

Nadolu Alexandru: Parking Sensor

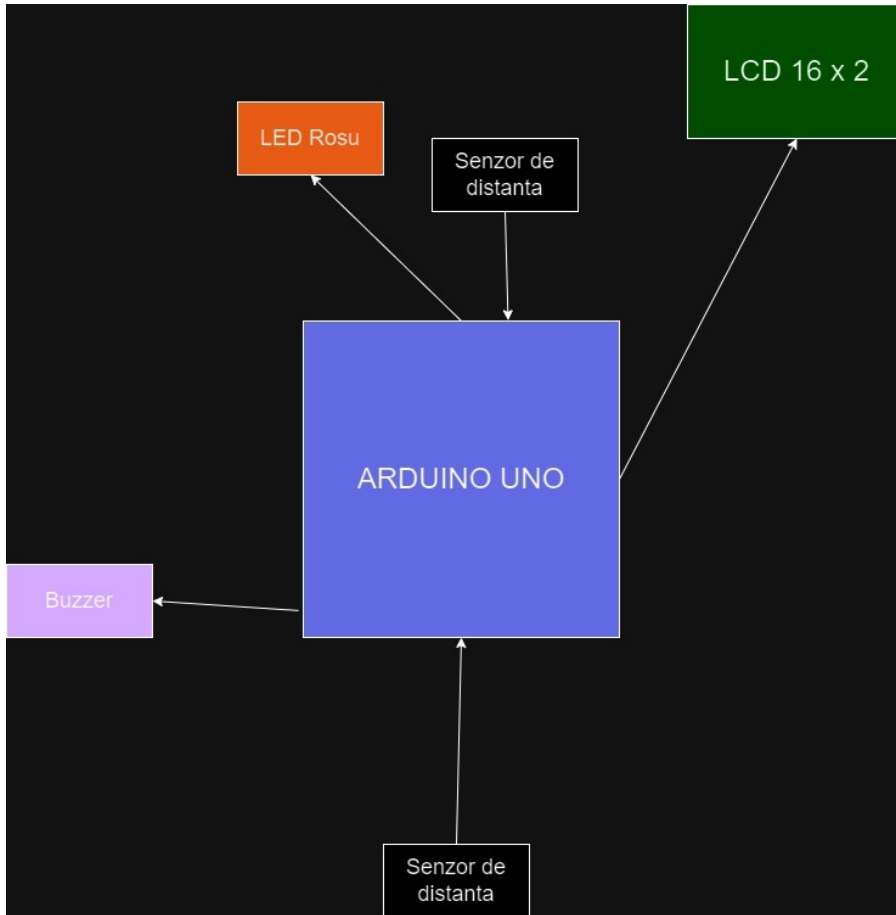
Introducere

Proiectul constă în dezvoltarea unui sistem de asistență la parcare, care utilizează senzori pentru detectarea obstacolelor din mediul înconjurător și oferă feedback vizual și auditiv șoferului pentru a-l ajuta să parcheze în condiții de siguranță. Ideea de la care am pornit acest proiect este chiar cea mai esențială opțiune pe care o poate avea un autovehicul, senzorul de parcare.

Descriere generală

Sistemul va avea următoarele funcționalități:

- Detectarea distanței față de obstacole: Sistemul va măsura distanța dintre vehicul și obstacolele aflate atât în față, cât și în spatele acestuia.
- Indicarea distanței utilizând un LED, un buzzer și un ecran LCD.
- Frecvența cu care LED-ul va executa o serie de cicluri de aprindere și stingere și sunetul produs de buzzer vor varia în funcție de distanța vehiculului față de obstacole.
- Intensitatea LED-ului va depinde de distanța senzorului până la cel mai apropiat obstacol.



Hardware Design

Componente:

- 1x Arduino UNO
- 2x Senzor de distanta HC-SR04
- 1x Buzzer
- 1x Ecran LCD
- 1x LED
- 1x Breadboard
- 1x Rezistenta 430Ω



Proiectul in beta:



Software Design

Descrierea aplicației (firmware):

Această aplicație este destinată unui sistem de asistență la parcare pentru un vehicul. Utilizează un ecran LCD pentru afișarea distanțelor măsurate de senzorii ultrasunete montați în față și în spatele vehiculului. Un buzzer și un LED sunt folosite pentru a avertiza șoferul în cazul în care există un obstacol la o distanță periculoasă.

Cod:

```
#include <Wire.h>
#include <LiquidCrystal_I2C.h>

// LCD-ul este initializat de la adresa 0x3F
LiquidCrystal_I2C lcd(0x3F, 16, 2);
```

```
// pinii pentru HC-SR04
const int trigPinFront = 13;
const int echoPinFront = 12;
const int trigPinBack = 4;
const int echoPinBack = 3;

// distantele fata de obstacolele din fata si din spatele vehiculului
float distanceFront, distanceBack;

// pinul pentru buzzer
#define speakerPin 9

// pinul pentru led
#define ledpin 10

// variabile pentru masurarea timpului
unsigned long previousMillis = 0;
const long interval = 1000; // 1s

void setup() {
    Serial.begin(9600);

    // setez pinul ledului pe output
    pinMode(ledpin, OUTPUT);

    // initializez LCD-ul
    lcd.begin();
    lcd.backlight();

    // setez cursorul pe prima linie
    lcd.setCursor(0, 0);

    // initializez pinul pentru buzzer si ma asigur ca e oprit la inceput
    pinMode(speakerPin, OUTPUT);
    digitalWrite(speakerPin, LOW);

    // initializez pinii pentru senzori
    // trigger ON = incep sa transmit ultrasunete, echo masoara durata
    // semnalului
    pinMode(trigPinFront, OUTPUT);
    pinMode(echoPinFront, INPUT);

    pinMode(trigPinBack, OUTPUT);
    pinMode(echoPinBack, INPUT);
}

float getDistance(int trigPin, int echoPin) {
    // trimite un impuls pe pinul trig, ma asigur inainte ca trig este in
    // repaus
    digitalWrite(trigPin, LOW);
    delayMicroseconds(2);
```

```
digitalWrite(trigPin, HIGH);
delayMicroseconds(10);
digitalWrite(trigPin, LOW);

// masoara durata pe pinul echo
long duration = pulseIn(echoPin, HIGH);

// distanta = durata (dus - intors) * viteza de propagare a sunetului
float distance = (duration / 2.0) * 0.0343;

// returnez distanta
return distance;
}

void loop() {
  unsigned long currentMillis = millis();
  if (currentMillis - previousMillis >= interval) {
    previousMillis = currentMillis;

    // distanta pentru senzorul din fata
    distanceFront = getDistance(trigPinFront, echoPinFront);

    // distanta pentru senzorul din spate
    distanceBack = getDistance(trigPinBack, echoPinBack);

    // afisez distantele pe LCD
    lcd.clear();
    lcd.setCursor(0, 0);
    lcd.print("Front: ");
    lcd.print(distanceFront);
    lcd.print(" cm");

    // mut cursorul pe al doilea rand al ecranului
    lcd.setCursor(0, 1);
    lcd.print("Back: ");
    lcd.print(distanceBack);
    lcd.print(" cm");
  }

  // verific care este cea mai mica distanta pana la obstacol (fata sau
  spate)
  float minDistance = min(distanceFront, distanceBack);

  if (minDistance <= 20) {
    // calculez intensitatea led-ului in functie de distanta
    int ledIntensity = map(minDistance, 1.67, 20, 255, 1);

    // emit semnale acustice
    int buzzerSpeed = map(minDistance, 1, 20, 100, 500);
    unsigned long beepEnd = millis() + interval;
  }
}
```

```
while (millis() < beepEnd) {
  // setez intensitatea LED-ului
  analogWrite(ledpin, ledIntensity);
  analogWrite(speakerPin, 255);
  delay(buzzerSpeed / 2);
  analogWrite(speakerPin, 0);
  analogWrite(ledpin, 0);

  delay(buzzerSpeed / 2);
}
} else {
  // opresc semnalele acustice si luminoase
  digitalWrite(speakerPin, LOW);
  analogWrite(ledpin, 0);
}
}
```


Rezultate Obținute

Un videoclip cu proiectul se poate vedea mai jos:

Concluzii

Un proiect la care am lucrat de drag, foarte challenging si frumos, de care imi voi aminti cu placere.

Download

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

Fișierele se încarcă pe wiki folosind facilitatea *Add Images or other files. Namespace-ul în care se încarcă fișierele este de tipul *:pm:prj20??:c?* sau *:pm:prj20??:c?:nume_student** (dacă este cazul). *Exemplu:* Dumitru Alin, 331CC → *:pm:prj2022:cc:dumitru_alin*.

Jurnal

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

Bibliografie/Resurse

- <https://ocw.cs.pub.ro/courses/pm/lab/lab0-2023>
- <https://ocw.cs.pub.ro/courses/pm/lab/lab3-2023-2024>
- <https://ocw.cs.pub.ro/courses/pm/lab/lab6-2023-2024>
- <https://www.youtube.com/watch?v=0Lhgd8PQmn0>
- <https://www.ardumotive.com/how-to-use-a-buzzer-en.html>

[Export to PDF](#)

From:
<http://ocw.cs.pub.ro/courses/> - **CS Open CourseWare**

Permanent link:
<http://ocw.cs.pub.ro/courses/pm/prj2024/ddosaru/nadolu.alexandru>



Last update: **2024/05/26 18:19**