Sistem de monitorizare a numărului de locuri libere într-o parcare, implementat cu ajutorul unei machete demonstrative și incluzând o barieră comandată de un servomotor

Sisteme Incorporate

An studiu : III, CTI-Ro,Semigrupa 6.2

Studenți:

Ștefanică Nicoleta

Șușară Eliana

Cuprins

Descrierea proiectului

Sistemul de monitorizare a numărului de locuri libere într-o parcare ce include o barieră comandată de un servomotor funcționeaza în funcție de maşinile care intră si ies, detectate de senzori cu ultrasunete. Mașinile sunt detectate în urma unui proces de calculare a distanței la care ele sunt poziționate, folosindu-ne de senzori cu ultrasunete.

Bariera comandată de un servomotor va fi ridicată în momentul în care senzorul cu ultrasunet detectează mașina la o distanță de 10 cm. Acest lucru se întamplă atât pentru intrarea cât și pentru ieșirea a mașinii din parcare .

Afişarea numărului de locuri libere din parcare va fi realizată printr-un afişaj LCD,numarul inițial al locurilor de parcare fiind prestabilit.

Aplicaţia mai are în componența sa doua LED-uri: unul din ele va fi pornit atâta vreme cât mai există locuri libere, al doilea va fi pornit când nu vor mai fi locuri disponibile în cadrul parcării.

Descrierea plăcii.Caracteristici generale componente

Descrierea plăcii si a microcontrolerului ATmega328P

Placa de dezvoltare utilizată în acest proiect este Arduino Uno R3 care utilizeaza microcontrolerul ATmega328P.

ATmega328P este un microcontroler pe 8 biți din familia AVR, recunoscut pentru eficiența, flexibilitatea și ușurința de utilizare în diverse proiecte de electronică.

Specificații Tehnice

a) Arhitectură: AVR 8-bit RISC
b) Viteză de ceas: 16 MHz
c) Memorie Flash: 32 KB (din care 0.5 KB sunt utilizați de bootloader)
d) SRAM: 2 KB
e) EEPROM: 1 KB
f) Număr de pini I/O:
    - 14 pini digitali (dintre care 6 pot fi folosiți ca ieșiri PWM)
    - 6 pini analogici

Interfețe și Periferice

a) Interfețe de comunicație:
    - UART: 1 (pentru comunicație serială)
    - SPI: 1 (pentru comunicație de mare viteză cu alte dispozitive)
    - I2C: 1 (pentru comunicație cu senzori și alte periferice)

b) Conversie Analog-Digital (ADC):

- 6 canale ADC pe 10 biți, utilizate pentru a converti semnalele analogice în date digitale

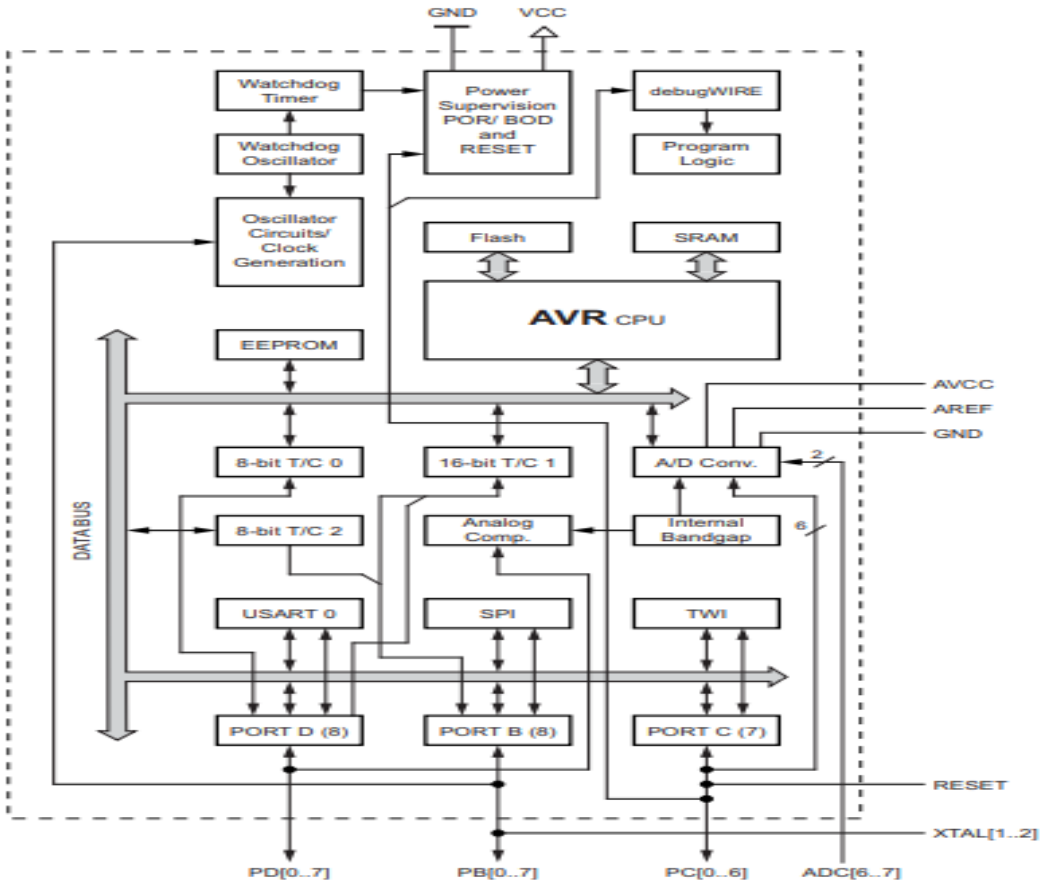
c) Timers:

- 3 Timere: 1 timer pe 8 biți și 2 timere pe 16 biți, folosiți pentru generarea de unde PWM, măsurători de timp și alte aplicații temporale

d) PWM (Pulse Width Modulation):

- 6 canale PWM pentru controlul motoarelor, reglarea luminozității LED-urilor și alte aplicații de modulație a lățimii impulsului

Schema bloc microcontroler



Caracteristici generale componente

Tabel componente utilizate

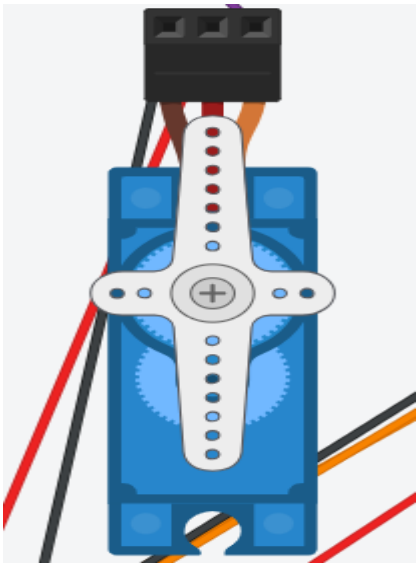
Name	Quantity	Component
U1	1	Arduino Uno R3
DIST1 DIST2	2	Ultrasonic Distance Sensor (4-pin)
SERVO3	1	Positional Micro Servo
U2	1	PCF8574-based, 32 (0x20) LCD 16 x 2 (I2C)
D1	1	Red LED
R1 R2	2	470 Ω Resistor
D2	1	Green LED

1.1. Senzorul Ultrasonic de Distanță

Senzorul de Distanță Ultrasonic HC-SR04 este utilizat pentru a măsura distanța față de un obiect folosind unde sonore. Acesta funcționează la 5V DC și consumă mai puțin de 2mA. Senzorul poate măsura distanțe cuprinse între 2 cm și 400 cm, cu o acuratețe de ±3 mm și are un unghi de detecție de 15 grade. Funcționează prin trimiterea unui impuls ultrasonic de 40 kHz de la pinul de declanșare.

1.2. Micro Servo

Micro Servo-ul Positional este un motor mic și versatil utilizat frecvent în proiectele Arduino pentru controlul precis al poziției. Funcționează la 4.8V până la 6V DC și poate roti între 0 și 180 de grade. Controlul se realizează prin semnale PWM (modulație a lățimii impulsurilor), unde durata impulsului determină unghiul de rotație. Micro Servo-ul are un cuplu suficient pentru a mișca obiecte mici și este ideal pentru aplicații cum ar fi controlul roților, articulațiilor robotice, și a clapelor sau pârghiilor în modele de aviație și proiecte de automatizare.



1.3. Led

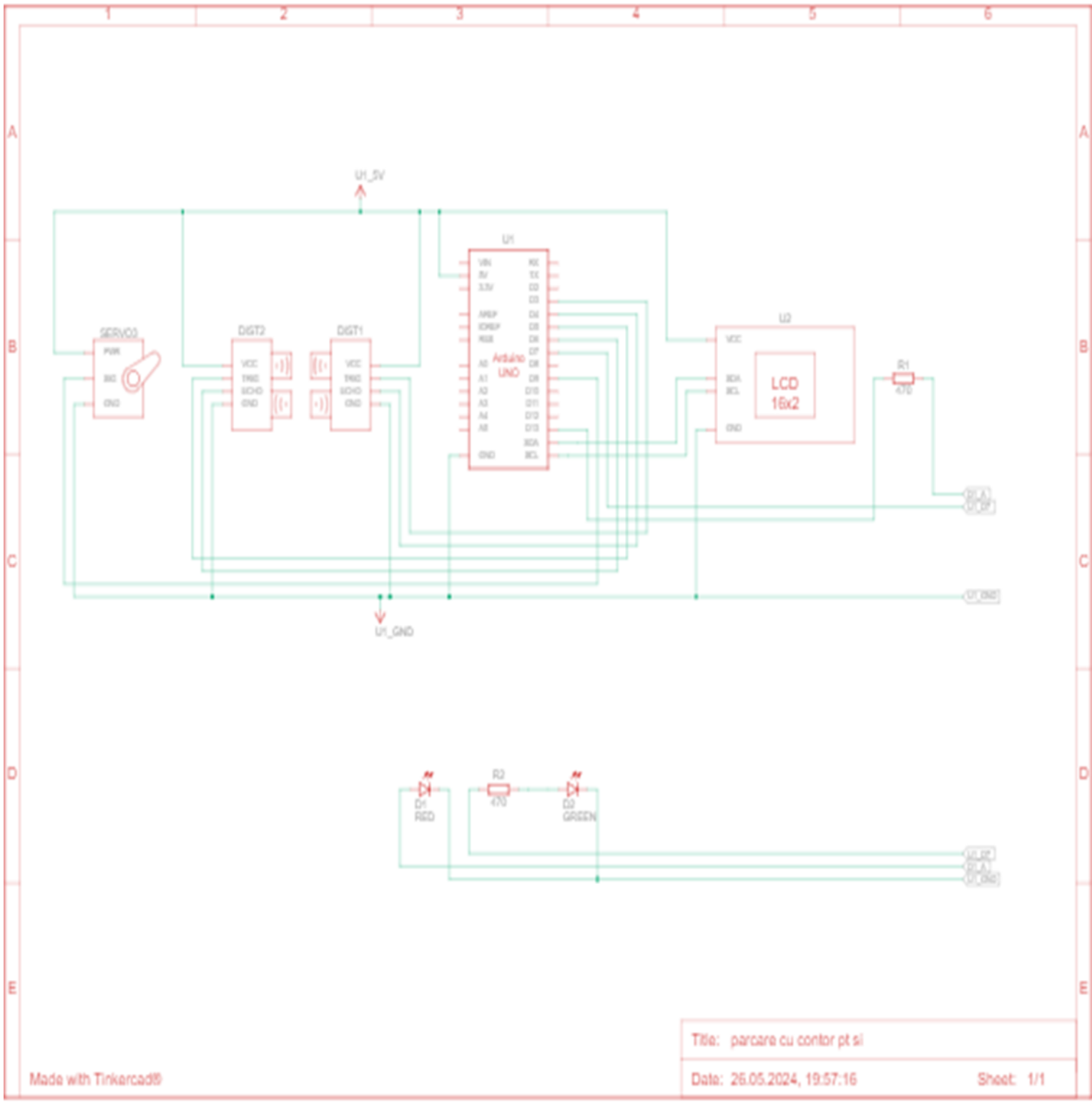
Un LED (Diodă Emitătoare de Lumină) este un dispozitiv electronic semiconductor care emite lumină atunci când este alimentat cu curent electric. Funcționează la o tensiune tipică de 2-3V și un curent de 10-20mA. LED-urile sunt disponibile într-o varietate de culori și intensități, și sunt utilizate pe scară largă în proiectele Arduino pentru indicarea stării, iluminare sau semnalizare. Sunt eficiente din punct de vedere energetic, au o durată de viață lungă și sunt ușor de controlat prin intermediul unui pin digital al plăcii Arduino. Integrarea unui rezistor în serie este necesară pentru a limita curentul și a proteja LED-ul de supraîncărcare.

1.4. LCD

Un LCD de 16x2 bazat pe PCF8574 este un afișaj cu cristal lichid care poate afișa două rânduri de câte 16 caractere, utilizat frecvent în proiectele Arduino pentru a vizualiza text și date. Acesta utilizează un modul I2C (Inter-Integrated Circuit) cu adresa implicită 0x20(in IDE-ul de la Arduino vom utiliza adresa 0x27) , care simplifică conexiunile prin reducerea numărului de pini necesari la doar patru (VCC, GND, SDA, SCL).

LCD-ul oferă un mod eficient de a prezenta informații despre locurile de parcare disponibile și este compatibil cu biblioteca LiquidCrystal_I2C din Arduino IDE, care facilitează programarea.

Arhitectura sistemului - conectarea hardware însoţită de descriere



Această schemă hardware ilustrează un sistem de monitorizare a locurilor de parcare, care utilizează un microcontroler Arduino pentru a controla o barieră, detecta mașini și afișa numărul de locuri disponibile. Componentele principale includ un servomotor, senzori cu ultrasunete, un afișaj LCD, și două LED-uri pentru semnalizare.

1.Componente si conexiuni

1. Microcontroler (Arduino UNO) - U1
    - Pin 5V și GND: Alimentarea microcontrolerului.
    - Pini Digitali: Conectează senzori, servomotorul și LED-urile.
    - Pini Analogici: Nu sunt utilizați în această schemă.
2. Microcontroler (Arduino UNO) - U1
    - Pin 5V și GND: Alimentarea microcontrolerului.
    - Pini Digitali: Conectează senzori, servomotorul și LED-urile.
    - Pini Analogici: Nu sunt utilizați în această schemă.
3. Senzori cu Ultrasunete - U2, U3
    - Senzor de Intrare
        - Trigger (Trig) la pinul D7.
        - Echo la pinul D6.
    - Senzor de Ieșire
        - Trigger (Trig) la pinul D5.
        - Echo la pinul D4.

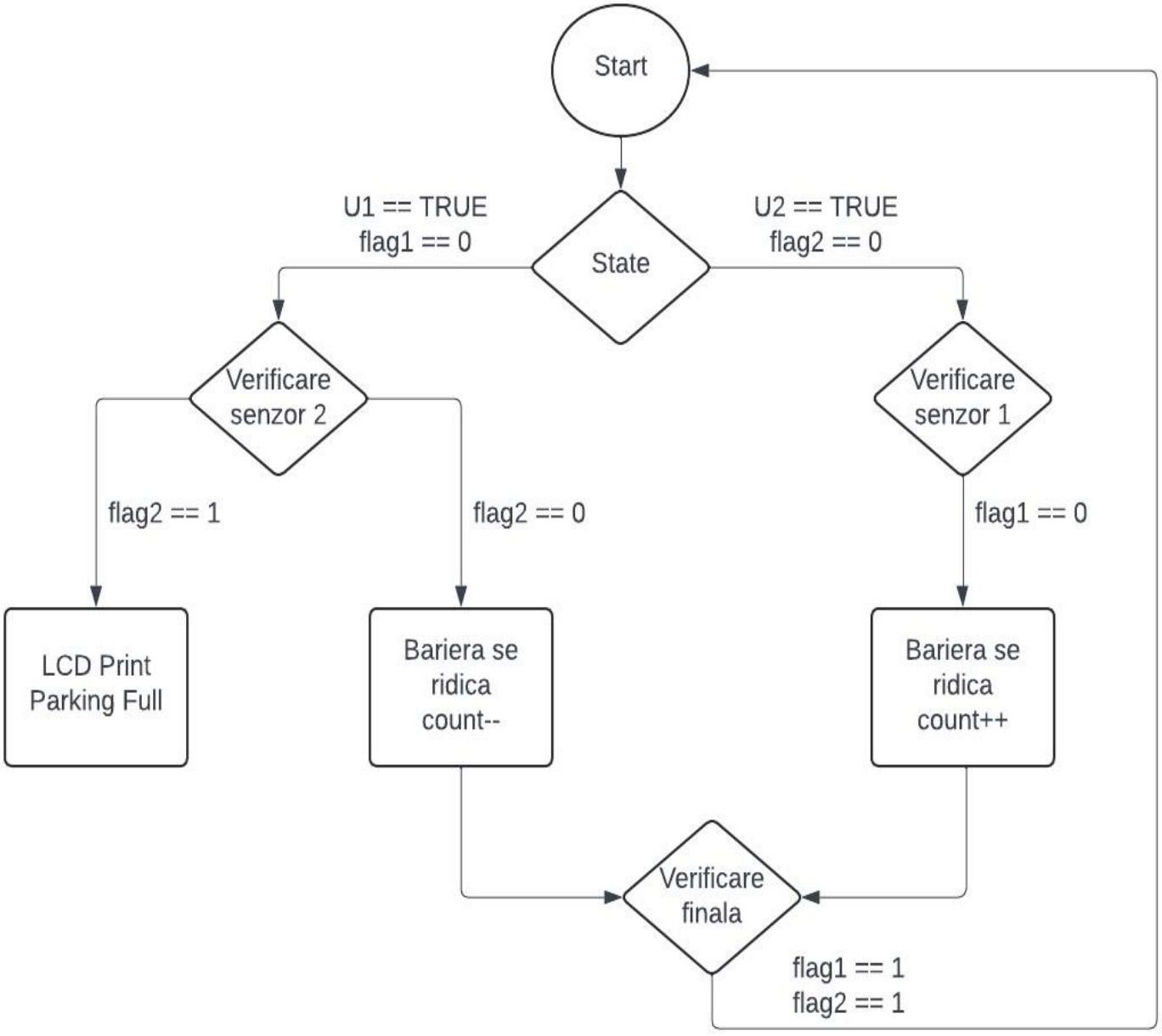
4. Afișaj LCD 16x2 cu Interfață I2C - U4
    - Alimentare 5V și GND.
    - Conexiuni I2C (SDA și SCL) la pinii A4 și A5 ai microcontrolerului.
    - Afișează numărul de locuri libere din parcare.

5. LED-uri de Semnalizare - D1 și D2

- LED Verde (D1): Indică locuri libere, conectat la pinul D12.
- LED Roșu (D2): Indică parcare completă, conectat la pinul D11.
- Rezistențe limitatoare de curent pentru fiecare LED.

6. Rezistențe - R1, R2
    - Limitatoare de curent pentru LED-uri, asigurând protecția acestora.

Diagrama de stare



Codul sursă

Simulare Tinkercard : https://www.tinkercad.com/things/1RLLBX53fVn-copy-of-parcare-cu-contor-pt-si?sharecode=DIBptZ1gmddUzSGalFjxlZW_8Ae5UnmjA5YC9NeZZH4

```
#include <Servo.h>
Servo myservo;  // creeaza obiect servo
#include <Wire.h>
#include <LiquidCrystal_I2C.h>

// Adresele I2C standard pentru diferite dimensiuni ale ecranului LCD
#define I2C_ADDR 0x27 // Adresa I2C a modulului LCD

// Definirea numărului de coloane și rânduri ale LCD-ului
#define LCD_COLS 12
#define LCD_ROWS 2

// Inițializarea obiectului LCD cu adresa I2C specificată mai sus
LiquidCrystal_I2C lcd(I2C_ADDR, LCD_COLS, LCD_ROWS);

//PINS:
//Ultrasonic sensor 1
const int sensor1trigPin = 3;
```

```
const int sensor1echoPin = 4;
//Ultrasonic sensor 2
const int sensor2trigPin = 5;
const int sensor2echoPin = 6;

const long intervalCoborareBariera = 3000; // Intervalul de timp în milisecunde pt coborarea barierei
void setup() {

  Serial.begin(9600);
  //Servo
  myservo.attach(9);  // (pin, min, max)
  //UltraSensor 1
  pinMode(sensor1trigPin, OUTPUT);
  pinMode(sensor1echoPin, INPUT);
  //UltraSensor 2
  pinMode(sensor2trigPin, OUTPUT);
  pinMode(sensor2echoPin, INPUT);

  //LCD
  // initializare ecran lcd
  lcd.init();
  // turn on the backlight
  lcd.backlight();
```

```
  //LED-URI
  pinMode ( 13, OUTPUT);
  pinMode ( 7, OUTPUT);

  //initial bariera este inchisa
  myservo.write(0);
}

void functieDelayMicroseconds(unsigned int microseconds) {
  unsigned long start = micros();
  while (micros() - start < microseconds) {
    //nothing
  }
}


// Sensor ultrasonic 1
long U1()
{
  long duration, distance;
  // Clears the trigPin
  digitalWrite(sensor1trigPin, LOW);

  functieDelayMicroseconds(2);
  //Seteaza trigPin in starea HIGH pt 10 microssecunde
```

```
  digitalWrite(sensor1trigPin, HIGH);
  functieDelayMicroseconds(10);
  digitalWrite(sensor1trigPin, LOW);

  duration = pulseIn(sensor1echoPin, HIGH);

  distance = duration * 0.034 / 2;

  return distance;
}

long U2()
{
  long duration, distance;
  // Clears the trigPin
  digitalWrite(sensor2trigPin, LOW);

  functieDelayMicroseconds(2);
  //Seteaza trigPin in starea HIGH pt 10 microssecunde
  digitalWrite(sensor2trigPin, HIGH);
  functieDelayMicroseconds(10);
  digitalWrite(sensor2trigPin, LOW);

  duration = pulseIn(sensor2echoPin, HIGH);
  // calculam distanta
```

```
  distance = duration * 0.034 / 2;
  return distance;
}
// Variabile
int freeParkingSpots = 4; //consideram ca parcarea are initial 4 locuri libere si totodata 4 locuri in total
unsigned long previousMillisBariera = 0; // Pentru a urmări timpul anterior cand s-a ridicat bariera
unsigned long previousMillisLCD = 0;
int flag1 = 0;
int flag2 = 0;

void loop() {
  unsigned long currentMillis = millis();
  int distance1 = U1(); //distanta fata de senzorul 1 la intrarea in parcare
  int distance2 = U2(); //distanta fata de senzorul 2 la iesirea din parcare
  if(distance1 < 10 && flag1==0){ //daca senzorul de la intrare detecteaza masina
    if(freeParkingSpots > 0){ //daca mai sunt locuri de parcare libere
      flag1 = 1;
      if(flag2==0){
        myservo.write(90);
        previousMillisBariera = currentMillis;
        freeParkingSpots = freeParkingSpots-1;
```

```
    }
  }else{
    lcd.setCursor(0, 0);
    lcd.clear();
    lcd.print(" Parking is full ");
    previousMillisLCD = currentMillis; // Start timer pt LCD
  }
 }

 if(distance2 < 10 && flag2==0){ //daca senzorul de la iesire detecteaza masina
  flag2 = 1;
  if(flag1 == 0){
   myservo.write(90);
   previousMillisBariera = currentMillis;
   if(freeParkingSpots < 4)
    freeParkingSpots = freeParkingSpots + 1;
  }
 }

 if(flag1==1 && flag2==1 && (currentMillis - previousMillisBariera >= intervalCoborareBariera)){ //daca ambii senzori au detectat o masina, inseamna ca ea a trecut de ambii,
                                                                                                  //iar dupa 3 secunde, bariera se inchide
  myservo.write(0);
```

```
    flag1=0;
    flag2=0;
  }

  if (currentMillis - previousMillisLCD >= 1000) {
    previousMillisLCD = currentMillis;
    // write on the top row
    lcd.setCursor(0, 0);
    lcd.print("Available spots:");
    // write on the bottom row
    lcd.setCursor(0, 1);
    lcd.print(freeParkingSpots);
  }

  //leduri
  if(freeParkingSpots > 0){ //daca mai sunt locuri libere, led-ul galben
se aprinde
    digitalWrite (13, LOW);
        digitalWrite (7, HIGH);
  }else{ //daca nu mai sunt locuri libere in parcare, ledul rosu se
aprinde
    digitalWrite (13, HIGH);
        digitalWrite (7, LOW);
  }
}
```

Biblioteci și funcții utilizate în cod

1. Servo.h
    - Descriere: permite plăcii arduino să lucreze cu servomotoare
    - Funcție utilizată:
        - Servo myservo; - Creează un obiect servo pentru a controla un servomotor.
        - myservo.attach(9); - Atașează servomotorul la pinul 9.
        - myservo.write(0); - Setează poziția servomotorului (în grade).
2. Wire.h
    - Descriere: Biblioteca Wire permite comunicația I2C între microcontroler și alte dispozitive I2C.
3. LiquidCrystal_I2C.h
    - Descriere: Biblioteca LiquidCrystal_I2C permite controlul unui ecran LCD prin comunicație I2C.
    - Funcții utilizată: LiquidCrystal_I2C lcd(I2C_ADDR, LCD_COLS, LCD_ROWS); - Inițializează un obiect LCD cu adresa și dimensiunile specificate.
        - lcd.init(); - inițializează ecranul LCD.
        - lcd.backlight(); - pornește iluminarea de fundal a LCD-ului.
        - lcd.setCursor(0, 0); - setează cursorul la poziția specificată pe ecran.
        - lcd.print("text"); - afișează textul specificat pe ecran.
        - lcd.clear(); - șterge conținutul ecranului LCD.

Bibliografie

- https://www.arduino.cc/
- https://howtomechatronics.com/how-it-works/how-servo-motors-work-how-to-control-servos-using-arduino/
- https://howtomechatronics.com/tutorials/arduino/ultrasonic-sensor-hc-sr04/