**НП „Обучение за ИТ умения и кариера“**

**Модул 8: Въведение в операционни и вградени системи**

**КУРСОВ ПРОЕКТ**на тема: **Автоматизирана система за паркинг**

**Изготвил:**  
Мартин Недев

**Преподавател:**

Коля Петрова

**Група 08**

**Документация на Автоматизирана система за паркинг**

**Съдържание:**

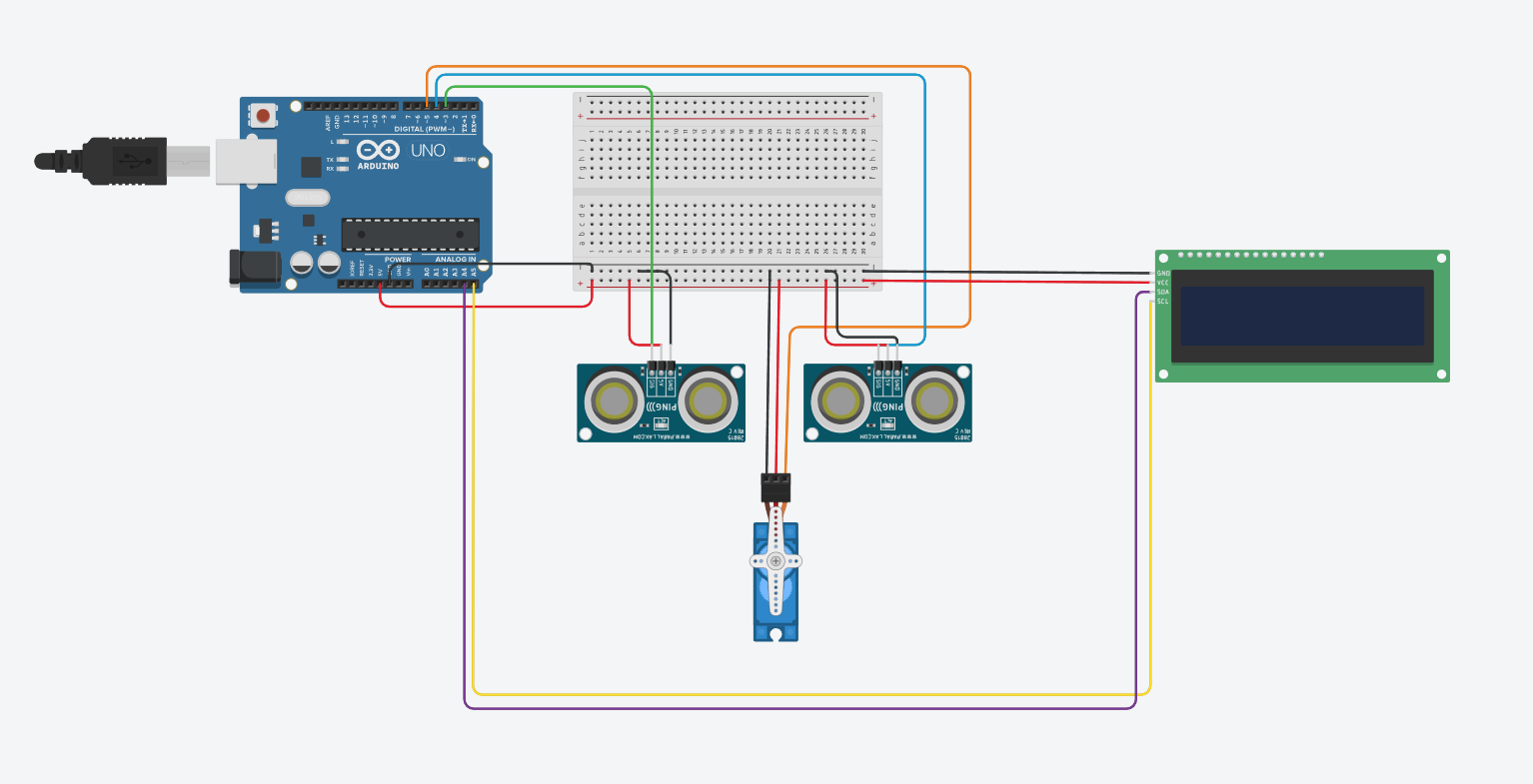
1. Описание на проекта
2. Блокова схема
3. Електрическа схема
4. Списък съставни части
5. Сорс код – описание на функционалността
6. Заключение
7. **Описание на проекта**

Проектът представлява симулация на автоматизирана система за управление на паркинг. Целта на системата е да следи и контролира броя на свободните паркоместа и да управлява достъпа на автомобили чрез автоматична бариера.

**Основна функционалност:**

* **Преброяване на автомобили:** Системата използва два ултразвукови сензора, разположени на входа и на изхода на паркинга. Чрез тях се засича посоката на движение на автомобила (влизане или излизане).
* **Контрол на достъпа:** Серво мотор управлява бариера, която се отваря, за да пропусне автомобил. Бариерата се отваря на входа само ако има свободни места. При излизане бариерата се отваря винаги.
* **Визуална информация:** LCD дисплей показва в реално време броя на оставащите свободни паркоместа.
* **Управляващ модул:** Централният компонент е микроконтролерна платка Arduino Uno, която обработва данните от сензорите и управлява серво мотора и дисплея.

Системата е проектирана да бъде напълно автономна. При засичане на автомобил на входа, системата проверява броя на свободните места. Ако има такова, бариерата се вдига. След като автомобилът премине и през втория сензор, системата регистрира влизане, броячът на свободни места намалява с единица и бариерата се спуска. Процесът е аналогичен при излизане, като броячът се увеличава.

1. **Блокова схема**
2. **Електрическа схема**

**A computer circuit board with red and green lines

AI-generated content may be incorrect.**

|  |  |  |
| --- | --- | --- |
| Компонент (Component) | Пин на Компонента (Pin) | Свързан към Пин на Arduino (Arduino Pin) |
| Arduino UNO (U1) | 5V |  |
| GND |  |
|  |  |  |
| Входен Сензор (PING1) | VCC | 5V |
| GND | GND |
| SIG | D3 |
|  |  |  |
| Изходен Сензор (PING2) | VCC | 5V |
| GND | GND |
| SIG | D4 |
|  |  |  |
| Серво Мотор (SERVO1) | VCC | 5V |
| GND | GND |
| SIG | D5 |
|  |  |  |
| LCD Дисплей 16x2 (U2) | VCC | 5V |
| GND | GND |
| SDA | A4 |
| SCL | A5 |

**4. Списък съставни части**

|  |  |  |
| --- | --- | --- |
| Означение | Количество | Компонент |
| U1 | 1 | Arduino Uno R3 |
| SERVO1 | 1 | Позиционен микро серво мотор (Positional Micro Servo) |
| PING1, PING2 | 2 | Ултразвуков сензор (PING))) |
| U2 | 1 | LCD 16x2 с I2C интерфейс (PCF8574, адрес 0x20) |

**5. Сорс код – описание на функционалността**

#include "Servo.h"

#include "LiquidCrystal\_I2C.h"

LiquidCrystal\_I2C lcd(32, 16, 2);

const int entrySensorPin = 3;

const int exitSensorPin = 4;

const int servoPin = 5;

const long thresholdDistance = 300; //cm

const double speedOfSound = 3.43e-2; //in cm/microseconds

const double microsecondsToCentimetersConstant = speedOfSound \* 0.5;

const int limitTime = 3500;

const int totalSpaces = 3;

int freeSpaces = 3;

Servo barrier;

int t1\_entered, t2\_entered, t1\_exited, t2\_exited;

bool isDetecting1, isDetecting2;

long microsecondsToCentimeters(long duration)

{

return (long)((double)duration \* microsecondsToCentimetersConstant);

}

long readDistance(int pingPin)

{

//send pulse

pinMode(pingPin, OUTPUT);

digitalWrite(pingPin, LOW);

delayMicroseconds(3);

digitalWrite(pingPin, HIGH);

delayMicroseconds(5);

digitalWrite(pingPin, LOW);

//read echoed pulse

pinMode(pingPin, INPUT);

long duration = pulseIn(pingPin, HIGH); //in microseconds

return microsecondsToCentimeters(duration);

}

void updateDisplay()

{

lcd.clear();

lcd.print("Free spaces: ");

lcd.print(freeSpaces);

}

void openBarrier()

{

barrier.write(90);

}

void closeBarrier()

{

delay(300);

barrier.write(0);

}

void handleEntering()

{

freeSpaces = max(freeSpaces-1, 0);

updateDisplay();

closeBarrier();

}

void handleExiting()

{

freeSpaces = min(freeSpaces+1, totalSpaces);

updateDisplay();

closeBarrier();

}

void setup()

{

t1\_entered = t2\_entered = 0;

t1\_exited = t2\_exited = 0;

barrier.attach(servoPin);

barrier.write(0);

lcd.init();

lcd.backlight();

lcd.setCursor(0, 0);

updateDisplay();

lcd.display();

}

void loop()

{

int currTime = millis();

long enterDistance = readDistance(entrySensorPin);

long exitDistance = readDistance(exitSensorPin);

isDetecting1 = (enterDistance < thresholdDistance);

isDetecting2 = (exitDistance < thresholdDistance);

if (enterDistance < thresholdDistance)

{

t1\_exited = currTime;

if (t1\_entered == 0)

{

if (freeSpaces > 0)

openBarrier();

t1\_entered = currTime;

}

}

if (exitDistance < thresholdDistance)

{

t2\_exited = currTime;

if (t2\_entered == 0)

{

openBarrier();

t2\_entered = currTime;

}

}

if (t1\_exited != 0 && currTime - t1\_exited > limitTime)

{

if (isDetecting2 == false)

{

t1\_entered = t1\_exited = 0;

closeBarrier();

}

}

if (t2\_exited != 0 && currTime - t2\_exited > limitTime)

{

if (isDetecting1 == false)

{

t2\_entered = t2\_exited = 0;

closeBarrier();

}

}

if (t1\_exited > 0 && t2\_exited > 0 && isDetecting1 == false && isDetecting2 == false)

{

if (t1\_entered < t2\_entered)

{

handleEntering();

}

else

{

handleExiting();

}

t1\_entered = t1\_exited = 0;

t2\_entered = t2\_exited = 0;

}

if (isDetecting1 == false && freeSpaces == 0)

{

t1\_entered = t1\_exited = 0;

}

}

**Описание на функционалността на кода**

* **Глобални променливи:**
  + **entrySensorPin, exitSensorPin, servoPin:** Дефинират пиновете, към които са свързани компонентите.
  + **totalSpaces, freeSpaces:** Определят капацитета на паркинга и текущата заетост.
  + **t1\_entered, t2\_entered, t1\_exited, t2\_exited:** Времеви маркери, които се използват за определяне на посоката на движение на автомобила.
* **Функция setup():**
  + Изпълнява се еднократно при стартиране на микроконтролера.
  + Инициализира серво мотора към пин servoPin и го поставя в начална позиция "затворено" (0 градуса).
  + Инициализира LCD дисплея, включва подсветката и изписва първоначалния брой свободни места.
* **Функция readDistance(int pingPin):**
  + Тази функция управлява един 3-пинов PING))) сензор.
  + Изпраща кратък ултразвуков импулс и след това измерва времето, за което импулсът се връща на същия пин.
  + Преобразува измереното време в микросекунди в разстояние в сантиметри.
* **Функции за управление на бариерата и дисплея:**
  + **openBarrier():** Завърта сервото на 90 градуса (отворено положение).
  + **closeBarrier():** Връща сервото в начална позиция (затворено).
  + **updateDisplay():** Изчиства дисплея и изписва актуалния брой свободни места.
* **Функция loop():**
  + Изпълнява се в безкраен цикъл.
  + В началото на всеки цикъл прочита разстоянията от двата сензора.
  + Логика за детекция:
    1. Когато някой от сензорите засече обект (distance < thresholdDistance), системата записва текущото време (currTime) като време на първо засичане (t1\_entered или t2\_entered). Ако това е входният сензор и има свободни места, бариерата се отваря.
    2. Системата продължава да обновява времето на "последно засичане" (t1\_exited, t2\_exited), докато обектът е пред сензора.
    3. Определяне на посока: Когато обектът вече не е пред нито един от сензорите (isDetecting1 == false && isDetecting2 == false), системата сравнява времената на първо засичане.
       - Ако t1\_entered < t2\_entered, значи първо е задействан входният сензор -> автомобилът влиза. Извиква се handleEntering().
       - Ако t2\_entered < t1\_entered, значи първо е задействан изходният сензор -> автомобилът излиза. Извиква се handleExiting().
    4. След регистриране на събитие, всички времеви маркери се нулират.
  + **Таймаут логика:**Ако обект остане пред един от сензорите за повече от limitTime и не задейства другия, системата приема, че е фалшива аларма или спрял автомобил, нулира състоянието и затваря бариерата.

**6. Заключение**

Разработеният проект успешно симулира работата на автоматизирана паркинг система. Постигнати са основните цели: надеждно отчитане на влизащи и излизащи автомобили, управление на бариера и визуализация на наличните места

1. **Връзки към проекта**

Tinkercad: <https://www.tinkercad.com/things/blgF4JnYYq6-parking-system?sharecode=-KcXJ7TWoeJeKd_iPmBfKFTkKNStKvSNpXK1vxjH2Gs>

Github: <https://github.com/MartinNN07/AutomaticParkingSystem>