Senzor de parcare

Autor: Grigoras Theodor-Andrei

Introducere

Proiectul constă în implementarea unui sistem care simulează funcționalitatea senzorilor de parcare de pe mașinile moderne. Sistemul are la bază trei senzori ultrasonici de distanță care scaneaza trei zone diferite, iar cand un obiect se află la o distanță seminficativ de mică, utilizatorul va fi anunțat.

Descriere generală

Senzorul Ultrasonic de distanță HC-SR04 este un sensor cu un timp de răspuns foarte mic, dar în cazul in care obiectul se află la o distanță foarte mică, rezultatele obținute sunt eronate. În cadrul proiectului, am folosit trei astfel de senzori, conectivitatea cu placa Arduino făcându-se cu ajutorul unui Breadboard. Senzorul este format din 4 pini: VCC, GND, Echo si Trigger. Pentru a funcționa, Trigger Pin-ul senzorului are nevoie de un implus de cel puțin 10µS, moment în care senzorul va trimite 8 impulsuri de 40KHz care formează “unda ultrasonică”. În momentul în care unda intâlnește un obiect în calea ei, se reflectă, iar această unda este receptată de catre Pin-ul ECHO, dupa o durată de timp, care va fi transformată în distanță.

Matricea de LED-uri 8×8, vine impreuna cu un modul MAX7291, pentru a minimiza numarul de pini folosiți si pentru a putea folosi o bibliotecă adiționala Arduino (LedControl.h). Modulul foloseste 5 pini: VCC, GND, DIN, CS si CLK care sunt legați de placa Ardunio cu ajutorul unui Breadboard.

Schema bloc

Hardware Design

Componente utilizate:

- Arduino Uno
- Breadboard
- 3x Ultrasonic Sensor HC-SR04
- 8×8 LED Matrix (+ MAX7291 Module)

- Active Buzzer

Schema electrica

Software Design

Am utilizat Arduino IDE pentru partea de software și o librărie externă:

- LedControl.h (pentru a putea controla matricea de LED-uri intr-un mod usor)

Funcțiile principale sunt **setup()** si **loop()**, la care am adăugat doua functii auxiliare

- SonarSensor
- printByte

SonarSensor este o funcție care calculează distanta pana cel mai apropiat obiect pentru un senzor anume. **printByte** este o funcție care afișează o matrice de 8×8 biți (afișarea se face linie cu linie).

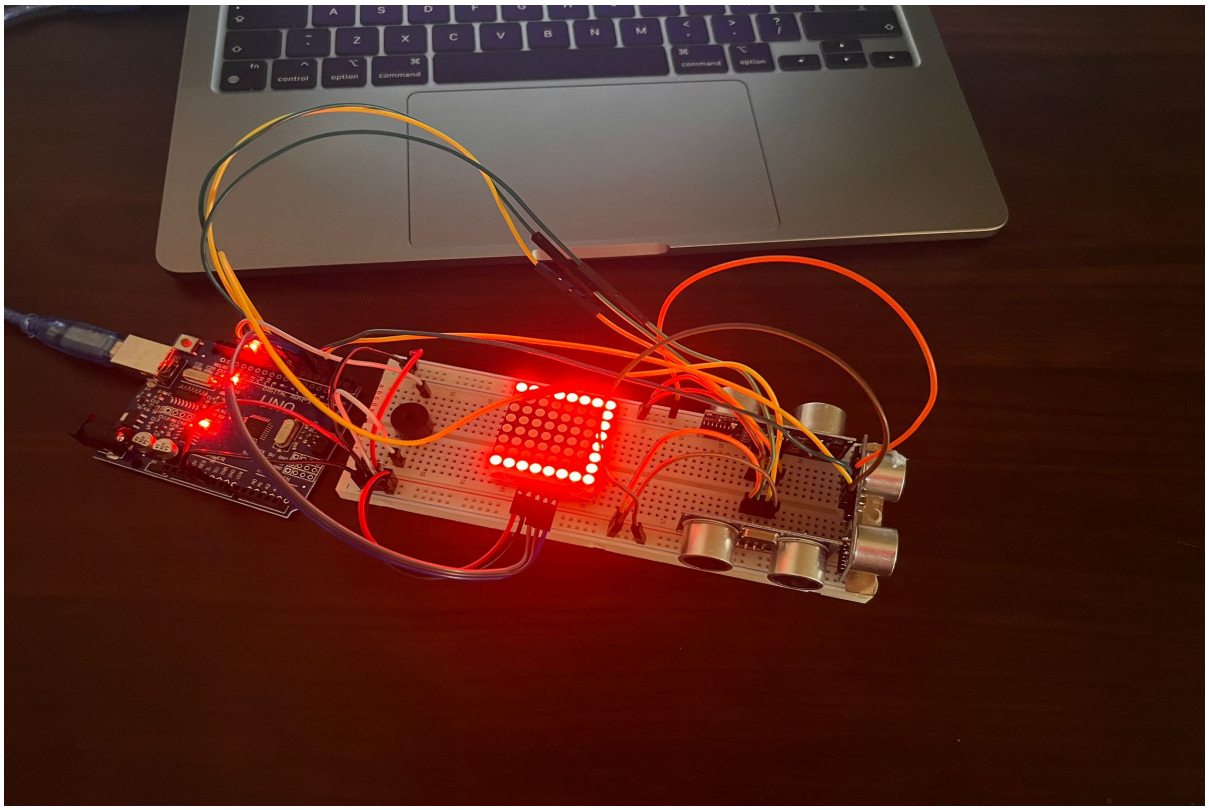
```
#include <LedControl.h>

const int echoPinRight = 8;
const int trigPinRight = 9;

const int echoPinLeft = 10;
const int trigPinLeft = 11;

const int echoPinBack = 2;
const int trigPinBack = 3;

const int BUZZER = 12;

const int DIN =  7;
const int CS = 6;
const int CLK = 5;

LedControl lc = LedControl(DIN, CLK, CS, 0);

long duration;
int distance;

int distanceRight;
int distanceLeft;
int distanceBack;

int safetyDistance = 3;
```

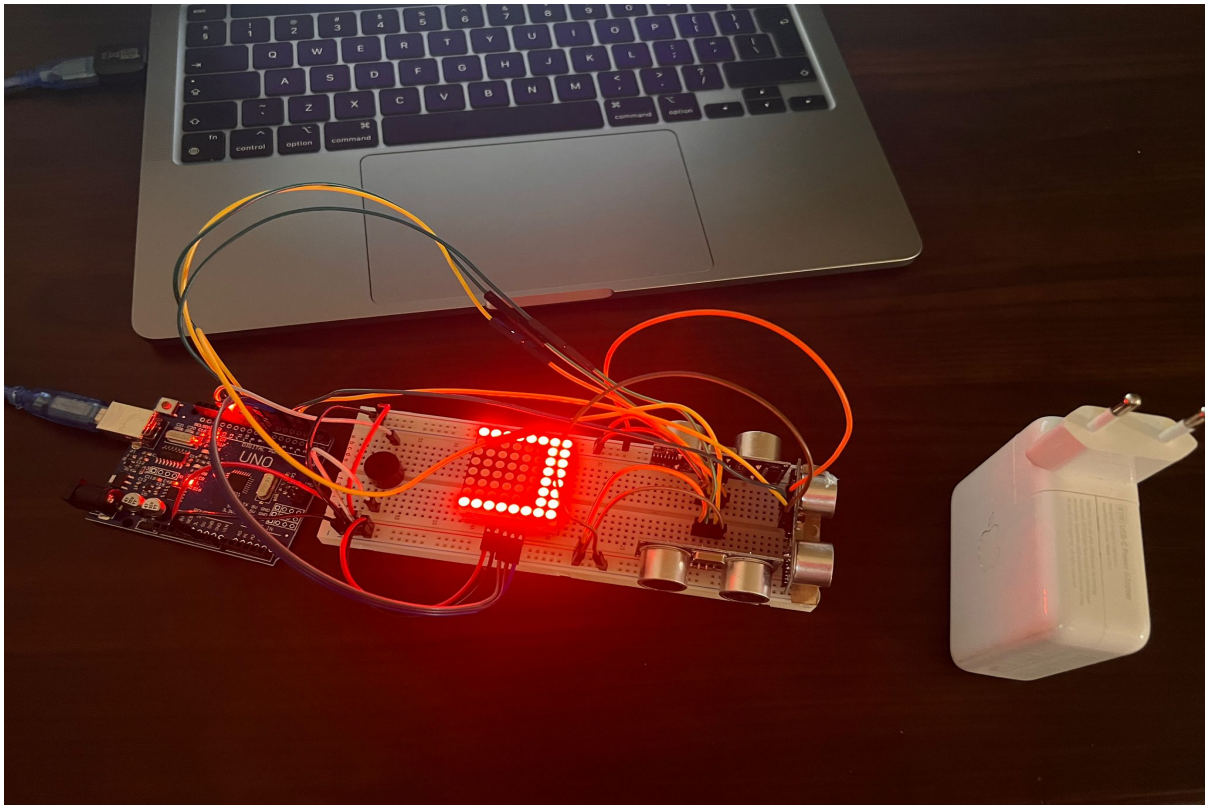
```
void setup() {
  // put your setup code here, to run once:
  pinMode(echoPinRight, INPUT);
  pinMode(trigPinRight, OUTPUT);

  pinMode(echoPinLeft, INPUT);
  pinMode(trigPinLeft, OUTPUT);

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

  pinMode(BUZZER, OUTPUT);

  lc.shutdown(0, false);
  lc.setIntensity(0, 1);
  lc.clearDisplay(0);

  Serial.begin(9600);
}

void loop() {
  // put your main code here, to run repeatedly:
  byte initial[8] = {0b11111111, 0b10000001, 0b10000001, 0b10000001,
0b10000001, 0b10000001, 0b10000001, 0b10000001};

  // obtain the distance from every sensor
  SonarSensor(trigPinLeft, echoPinLeft);
  distanceLeft = distance;

  SonarSensor(trigPinRight, echoPinRight);
  distanceRight = distance;

  SonarSensor(trigPinBack, echoPinBack);
  distanceBack = distance;

  // check the distance
  if(distanceBack < safetyDistance || distanceRight < safetyDistance ||
distanceLeft < safetyDistance) {
    digitalWrite(BUZZER, HIGH);
  } else {
    digitalWrite(BUZZER, LOW);
  }

  // set distance matrix according to all sensors
  if(distanceBack < 7) {
    initial[1] = initial[0];
  }
  if(distanceBack < safetyDistance) {
    initial[2] = initial[0];
  }
```

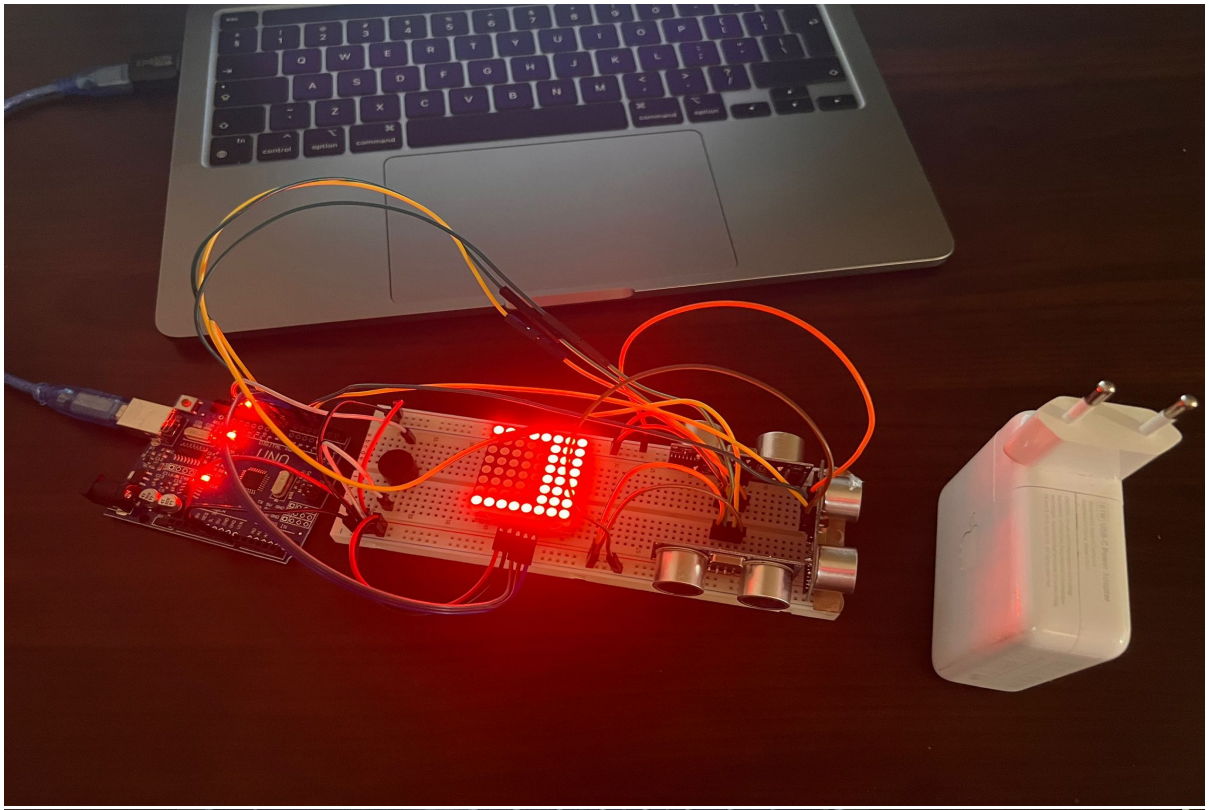
```
  if(distanceLeft < 7) {
    for(int i = 0; i < 8; i++) {
      initial[i] = initial[i] | 0b01000000;
    }
  }

  if(distanceLeft < safetyDistance) {
    for(int i = 0; i < 8; i++) {
      initial[i] = initial[i] | 0b00100000;
    }
  }

  if(distanceRight < 7) {
    for(int i = 0; i < 8; i++) {
      initial[i] = initial[i] | 0b00000010;
    }
  }

  if(distanceRight < safetyDistance) {
    for(int i = 0; i < 8; i++) {
      initial[i] = initial[i] | 0b00000100;
    }
  }


  // print the matrix on LED Matrix
  printByte(initial);
}

void SonarSensor(int trigPin,int echoPin) {
  // disable the trigger pin
  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);

  // enable the trigger pin for 10 microseconds
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);

  // receive the duration
  duration = pulseIn(echoPin, HIGH);
  distance = duration * 0.034 / 2;
}


void printByte(byte character []) {
  int i = 0;
  for(i = 0; i < 8; i ++) {
    lc.setRow(0, i, character[i]);
  }
```

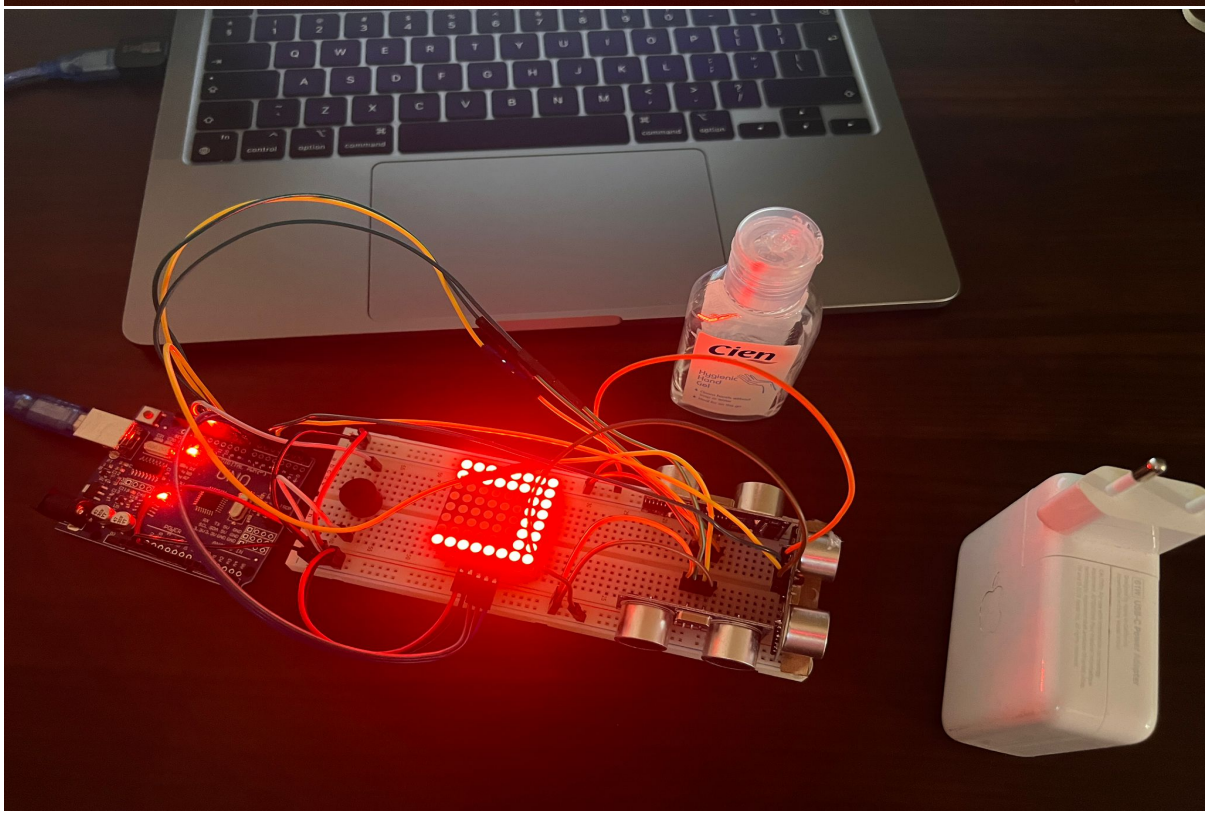
```
}
```

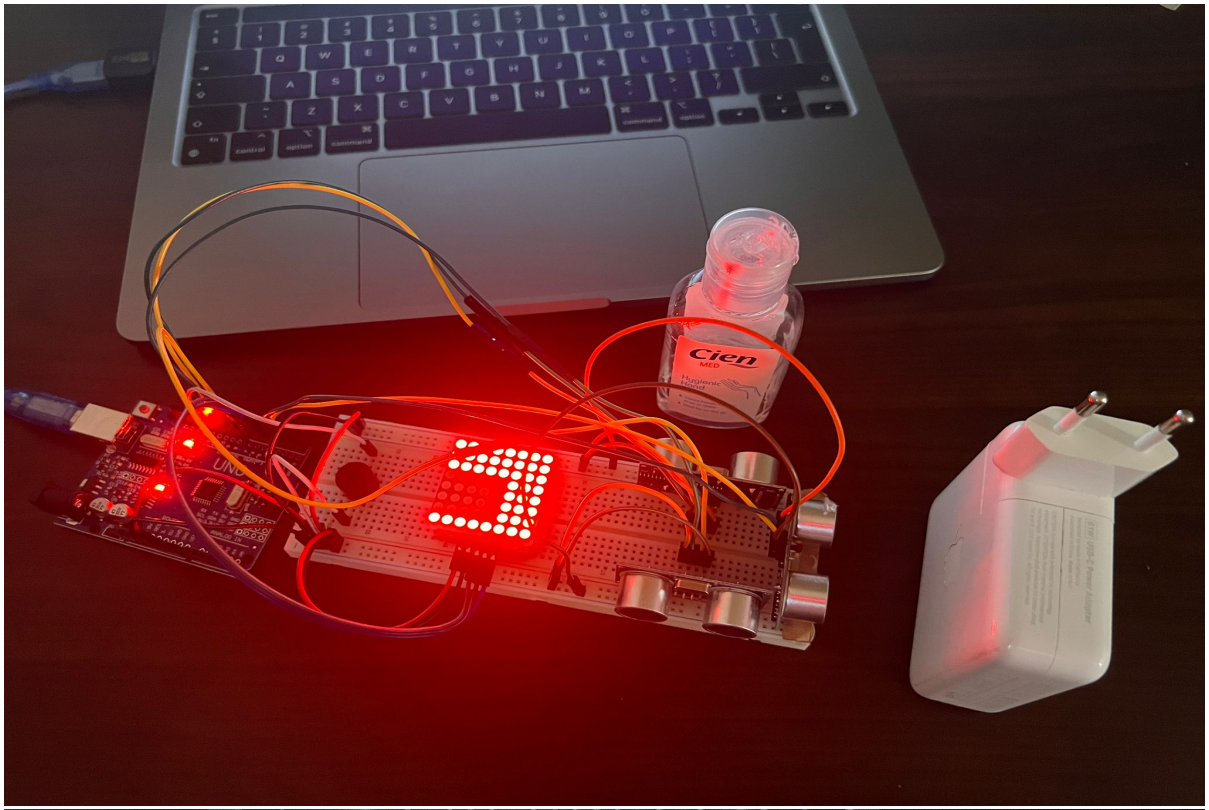
Rezultate Obţinute

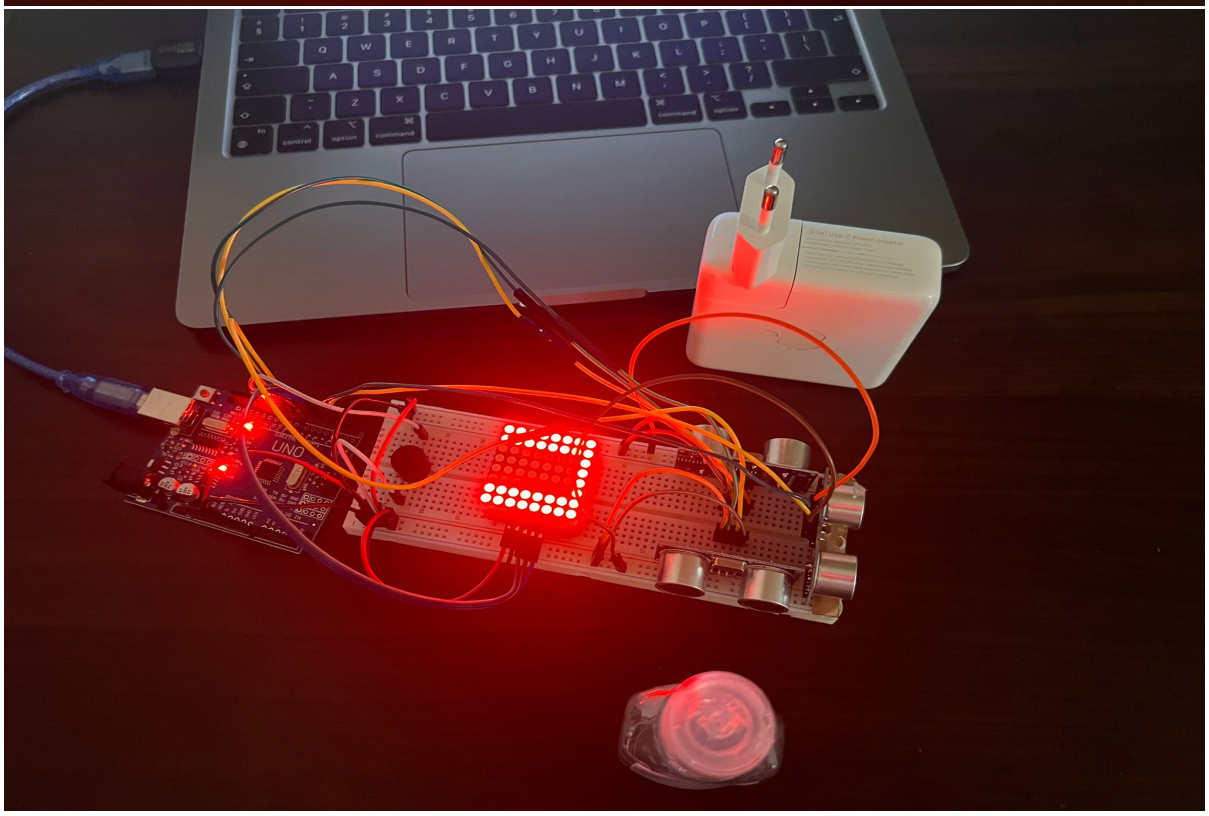







Cien





Concluzii

Fiind primul meu proiect de acest tip, am învățat lucruri noi si utile printre care și lipirea folosind Letcon.

O problemă prezentă in proiect sunt senzorii, fiind niște senzori basic, in momentul in care distanța

față de obiect este mai mică de 2cm, senzorul începe sa ofere rezultate eronate.

Proiectul poate fi folositor în cazul în care vrem să construim o mașinuța electrică si dorim ca această mașină sa dispună de un sistem ajutător pentru parcare. }

Download

Arhiva Proiectului

Jurnal

- **29 aprilie 2022**: crearea paginii de wiki, alegerea temei proiectului
- **22 mai 2022**: introducere, componente utilizate
- **23 mai 2022**: finalizare proiect(parte software, parte hardware, descriere generală, schemă bloc, schemă electrică)

Bibliografie/Resurse

- HC-SR04 Ultrasonic Sensor
- 8x8 LED Matrix
- 8x8 LED Matrix generator

Export to PDF