
    if dist < 10:
        print("🚗 Maşină detectată! Ridic bariera...")
        misca_90_grade()
        sleep(3)

        # Aşteptăm până nu mai e nicio maşină sub barieră
        while True:
            dist_noua = masoara_distantă()
            print("Verific din nou distanţa:", dist_noua, "cm")
            if dist_noua > 10:
                print("✅ Maşina a trecut. Cobor bariera.")
                revino_la_start()
                break
            else:
                print("⛔ Maşina încă e acolo! Aştept...")
                sleep(1)
```

Detalii secvență demonstrativă

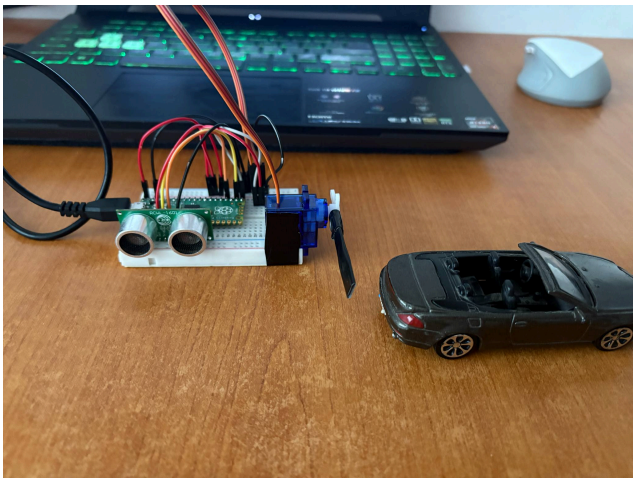
Secvenţa demonstrativă este prezentată prin imagini. Acestea surprind:
- Poziţia iniţială a barierei;



- Ridicarea barierei când o maşină se apropie;



- Coborârea barierei după 3 secunde, dacă nu mai este niciun obstacol.



Probleme întâmpinate și modul de rezolvare

- Senzorul ultrasonic returna uneori valori instabile — rezolvat prin delay suplimentar.
- Servomotorul nu revenea corect — calibrat valorile de PWM.

Ce se învață dacă se replică proiectul

- Utilizarea senzorilor ultrasonic HC-SR04
- Controlul servomotoarelor cu PWM
- Programare în MicroPython pentru Raspberry Pi Pico
- Integrarea hardware-software într-un sistem embedded simplu

Bibliografie

- https://www.raspberrypi.com/documentation/microcontrollers/micropython.html
- https://randomnerdtutorials.com/micropython-hc-sr04-ultrasonic-esp32-esp8266/
- https://how2electronics.com/control-servo-motor-using-micropython-and-raspberry-pi-pico/
- https://chat.openai.com/