

---

# Parcare inteligenta

**Proiect realizat la Codesign Hardware-Software**

Studenta: **Nica Andreea Luyza**

Timisoara  
Decembrie, 2023

# Cuprins

1. Introducere.....	3
2. Analiza domeniului.....	4
3. Tehnologii folosite	
3.1. Descrierea plăcii Arduino UNO R3.....	5
3.2. Arhitectura sistemului.....	8
3.3. Descrierea modulelor.....	9
4. Implementarea funcționalității.....	14
5. Concluzii.....	18
6. Bibliografie.....	19

## **1.Introducere**

Proiectul presupune construirea unei parcări cu bariere. Parcarea are un număr fix de 2 locuri disponibile. Dacă cele două locuri sunt ocupate, un LED roșu se va aprinde, iar un afișaj LCD va arăta mesajul "Parcarea este ocupată". În cazul în care mai sunt locuri disponibile (adică numărul de locuri disponibile este mai mare decât 0 și mai mic sau egal cu 2), un LED verde se va aprinde, iar pe afișajul LCD se va afișa numărul de locuri libere.

Intrarea în parcarea este controlată de doi senzori: un senzor infraroșu și un senzor ultrasunete. Senzorul infraroșu detectează când o mașină dorește să intre, în timp ce senzorul ultrasunete măsoară înălțimea mașinii. Dacă există locuri disponibile și înălțimea mașinii este în limita admisă, bariera va fi ridicată cu ajutorul unui servomotor.

Când o mașină ajunge într-un loc de parcare, un alt senzor infraroșu o va detecta și va începe să numere timpul petrecut de mașină în acel loc.

Pentru a părasi parcarea, șoferul trebuie să plătească taxa corespunzătoare timpului petrecut în parcarea. Suma trebuie introdusă în Serial Monitor. Doar după ce taxa (calculată ca fiind dublul numărului de secunde petrecute în parcarea respectivă) a fost plătită, bariera de ieșire va fi ridicată.

## 2. Analiza domeniului

Parcarile cu bariere au devenit o prezență tot mai obișnuită în toate orașele, datorită capacitatea lor de a monitoriza și gestiona eficient numărul de locuri de parcare disponibile. Există diverse sisteme de parcare, unele rezervate exclusiv angajaților sau studentilor, cu acces controlat prin intermediul legitimatiilor, altele cu tarifare pe întreaga perioadă de staționare, și chiar parcări cu ofertă de gratuitate în prima oră de parcare.

În continuare, voi analiza două exemple care se asemănă cel mai mult cu proiectul pe care l-am dezvoltat: parcarea din Piața 700 și parcarea de la Isho.

### 1. Parcarea din piața 700

Aceasta parcare este dotată cu două bariere distincte, una pentru intrare și cealaltă pentru ieșire. În momentul în care un vehicul intră în parcare, se emite un bilet care servește la monitorizarea duratei de staționare. Pentru a părăsi parcarea, utilizatorul se îndreaptă către un automat dedicat, unde introduce atât biletul cât și suma corespunzătoare tarifului. Ulterior, se deplasează către bariera de ieșire, unde există un alt automat destinat introducerii biletului. Dacă plata a fost efectuată cu succes, bara de ieșire se ridică, permitând astfel părăsirea parcării.

### 2. Parcarea de la Isho

Acesta este o parcare închisă, care reprezintă un tip special de parcare, cu un design impresionant pe mai multe nivele, în total 9. O caracteristică distinctivă a acesteia constă în implementarea unor senzori avansați care detectează automat numărul de înmatriculare al vehiculului. Această tehnologie elimină necesitatea utilizării biletelor traditionale, simplificând astfel întregul proces. Angajații de la Isho beneficiază de acces gratuit la această parcare, în timp ce restul populației are tarife de plată.

Caracteristici	Parcarea dezvoltată de mine	Parcarea din Piața 700	Parcarea Isho
Restrictii înălțime	DA	NU	DA
Bariera intrare	DA	DA	DA
Bariera ieșire	DA	DA	DA
Parcare cu plata	DA	DA	DA
Parcare gratis angajați	NU	NU	DA

### 3.Tehnologii folosite

#### 3.1. Descrierea plăcii Arduino UNO R3

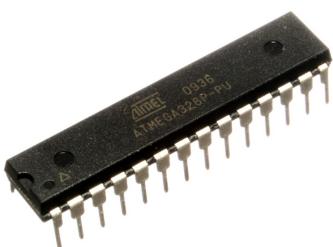


Arduino Uno R3 este o placă de dezvoltare open-source bazată pe un microcontroler ATmega328P.

Aceasta este una dintre cele mai populare plăci Arduino, datorită dimensiunii sale compacte și a versatilității sale.

Arduino Uno R3 are 14 pini digitali de intrare / ieșire (din care 6 pot fi utilizate ca ieșiri PWM), 6 intrări analogice, un oscilator de 16 MHz, un port USB, o conexiune ICSP și un buton de reset.

Aceasta poate fi programată utilizând mediul de dezvoltare Arduino și limbajul de programare C/C++.

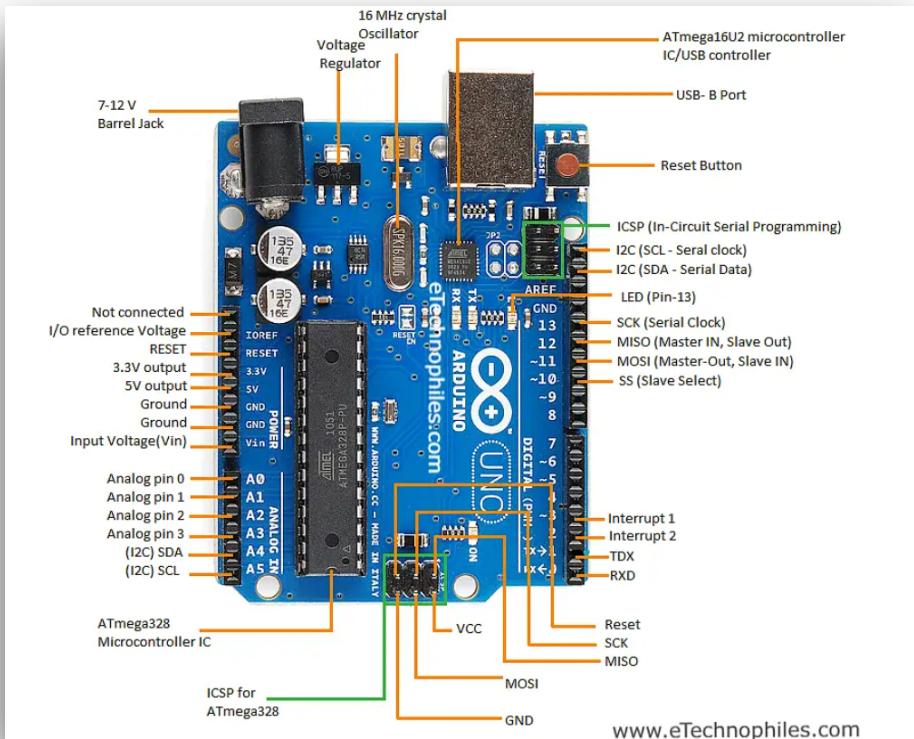


Microcontrolerul ATmega328P are o arhitectură de procesor RISC de 8 biți, care funcționează la o frecvență de până la 20 MHz și este capabil să execute instrucțiuni cu o singură clătinare a oscilatorului. Aceasta include 32KB de memorie flash, 2KB de memorie SRAM și 1KB de memorie EEPROM.

#### Specificatii:

- Microcontroler: ATmega328
- Tensiune de lucru: 5V
- Tensiune de intrare (recomandat): 7-12V
- Tensiune de intrare (limita): 6-20V
- Pini digitali: 14 (6 PWM output)
- Pini analogici: 6
- Current per pin I/O: 40 mA
- Current 3.3V: 50 mA
- Memorie Flash: 32 KB (ATmega328)  
0.5 KB pentru bootloader
- SRAM: 2 KB (ATmega328)
- EEPROM: 1 KB (ATmega328)
- Clock Speed: 16 MHz

Descrierea pinilor:



Există 14 pini digitali de intrare / ieșire (I/O sau input/output).

Aceștia operează la o tensiune de 5 volți și pot fi controlați cu una din funcțiile

- **0 (serial) RX** – pin serial, utilizat în special pentru recepția (intrare – Rx) datelor seriale asincrone
- **1 (serial) TX** – pin serial, utilizat pentru trimitera datelor asincrone (ieșire – Tx)
- **2 (întrerupere externă)** - acest pin poate fi configurat pentru a declanșa o întrerupere la o valoare mică, un front crescător sau descrescător, sau o schimbare în valoare.
- **3 (întrerupere externă)** - identic cu pinul 2.
- **4 (I/O)** - pin standard intrare/iesire
- **5 (PWM)** - poate furniza control de ieșire pe 8-bit pentru controlul PWM.
- **6 (PWM)**
- **7 (I/O)** - pin standard intrare/iesire
- **8 (I/O)** - pin standard intrare/iesire
- **9 (PWM)**
- **10 (PWM + SPI)** - suportă comunicare prin interfață serială (Serial Peripheral Interface). SPI-ul are patru semnale logice specifice iar acest pin se folosește pentru SS – Slave Select.
- **11 (PWM + SPI)** - suportă SPI, iar acest pin se folosește pentru MOSI/SIMO – Master Output, Slave Input

- **12 (SPI)** - suportă SPI, iar acest pin se folosește pentru MISO/SOMI – Master Input, Slave Output
- **13 (LED + SPI)** - suportă SPI, iar acest pin se folosește pentru SCK/SCLK – Ceas serial
- **14 (GND)** - împământare
- **15 (AREF)** - Analog REFference pin – este utilizat pentru tensiunea de referință pentru intrările analogice.
- **16 (SDA)** - comunicare I2S
- **17 (SCL)** - comunicare I2

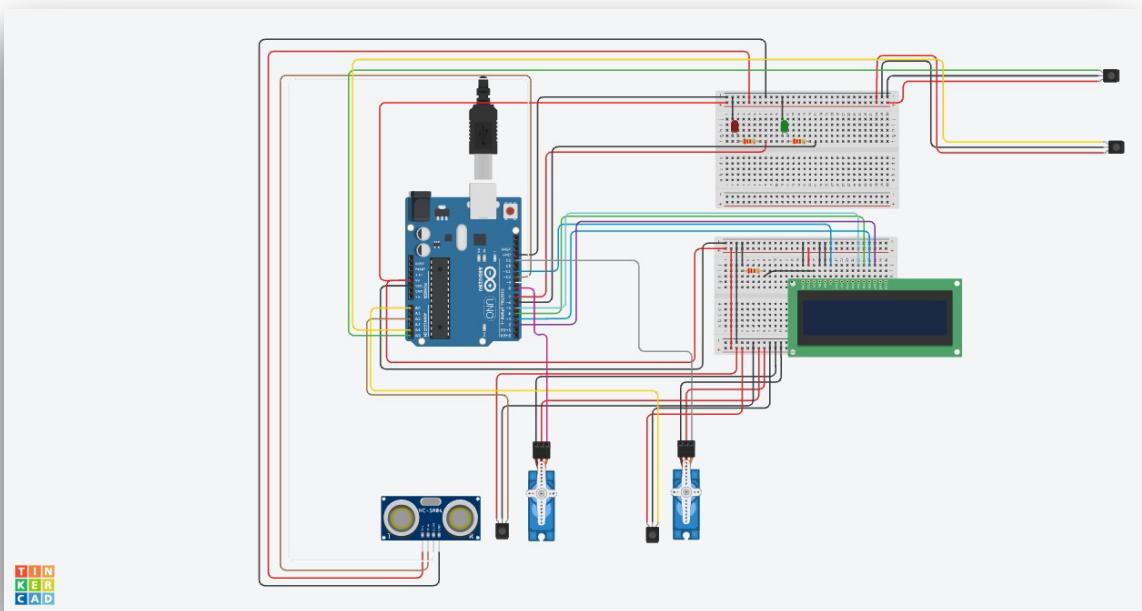
În partea de jos există o serie de 6 pini pentru semnal analogic, numerotați de la A0 la A5. Fiecare din ei poate furniza o rezoluție de 10 biți (adică maxim 1024 de valori diferite). În mod implicit se măsoară de la 0 la 5 volți, deși este posibil să se schimbe limita superioară a intervalului lor folosind pinul 15 AREF și funcția analogReference(). De asemenea, și aici anumiți pini au funcții suplimentare descrise mai jos:

- A1 standard analog pin
- A2 standard analog pin
- A3 standard analog pin
- A4 (SDA) suportă comunicarea prin 2 fire (I2C (I-two-C) sau TWI (Two wire interface)). Acest pin este folosit pentru SDA (Serial Data) la TWI.
- A5 (SCL) identic cu pinul 4, doar că acest pin este folosit pentru SCL (Serial Clock) la TWI.

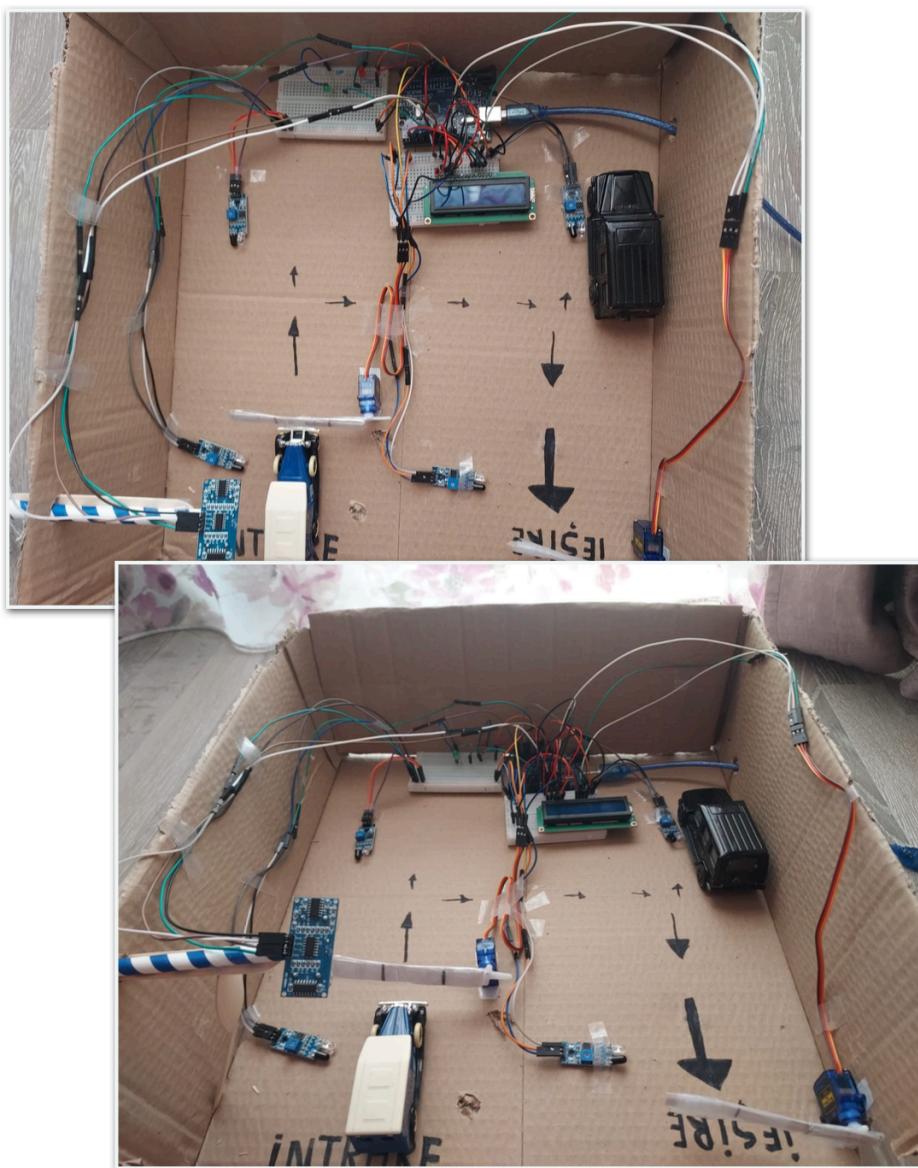
Lângă pinii analogici arătați mai există o secțiune de pini notată POWER. Acesteia sunt (începând de lângă pinul analog A0) :

- **1 Vin** – intrarea pentru tensiune din sursă externă (input Voltage)
- **2 GND** - negativul pentru tensiune din sursă externă (ground Voltage)
- **3 GND** - negativ. Se folosește pentru piesele și componentele montate la arduino ca și masă/împământare/negativ.
- **4 5V** - ieșire pentru piesele și componentele montate la arduino.
- **5 3,3V** - ieșire pentru piesele și senzorii care se alimentează la această tensiune.
- **6 RESET** - se poate seta acest pin pe LOW pentru a reseta controlerul de la Arduino.
- **7 5VREF** - este folosit de unele shield-uri ca referință pentru a se comuta automat la tensiunea furnizată de placă Arduino.
- **8 pin neconectat**, este rezervat pentru utilizări ulterioare

### 3.2. Arhitectura sistemului

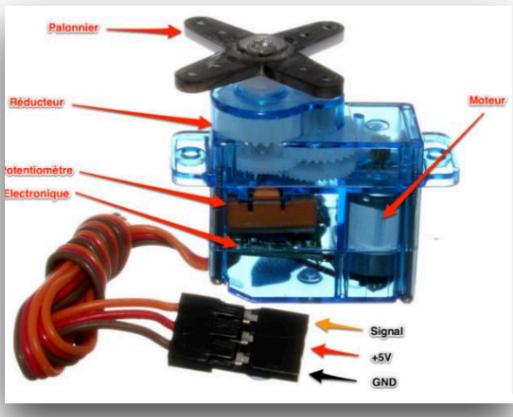


Schema fizica a sistemului



### 3.3. Descrierea modulelor

## Servomotor



SG90 este un motor de tip servo, ceea ce înseamnă că este capabil să se rotească la un anumit unghi specific, determinat de semnalul de control trimis către acesta. Acest semnal este de obicei generat de un microcontroler sau un alt dispozitiv electronic.

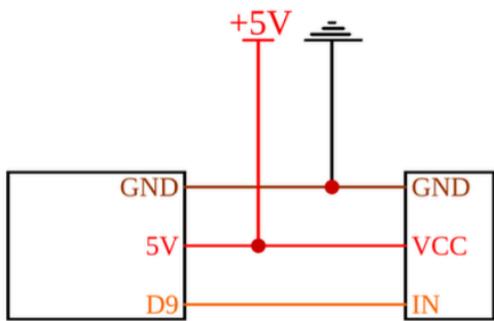
Descrierea pinilor :

Servo motor	Connection
Power (red)	5 V power supply
Ground (black or brown)	Power supply ground and Arduino GND
Signal (yellow, orange or white)	Pin 9 Arduino

Specificatii :

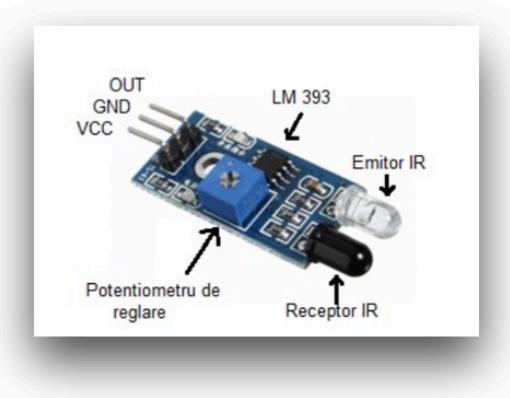
1. Tensiunea de lucru: între 4,8 și 6 volți.
2. Curentul de lucru: între 150mA și 500mA.
3. Viteza de rotație: 0,12 secunde pentru 60 de grade (în condiții ideale).
4. Unghiul de rotație: între 0 și 180 de grade.
5. Cuplul maxim: 1,8 kg-cm (la 4,8 volți).
6. Dimensiuni: 23 mm x 12 mm x 29 mm.
7. Greutate: aproximativ 9 grame.
8. Tipul de mufă de conectare: mufă tip JR.
9. Temperatura de funcționare: între 0 și 55 de grade Celsius.
10. Precizia poziționării: +/- 3 grade.

## Conecțarea cu Arduino :



Controlul servomotorului se realizează cu ajutorul unui semnal de tip PWM.

## Senzor infraroșu



Un senzor infraroșu este un dispozitiv electronic care poate detecta radiația infraroșie emisă de obiecte sau surse de căldură din mediul înconjurător și poate transforma această informație în semnale electrice pentru a fi prelucrate și utilizate de alte dispozitive.

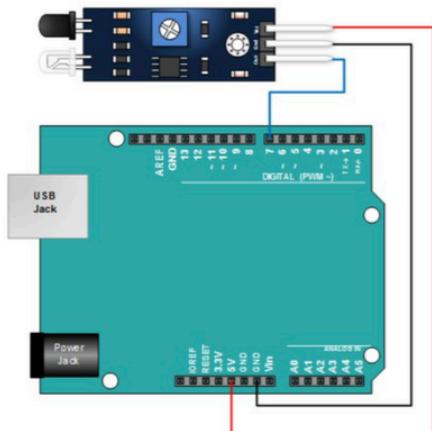
## Descrierea pinilor :

1. Alimentare(VCC) : 3.3-5V, se furnizează tensiunea necesară alimentării senzorului
2. Masa(GND) – asigură o referință comună pentru semnalele de pe senzor
3. OUT – furnizează semnalul de ieșire al senzorului (LOW-dacă nu s-a detectat mișcare, HIGH-dacă s-a detectat)

## Specificatii :

1. Distanță de detecție: 1 ~ 25 mm;
2. Semnal digital sau analogic;
3. Potențiometru pentru ajustare prag de detecție;
4. Current maxim ieșire semnal digital: 15mA;
5. Tensiune de alimentare: 3.3V sau 5V;
6. Lungime de undă radiație: 950nm.

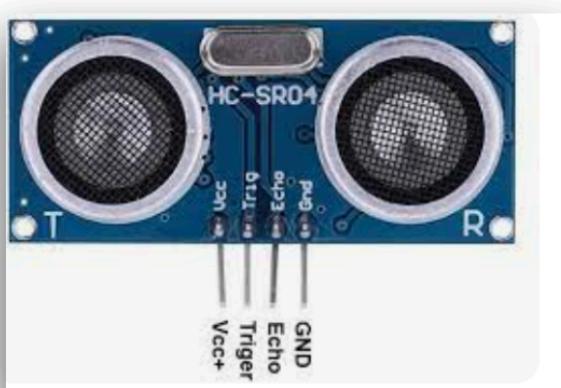
## Conecțarea cu Arduino :



Senzorul de obstacole se bazează pe reflexia radiației IR de către obstacol. Radiația IR este emisă de către un LED și este recepționată de către o fotodiodă. Output-ul senzorului este digital. Distanța de detecție poate fi reglată dintr-un potențiometru.

Modulul conține două led-uri indicate, unul pentru alimentare și celălalt pentru detectarea obstacolului. Prin urmare conectarea se face astfel : pinul VCC al senzorului la pinul 5V al Arduino, pinul GND al senzorului la pinul GND al Arduino, pinul OUT al senzorului la un pin digital al Arduino (în cazul

## Senzor ultrasonic ( HC-SR04)



### Descrierea pinilor:

- VCC (alimentare)
- Trig (declanșator)
- Echo (recepție)
- GND (masă).

Senzorul ultrasunete HC-SR04 este un dispozitiv folosit pentru a măsura distanța între senzor și un obiect folosind unde sonore ultrasunete.

Pentru a măsura distanța, senzorul emite un semnal ultrasunete de înaltă frecvență (aproximativ 40 kHz) folosind emițătorul.

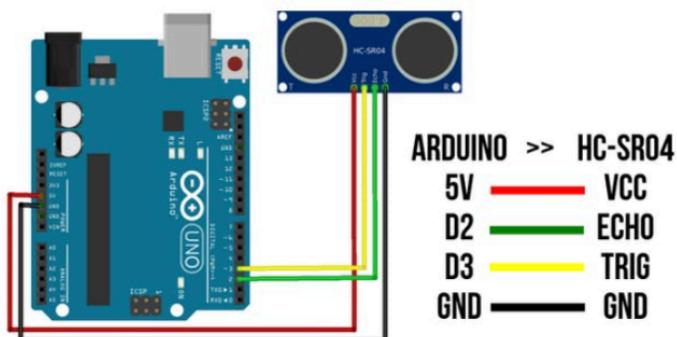
Acet semnal ultrasunete se propagă în aer și atinge un obiect din apropiere. Când semnalul întâlnește obiectul, acesta se reflectă înapoi spre senzor. Senzorul utilizează receptorul pentru a detecta semnalul ultrasunete reflectat.

Timpul necesar pentru ca semnalul să parcurgă distanța dus-întors (emisie și recepție) este măsurat de senzor. Senzorul convertește acest timp într-o valoare de distanță utilizând formula: distanță = (timpul în secunde \* viteza sunetului în aer) / 2.

### Specificatii:

- Tensiune de alimentare: 5V
- Current consumat: 15mA
- Distanță de funcționare: 2cm - 4m
- Unghi de măsurare: 15 grade
- Eroare de doar 3mm
- Durată semnal input: 10us
- Dimensiuni: 45mm x 20mm x 15mm

## Conecțarea cu Arduino:



Cu ajutorul pinilor Trig și Echo, senzorul este controlat de placă Arduino. Pinul Trig este utilizat pentru a declanșa emisia ultrasunetelor, iar pinul Echo este utilizat pentru a măsura timpul în care semnalul este reflectat.

Arduino poate calcula apoi distanța utilizând formula menționată mai sus.

## LCD



LCD 16x2 (Liquid Crystal Display 16 caractere pe 2 linii) este un afișaj alfanumeric cu cristale lichide format dintr-un ecran de 16 caractere și 2 linii de afișare. Fiecare caracter este afișat pe ecran sub forma unor cifre sau litere și poate fi controlat prin intermediul unor pini de control.

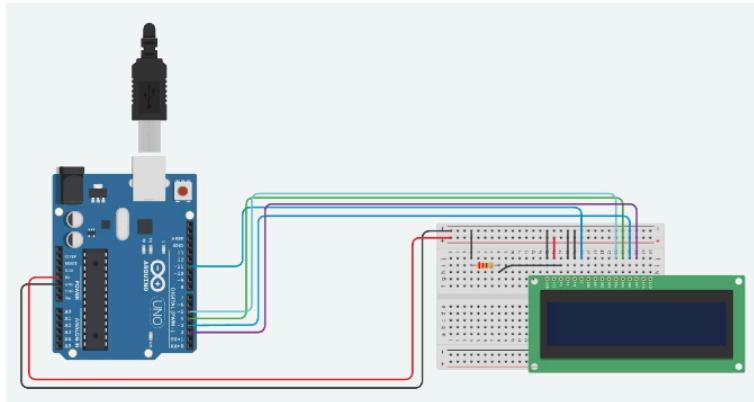
## Descrierea pinilor :

Pin No	Symbol	Level	Description
1	V <sub>SS</sub>	0 V	Ground
2	V <sub>DD</sub>	5.0V	Supply voltage for logic
3	V <sub>O</sub>	variable	Operating voltage for LCD
4	RS	H/L	H data/L instruction code
5	R/W	H/L	H/Read (MPU-module) L/write (MPU-module)
6	E	H, H-L	Chip enable signal
7	DB0	H/L	Data bit 0
8	DB1	H/L	Data bit 1
9	DB2	H/L	Data bit 2
10	DB3	H/L	Data bit 3
11	DB4	H/L	Data bit 4
12	DB5	H/L	Data bit 5
13	DB6	H/L	Data bit 6
14	DB 7	H/L	Data bit 7
15	A		Power supply for LED backlight (+)
16	K		Power supply for LED backlight (-)

## Specificatii :

- 1.Material PCB + plastic
- 2.Tip ecran LCD
- 3.Dimensiune ecran 2.6 inch
- 4.Rezoluție 80 x 16
- 5.Tensiune de lucru 4.5 ~ 5.5V
- 6.Curent de lucru 80mA
- 7.Dimensiuni 3,15 in x 1,42 in x 0,71 in (8 cm x 3,6 cm x 1,8 cm)
- 8.Greutate 34 g

## Conecțarea cu Arduino:



Conecțarea cu Arduino se face prin intermediul a patru pini de semnalizare: VCC (alimentare), GND (masă), pinul RS (Selectare regiszru), pinul E (Enable) și pin din Arduino-linii de date (se pot utiliza fie pinii digitali, fie cei analogici cu, conditia ca se preciza în program că acești pini sunt utilizati ca și digitali prin funcția pinMode()).

## 4. Implementarea funcționalității

```
#include <Servo.h>
#include <LiquidCrystal.h>
```

Biblioteci folosite pentru a controla servomotoarele și LCD

```
#define RS 12
#define E 11
#define D4 2
#define D5 3
#define D6 4
#define D7 5
#define MAX_LOCURI_PARCARE 2
#define PARCARE_PLINA 0
```

Define-uri pentru pinii la care este conectat LCD-ul, respectiv pentru numărul maxim de locuri de parcare și numărul de locuri disponibile în cazul în care parcarea este plină.

```
int nrLocuriParcare = MAX_LOCURI_PARCARE;

Servo barieraIntrare, barieraIesire;

LiquidCrystal lcd(RS, E, D4, D5, D6, D7);

const int echoPin = 10 ;
const int triggerPin = 9;

const int ledVerde = 6;
const int ledRosu = 7;

int senzorIntrare = A0;
int senzorIesire = A1;
int senzorLoc1 = A5;
int senzorLoc2 = A4;

unsigned long startStationare1 = 0, startStationare2=0;
unsigned int suma1 = 0, suma2=0;
int locOcupat1 = false, locOcupat2 = false
```

Definirea variabilelor și a obiectelor utilizate în gestionarea sistemului.

```
void ridicaBariera(Servo servo){
    servo.write(0);
    delay(3000);
    servo.write(90);
}
```

Funcție responsabilă de controlul mișcării barierei de intrare/iesire pentru o anumită perioadă de timp(3 secunde), pentru a permite trecerea unui vehicul în parcare sau ieșirea din parcare.

```
long citesteDistanțaSenzor() {
    long duration, distance;

    digitalWrite(triggerPin, LOW);
    delayMicroseconds(2);
    digitalWrite(triggerPin, HIGH);
    delayMicroseconds(10);
    digitalWrite(triggerPin, LOW);

    duration = pulseIn(echoPin, HIGH);
    distance = (duration / 2) / 29.1;

    return distance;
}
```

Funcția utilizează un senzor de distanță ultrasunete pentru a măsura distanța până la un obiect și returnează această distanță în centimetri. Funcția inițiază semnalul de trigger către senzor, măsoară timpul de întoarcere a semnalului de ecou și calculează

```
void gestioneazaIntrare() {
    long distanta= citesteDistantaSenzor();
    int stareIntrare = digitalRead(senzorIntrare);

    if (stareIntrare == LOW && nrLocuriParcare >
PARCARE_PLINA && nrLocuriParcare <=
MAX_LOCURI_PARCARE && distanta<8) {
        ridicaBariera(barieraIntrare);
        nrLocuriParcare--;
    }
}
```

distanța pe baza acestui timp.

Funcția verifică starea senzorului de intrare și distanța măsurată de senzorul ultrasunete. Dacă senzorul de intrare este în starea LOW (un vehicul se apropie de intrare), numărul de locuri de parcare nu este plin și distanța măsurată este mai mică de 8 cm (înălțimea vehiculului), atunci funcția ridică bariera de intrare și decrementează numărul de locuri de parcare.

```
void gestioneazaIesire() {
    int stareIesire = digitalRead(senzorIesire);

    if (stareIesire == LOW && nrLocuriParcare <
MAX_LOCURI_PARCARE) {
        ridicaBariera(barieraIesire);
        nrLocuriParcare++;
    }
}
```

Functia verifică starea senzorului de ieșire, apoi ridică bariera de ieșire și incrementează numărul de locuri de parcare.

```
void afiseazaNumarLocuri() {
    lcd.setCursor(8, 0);
    lcd.print("  ");
    lcd.setCursor(8, 0);
    lcd.print(nrLocuriParcare);
}
```

Funcția actualizează afișajul LCD pentru a afișa numărul de locuri de parcare disponibile.

```
void gestioneazaLeduri(){
    if (nrLocuriParcare == 0) {
        digitalWrite(ledVerde, LOW);
        digitalWrite(ledRosu, HIGH);
    } else {
        digitalWrite(ledVerde, HIGH);
        digitalWrite(ledRosu, LOW);
    }
}
```

Funcția gestionează starea a două LED-uri (verde și roșu) în funcție de numărul de locuri de parcare disponibile. Dacă nu mai sunt locuri de parcare disponibile, atunci se aprinde LED-ul roșu și se stinge LED-ul verde. În caz contrar, se aprinde LED-ul

```
void afisareSumaPlata(int sumaAfisare){
    Serial.print("Aveti de plata suma de: ");
    Serial.print(sumaAfisare);
    Serial.println(" lei");
    Serial.println("\n");
    Serial.println("Va rugam introduceti suma de plata!");
}
```

Functia afiseaza mesajul de plata al sumei date ca parametru, in Serial Monitor.

```

void calculeazaSuma(int suma){
afisareSumaPlata(suma);

while (Serial.available() == 0) {
}

int numar = Serial.parseInt();
if (suma == numar) {
    Serial.print("Suma a fost achitata, va multumim si
o zi frumoasa ");
}

while (numar < suma) {
    suma = suma - numar;
    Serial.print("Mai aveti de introdus: ");
    Serial.print(suma);
    Serial.println(" lei");
    numar = Serial.parseInt();
}

if (numar > suma) {
    Serial.print("Va rugam ridicati restul: ");
    Serial.print(numar - suma);
    Serial.println(" lei");
}
suma = 0;
Serial.print("Suma a fost achitata, va multumim si
o zi frumoasa! ");
Serial.println('\n');
}

```

Funcția afișează suma de plată și solicită utilizatorului să introducă suma corespunzătoare pe portul serial. Apoi, funcția verifică suma introdusă de utilizator și afișează mesaje corespunzătoare în funcție de 3 scenarii diferite (utilizatorul a introdus suma exactă/mai mare de introdus pentru a achita suma/a introdus o suma mai mare și primește rest).

```

void gestioneazaLocParcare(int senzorLoc, int
&locOcupat, unsigned long &startStationare, unsigned int
&suma) {
int numar = 0;
int stareLocParcare = digitalRead(senzorLoc);
if (stareLocParcare == LOW && !locOcupat) {
    locOcupat = true;
    startStationare = millis();
    Serial.println("Masina a intrat in locul de parcare");
}
else if (stareLocParcare == HIGH && locOcupat) {
    locOcupat = false;
    unsigned int suma =
calculTimpParcare(startStationare);
    if(suma > 0){
        calculeazaSuma(suma);
    }
}
}

```

Funcția monitorizează un senzor care detectează starea unui loc de parcare. Funcția urmărește dacă locul de parcare este ocupat sau nu, înregistrează timpul de începere a staționării și afișează starea locului de parcare pe portul serial. Dacă un vehicul intră în locul de parcare (starea LOW și locul nu este deja ocupat), atunci locul de parcare este marcat ca ocupat și se înregistrează timpul de începere al staționării. Dacă un vehicul părăsește locul de parcare (starea HIGH și locul de parcare este deja ocupat), atunci locul de parcare este marcat ca disponibil, se calculează timpul petrecut în locul de parcare și se apelează funcția pentru calculul și gestionarea plății pentru timpul de parcare.

```

calculTimpParcare(unsigned long start){
    unsigned long stopStationare = millis();
    unsigned long timpParcare = stopStationare - start;

    Serial.print("Masina a stat in parcare pentru: ");
    Serial.print(timpParcare / 1000);
    Serial.println(" secunde");

    return timpParcare / 1000 * 2;
}

```

Funcția primește ca parametru timpul de începere al staționării, calculează timpul petrecut în parcare pe baza timpului dat ca parametru și afișează durata staționării pe portul serial în secunde. Funcția returnează suma de plată corespunzătoare timpul staționării (2 lei /sec).

```

void setup() {
    Serial.begin(9600);
    barieraIntrare.attach(8);
    barieraLesire.attach(13);

    pinMode(echoPin, INPUT);
    pinMode(triggerPin, OUTPUT);

    pinMode(ledVerde, OUTPUT);
    pinMode(ledRosu, OUTPUT);

    lcd.begin(16, 2);
    lcd.print("Locuri: ");
    lcd.setCursor(8, 0);
    lcd.print(MAX_LOCURI_PARCARE);

    digitalWrite(ledVerde, HIGH);
    digitalWrite(ledRosu, LOW);

    pinMode(senzorIntrare, INPUT);
    pinMode(senzorLesire, INPUT);

    pinMode(senzorLoc1, INPUT);
    pinMode(senzorLoc2, INPUT);

    barieraIntrare.write(90);
    barieraLesire.write(90);
}

```

Funcția se ocupa de configurarea inițială a perifericelor și componentelor utilizate în sistem.

```

void loop() {
    gestioneazaLeduri();
    gestioneazaIntrare();
    gestioneazaLesire();
    afiseazaNumarLocuri();

    if(nrLocuriParcare>=PARCARE_PLINA &&
    nrLocuriParcare<MAX_LOCURI_PARCARE){
        gestioneazaLocParcare(senzorLoc1, locOcupat1,
        startStationare1, suma1);
        gestioneazaLocParcare(senzorLoc2, locOcupat2,
        startStationare2, suma2);
    }
}

```

Functia coordonează funcționalitățile sistemului de parcare.

## 5. Concluzii

Proiectul propus oferă o soluție eficientă pentru gestionarea unei parcări cu bariere, integrând tehnologii moderne și asigurând o experiență convenabilă pentru utilizatori:

**-Eficiență în Utilizarea Spațiului:** Parcarea cu bariere asigură o utilizare eficientă a spațiului disponibil, permitând gestionarea a două locuri de parcare. Acest aspect contribuie la optimizarea resurselor și la evitarea supraaglomerării.

**-Monitorizare în Timp Real:** Sistemul este dotat cu senzori infraroșu și ultrasunete pentru detectarea vehiculelor și măsurarea înălțimii acestora. Această monitorizare în timp real permite sistemului să reacționeze rapid la sosirea sau plecarea mașinilor, asigurând astfel un control precis al accesului în parcarea respectivă.

**-Indicatori Vizuali pentru Șoferi:** LED-urile roșu și verde furnizează indicatori vizuali simpli pentru șoferi, informându-i cu privire la disponibilitatea locurilor de parcare. Afisajul LCD oferă informații suplimentare, cum ar fi numărul de locuri libere sau un mesaj de ocupare, contribuind la o comunicare eficientă cu utilizatorii.

**-Securitate Îmbunătățită:** Ridicarea barierei este controlată de un servomotor, care este activat doar dacă există locuri disponibile și înălțimea mașinii este în intervalul admis. Această caracteristică adaugă un nivel suplimentar de securitate și previne accesul neautorizat.

**-Gestionarea Timpului de Parcare:** Sistemul monitorizează timpul petrecut de fiecare mașină în locul de parcare utilizând un senzor infraroșu. Această funcționalitate poate fi utilă pentru implementarea unui sistem de taxare corect și echitabil pentru utilizatorii parcării.

**-Plata și Eliberarea Barierei:** Plata taxei pentru timpul petrecut în parcarea respectivă se realizează prin intermediul Serial Monitor. După ce taxa este plătită, barierea de ieșire este eliberată, oferind un sistem eficient de gestionare a ieșirii din parcare.

## 5. Bibliografie

- Banzi, M. (2011). "Arduino Programming in 24 Hours, Sams Teach Yourself." Sams Publishing.
- Lim, M. (2009). "Robot Builder's Bonanza." McGraw-Hill Education.
- Sclater, N., & Johnston, G. (2007). "Robot Builder's Cookbook: Build and Design Your Own Robots." Syngress.
- Hertzberg, M., et al. (2014). "Arduino: A Quick-Start Guide." Pragmatic Bookshelf.
- Monk, S. (2013). "Programming Arduino: Getting Started with Sketches." McGraw-Hill Education.
- McFarland, M. (2014). "Arduino for Beginners: Essential Skills Every Maker Needs." CreateSpace Independent Publishing Platform.