Shahjalal University of Science and Technology, Sylhet

Department of

Computer Science & Engineering

Course Name	: Microprocessor and Interfacing Lab
Course Code	: **CSE-368**
Group	: **07**
Project Title	: ***Automated Car Parking System***

Submitted To:

Abdullah Al Noman

Lecturer,

Department of Computer Science & Engineering, SUST

Submitted By:

Al-Amin

Reg. no.: **2020331057**

Session: 2020-21

Department of Computer Science & Engineering, SUST

Members:

1. Member -1:
   - Name : Al-Amin
   - Reg. no. : 2020331057
   - Email : alamin.sust.cse@gmail.com
2. Member-2:
   - Name : Riaz Ahmed
   - Reg. no. : 2020331079
   - Email : riazahmed2246@gmail.com
3. Member-3:
   - Name : Mahmudul Ferdous
   - Reg. no. : 2020331037
   - Email : mahmudulferdous2050@gmail.com
4. Member-4:
   - Name : Md. Sakib Hassan
   - Reg. no. : 2020331109
   - Email : mdsakibhassan256124@gmail.com
5. Member-5:
   - Name : MD Rafi
   - Reg. no. : 2019331096
   - Email : rafimd4212@gmail.com

Description of the Project

Abstract:

The automated car parking system aims to streamline parking management using microprocessor and microcontroller technology. The system automates the detection of parking space availability and controls a gate mechanism to manage vehicle entry.

Introduction:

This project presents an Automated Car Parking System using an Arduino UNO microcontroller. The system is designed to streamline the process of vehicle entry and exit in parking facilities by automating gate control and parking space management. It employs IR Proximity Sensors to detect vehicle presence, a 16x2 LCD i2c Display to show real-time parking status, and a Servo Motor to control the gate. The Arduino processes the sensor inputs to update the parking availability displayed on the LCD and to manage the gate's opening and closing actions. This project aims to provide an efficient and user-friendly solution for managing parking spaces, reducing human intervention, and enhancing overall parking efficiency. The system can be scaled and modified to accommodate various parking lot sizes and configurations, making it a versatile tool for modern parking management.

Components and Their Functions:

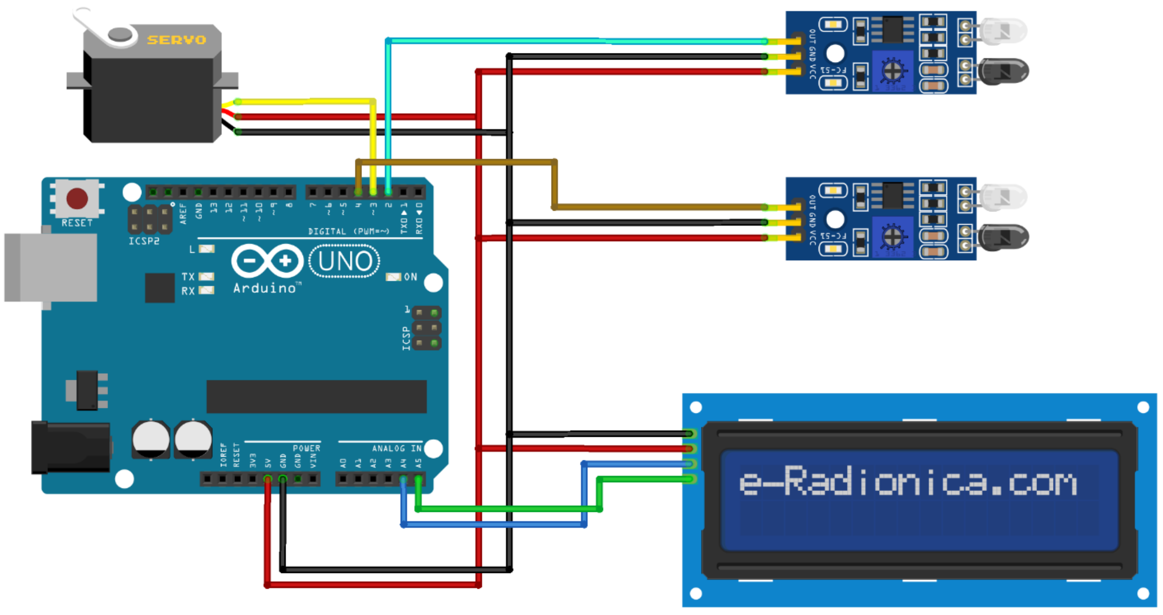
- **Arduino UNO**: The central microcontroller that processes sensor inputs and controls the servo motor.
- **IR Proximity Sensors**: Used to detect the presence of vehicles in parking spaces. One sensor detects entry, and another detects the occupation of the parking space.
- **16x2 LCD i2c Display**: Displays the status of the parking space, whether it is available or occupied.
- **Servo Motor**: Controls the gate mechanism to allow or restrict vehicle entry based on sensor inputs.

How It Works:

1. **Initialization**: The Arduino initializes the servo motor and the LCD display.
2. **Sensor Reading**: Continuously reads the state of the IR sensors to detect the presence or absence of vehicles.
3. **Display Update**: Updates the LCD display with the status of the parking space (Available or Occupied).
4. **Gate Control**: Opens the gate if the parking space is available and closes it when the space is occupied.

<u>Diagram of the Project Including Connections Between Components</u>

Circuit Diagram:



The diagram above illustrates the connections between the components of the automated car parking system project, which includes an Arduino UNO, IR proximity sensors, a 16x2 LCD i2c display, and a servo motor.

Circuit Assemble:

Assemble the circuit by connecting the components to the Arduino UNO. For the IR Proximity Sensor, connect VCC to the 5V pin, GND to the GND pin, and the signal pin to a digital pin (e.g., D2) on the Arduino. For the 16x2 LCD i2c Display, connect VCC to the 5V pin, GND to the GND pin, SDA to the A4 pin, and SCL to the A5 pin on the Arduino. For the Servo Motor, connect VCC to the 5V pin, GND to the GND pin, and the signal pin to a PWM-capable digital pin (e.g., D9) on the Arduino. Use jumpers to ensure secure and reliable connections for all components.

Code and GitHub Link of the Project

Code:

Here is the code in CPP language for the automated car parking system using Arduino.

Here is the explanation of the code:

Libraries and Definitions:

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

- **Wire.h and LiquidCrystal_I2C.h are included for I2C communication with the LCD.**
- **Servo.h is included to control the servo motor.**

LCD and Servo Initialization:

```cpp
// Initialize LCD and Servo objects
LiquidCrystal_I2C lcd(0x27, 16, 2);  // Change the HEX address if necessary
Servo myservo1;
```

- **The LCD is initialized with an I2C address of 0x27 and dimensions of 16 columns and 2 rows.**
- **A Servo object is created for controlling the servo motor.**

IR Sensor and Slot Definitions:

```cpp
// Define IR sensor pins
const int IR1 = 8;`
const int IR2 = 12;

// Define the total number of parking slots
int totalSlots = 4;
int availableSlots = totalSlots;

// Flags for IR sensors
bool carEntering = false;
bool carExiting = false;
```

- **IR1 and IR2 define the pins for the IR sensors.**
- **totalSlots and availableSlots track the total and available parking slots.**
- **carEntering and carExiting are flags to detect car movement.**

Setup Function:

```
void setup() {
  // Initialize serial communication
  Serial.begin(9600);

  // Initialize the LCD
  lcd.init();
  lcd.backlight();

  // Set IR sensor pins as input
  pinMode(IR1, INPUT);
  pinMode(IR2, INPUT);

  // Attach the servo to pin 3 and set initial position
  myservo1.attach(3);
  myservo1.write(100);

  // Display initial message on the LCD
  lcd.setCursor(0, 0);
  lcd.print("     ARDUINO    ");
  lcd.setCursor(0, 1);
  lcd.print(" PARKING SYSTEM ");
  delay(2000);
  lcd.clear();
}
```

- **Serial.begin(9600) initializes serial communication at 9600 baud rate.**
- **lcd.init() and lcd.backlight() initialize the LCD and turn on its backlight.**
- **pinMode(IR1, INPUT) and pinMode(IR2, INPUT) set the IR sensor pins as input.**
- **myservo1.attach(3) attaches the servo motor to pin 3 and sets it to the initial position (100).**
- **The LCD displays a welcome message for 2 seconds.**

Loop Function:

```
void loop() {
  // Check for car entering
  if (digitalRead(IR1) == LOW && !carEntering) {
    delay(50);  // Small delay to debounce
    if (digitalRead(IR1) == LOW) {  // Confirm the reading
      carEntering = true;
      if (availableSlots > 0) {
        myservo1.write(0);  // Open gate
        availableSlots -= 1;
        lcd.clear();
        lcd.setCursor(0, 0);
        lcd.print(" Car Entered ");
        delay(3000);  // Keep gate open for a while
        myservo1.write(100);  // Close gate
        lcd.clear();
      } else {
        lcd.clear();
        lcd.setCursor(0, 0);
        lcd.print("    SORRY :(    ");
        lcd.setCursor(0, 1);
        lcd.print("  Parking Full  ");
        delay(3000);
        lcd.clear();
      }
    }
  }

  // Check for car exiting
  if (digitalRead(IR2) == LOW && !carExiting) {
    delay(50);  // Small delay to debounce
    if (digitalRead(IR2) == LOW) {  // Confirm the reading
      carExiting = true;
      if (availableSlots < totalSlots) {
        myservo1.write(0);  // Open gate
        availableSlots += 1;
        lcd.clear();
        lcd.setCursor(0, 0);
        lcd.print(" Car Exited ");
        delay(3000);  // Keep gate open for a while
        myservo1.write(100);  // Close gate
        lcd.clear();
      }
    }
  }

  // Reset flags after sensors are no longer triggered
  if (carEntering) {
    if (digitalRead(IR1) == HIGH) {  // Wait for sensor to reset
      carEntering = false;
    }
  }

  if (carExiting) {
    if (digitalRead(IR2) == HIGH) {  // Wait for sensor to reset
      carExiting = false;
    }
  }

  // Display welcome message and available slots
  lcd.setCursor(0, 0);
  lcd.print("    WELCOME!    ");
  lcd.setCursor(0, 1);
  lcd.print("Slots Left: ");
  lcd.print(availableSlots);

  // Debugging information
  Serial.print("IR1: ");
  Serial.print(digitalRead(IR1));
  Serial.print(" IR2: ");
  Serial.print(digitalRead(IR2));
  Serial.print(" Slots: ");
  Serial.println(availableSlots);
}
```

- Car Entering: If IR1 detects a car, the system checks if slots are available. If so, it opens the gate (servo motor), updates the slot count, and displays "Car Entered" on the LCD. If full, it displays "Parking Full".
- Car Exiting: If IR2 detects a car exiting, the system opens the gate, updates the slot count, and displays "Car Exited" on the LCD.
- Reset Flags: The flags carEntering and carExiting are reset once the IR sensors no longer detect the car.
- Display Status: Continuously displays a welcome message and the number of available slots on the LCD.
- Debugging Information: Prints the status of the IR sensors and available slots to the serial monitor for debugging

GitHub Link:

Here attached a GitHub repository where we uploaded the code along with others additional files (schematics, libraries used, Project Proposal Documentation,). To demonstrate:

GitHub Repository Link

Steps to Set Up the Project

1. **Gather Components**

To begin, you need to gather all the necessary hardware components for the project. These include:

- **Arduino UNO**: The main microcontroller that will be used to process inputs and control outputs.
- **IR Proximity Sensor**: This sensor will detect the presence of a vehicle in the parking space.
- **16x2 LCD i2c Display**: This display will show the status of the parking space, such as "Parking Available" or "Parking Full".
- **Servo Motor**: The servo motor will act as a gate, opening and closing based on the parking space status.
- **Jumpers**: These are used to make the electrical connections between the components and the Arduino.

Ensure that all components are in working condition before proceeding to the next step.

2. **Circuit Assembly**

Next, you need to assemble the circuit by connecting the components to the Arduino UNO. Follow the provided schematic diagram carefully:

- **IR Proximity Sensor**:

  - ➢ Connect the VCC pin to the 5V pin on the Arduino.
  - ➢ Connect the GND pin to the GND pin on the Arduino.
  - ➢ Connect the signal pin to a digital pin on the Arduino (e.g., D2).

- **16x2 LCD i2c Display**:

  - ➢ Connect the VCC pin to the 5V pin on the Arduino.
  - ➢ Connect the GND pin to the GND pin on the Arduino.
  - ➢ Connect the SDA pin to the A4 pin on the Arduino.
  - ➢ Connect the SCL pin to the A5 pin on the Arduino.

- **Servo Motor**:

  - ➢ Connect the VCC pin to the 5V pin on the Arduino.
  - ➢ Connect the GND pin to the GND pin on the Arduino.
  - ➢ Connect the signal pin to a PWM-capable digital pin on the Arduino (e.g., D9).

- **Jumpers**:
    - ➢ Use the jumpers to make the above connections secure and reliable.

Double-check all connections to ensure they are correct and secure.

3. Code Development

With the hardware set up, the next step is to write the Arduino code to control the components. Open the Arduino IDE and write a sketch (program) that performs the following tasks:

- Initialize the LCD display and servo motor.
- Continuously read the input from the IR Proximity Sensor.
- Update the LCD display with the current parking status based on the sensor input.
- Control the servo motor to open or close the gate depending on whether the parking space is available or full.

Here is the code for the automated car parking system using Arduino:

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

// Initialize LCD and Servo objects
LiquidCrystal_I2C lcd(0x27, 16, 2);  // Change the HEX address if necessary
Servo myservo1;

// Define IR sensor pins
const int IR1 = 8;`
const int IR2 = 12;
// Define the total number of parking slots
int totalSlots = 4;
int availableSlots = totalSlots;

// Flags for IR sensors
bool carEntering = false;
bool carExiting = false;

void setup() {
  // Initialize serial communication
  Serial.begin(9600);

  // Initialize the LCD
  lcd.init();
  lcd.backlight();
  // Set IR sensor pins as input
  pinMode(IR1, INPUT);
  pinMode(IR2, INPUT);

  // Attach the servo to pin 3 and set initial position
  myservo1.attach(3);
  myservo1.write(100);

  // Display initial message on the LCD
  lcd.setCursor(0, 0);
  lcd.print("     ARDUINO    ");
  lcd.setCursor(0, 1);
  lcd.print(" PARKING SYSTEM ");
  delay(2000);
  lcd.clear();
}

void loop() {
  // Check for car entering
  if (digitalRead(IR1) == LOW && !carEntering) {
    delay(50);  // Small delay to debounce
    if (digitalRead(IR1) == LOW) {  // Confirm the reading
      carEntering = true;
      if (availableSlots > 0) {
        myservo1.write(0);  // Open gate
        availableSlots -= 1;
        lcd.clear();
        lcd.setCursor(0, 0);
        lcd.print(" Car Entered ");
        delay(3000);  // Keep gate open for a while
        myservo1.write(100);  // Close gate
        lcd.clear();
      } else {
        lcd.clear();
        lcd.setCursor(0, 0);
        lcd.print("    SORRY :(    ");
        lcd.setCursor(0, 1);
        lcd.print("  Parking Full  ");
        delay(3000);
        lcd.clear();
      }
    }
  }
  // Check for car exiting
  if (digitalRead(IR2) == LOW && !carExiting) {
    delay(50);  // Small delay to debounce
    if (digitalRead(IR2) == LOW) {  // Confirm the reading
      carExiting = true;
      if (availableSlots < totalSlots) {
        myservo1.write(0);  // Open gate
        availableSlots += 1;
        lcd.clear();
        lcd.setCursor(0, 0);
        lcd.print(" Car Exited ");
        delay(3000);  // Keep gate open for a while
        myservo1.write(100);  // Close gate
        lcd.clear();
      }
    }
  }

  // Reset flags after sensors are no longer triggered
  if (carEntering) {
    if (digitalRead(IR1) == HIGH) {  // Wait for sensor to reset
      carEntering = false;
    }
  }
  if (carExiting) {
    if (digitalRead(IR2) == HIGH) {  // Wait for sensor to reset
      carExiting = false;
    }
  }

  // Display welcome message and available slots
  lcd.setCursor(0, 0);
  lcd.print("    WELCOME!    ");
  lcd.setCursor(0, 1);
  lcd.print("Slots Left: ");
  lcd.print(availableSlots);

  // Debugging information
  Serial.print("IR1: ");
  Serial.print(digitalRead(IR1));
  Serial.print(" IR2: ");
  Serial.print(digitalRead(IR2));
  Serial.print(" Slots: ");
  Serial.println(availableSlots);
}
```

4. Testing

Once the code is ready, upload it to the Arduino UNO using the Arduino IDE. Ensure the board is connected to your computer via a USB cable. After the upload is complete, observe the system in action:

- Place an object in front of the IR Proximity Sensor to simulate a vehicle.
- Check if the LCD display updates to show "Parking Available" or "Parking Full" correctly.
- Verify that the servo motor opens and closes the gate as expected based on the sensor input.

Test the system under different conditions to ensure it behaves as intended.

5. Refinement

After the initial testing, you may need to make adjustments to optimize the system's performance. Consider the following refinements:

- **Code Adjustments**: Fine-tune the code to improve responsiveness, accuracy, and reliability. This may include adjusting sensor sensitivity, refining display messages, or optimizing the servo motor control.
- **Hardware Connections**: Ensure all connections are secure and free from any loose or intermittent connections. Use soldering if necessary for a more permanent setup.
- **Component Placement**: Optimize the placement of sensors and the servo motor to ensure they function effectively in the intended environment.

Continue testing and refining until the system performs reliably and meets the project objectives.

Data

The system was tested with a total of 4 parking slots. The following data was recorded during the test:

Time (min)	IR1 Sensor (Entry)	IR2 Sensor (Exit)	Available Slots	Gate Status
0	HIGH	HIGH	4	CLOSED
2	LOW	HIGH	3	OPEN
3	HIGH	HIGH	3	CLOSED

6	HIGH	LOW	4	OPEN
7	HIGH	HIGH	4	CLOSED
10	LOW	HIGH	3	OPEN
11	HIGH	HIGH	3	CLOSED
15	LOW	HIGH	2	OPEN
16	HIGH	HIGH	2	CLOSED
20	HIGH	LOW	3	OPEN
21	HIGH	HIGH	3	CLOSED

- **IR1 Sensor (Entry)**: Detects a car entering the parking lot.
- **IR2 Sensor (Exit)**: Detects a car exiting the parking lot.
- **Available Slots**: Tracks the number of parking slots available.
- **Gate Status**: Indicates whether the gate is open or closed based on sensor input and slot availability.

Results

The Automated Car Parking System successfully managed the entry and exit of vehicles, accurately updating the available parking slots on the LCD display. The IR Proximity Sensors reliably detected vehicle presence, and the Arduino-controlled servo motor effectively operated the gate, opening and closing as necessary. The system was able to maintain an accurate count of available parking slots, demonstrating its effectiveness in real-time parking management.

Conclusion

The Automated Car Parking System successfully managed the entry and exit of vehicles, accurately updating the available parking slots on the LCD display. The IR Proximity Sensors reliably detected vehicle presence, and the Arduino-controlled servo motor effectively operated the gate, opening and closing as necessary. The system was able to maintain an accurate count of available parking slots, demonstrating its effectiveness in real-time parking management.