Abstract

Parking congestion is a common issue in urban areas worldwide, where limited parking spaces and inefficient parking management systems result in long queues, increased traffic, and frustration among drivers. Traditional parking systems rely heavily on manual intervention by attendants to inform drivers about available parking spaces, which can cause delays and confusion. This project presents an automated toll gate parking system designed to solve these challenges.

The system utilizes IR sensors, servo motors, and an LCD display to manage parking slots efficiently. The IR sensors detect vehicle entry and exit, dynamically updating to show how many parking spaces are available. When a vehicle enters, the system decreases the slot count and opens the toll gate using the servo motor. On the other hand, when a vehicle exits, it increases the number of slots and closes the gate. The LCD display provides real-time information about the available slots to incoming vehicles, helping drivers make decisions without waiting in long queues.

By using an Arduino Uno, jumper wires, a breadboard, and other components, the system removes the need for manual involvement and minimizes the chances of human error. It simplifies parking management by automatically tracking available spaces and improving the flow of traffic. This solution can easily be adapted to different parking facilities, making it a practical way to tackle parking problems in cities around the world.

Table of Contents

Table of Figures

Table of Tables

1. Introduction

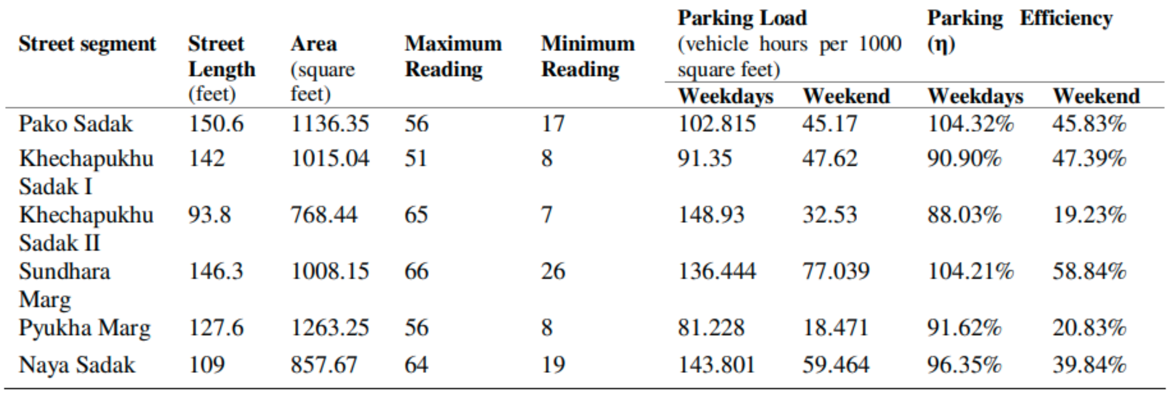
The Internet of Things (IoT) is changing the way industries operate by connecting devices through sensors, software and networks. These systems are developed to tackle everyday challenges, making daily tasks more efficient, safer and easier. By using components such as microcontrollers, sensors and automated systems, IoT solutions are being effectively applied in fields like transportation, healthcare, and urban management.

The “Automated Toll Gate Parking System” project focuses on solving the problem of parking congestion in urban areas. With limited parking spots and inefficient parking management, drives often face long waits, traffic jams, and frustrations. A smarter solution is introduced through this system by automating parking management, reducing human involvement, and providing drivers with real-time updates on available parking spots.

1.1. Current scenario

Parking management is a major issue in Nepal, especially in cities like Kathmandu. According to the Kathmandu Valley Traffic Police Office, there were 1.75 million vehicles in the valley in 2022, a number that continues to rise each year.

With the increase in vehicles, parking slots are becoming difficult to find, causing problems for drivers who often struggle to park. Many drivers get frustrated and end up parking on the roadside, which leads to traffic jams and makes the situation even worse. The lack of proper parking management only adds to these challenges, showing the need for a better solution.



Street segment	Street Length (feet)	Area (square feet)	Maximum Reading	Minimum Reading	Parking Load (vehicle hours per 1000 square feet)		Parking Efficiency (η)	
					Weekdays	Weekend	Weekdays	Weekend
Pako Sadak	150.6	1136.35	56	17	102.815	45.17	104.32%	45.83%
Khechapukhu Sadak I	142	1015.04	51	8	91.35	47.62	90.90%	47.39%
Khechapukhu Sadak II	93.8	768.44	65	7	148.93	32.53	88.03%	19.23%
Sundhara Marg	146.3	1008.15	66	26	136.444	77.039	104.21%	58.84%
Pyukha Marg	127.6	1263.25	56	8	81.228	18.471	91.62%	20.83%
Naya Sadak	109	857.67	64	19	143.801	59.464	96.35%	39.84%

Figure 1: Data Regarding the Parking in Kathmandu

Figure 2: Current scenario

1.2. Problem Statement and Project as a solution

Parking management is a significant issue in many urban areas, especially in cities like Kathmandu, where the growing number of vehicles has made it difficult to find available parking spaces. Drivers often face frustration and waste time searching for a spot. Additionally, the lack of effective parking management leads to vehicles being parked on roadsides, causing traffic jams and further complications.

This project aims to solve these problems by implementing an Automated Toll Gate Parking System. The system will include a display board that shows real-time updates on the availability of parking slots. The gate will only open if there is an available parking space, helping drivers find parking more efficiently. The system requires no manpower, making it cost-effective and reliable compared to traditional parking systems. By automating the process, it will reduce traffic congestion, save time, and improve the overall parking management experience.

Figure 3: Project as a solution

2. Aim and Objectives

2.1. Aim

The aim of this project is to develop an Automated Toll Gate System that detects available parking slots, allows vehicles to enter if spaces are free, and notifies users when parking is unavailable.

2.2. Objectives

The objective of the Automates Toll Gate Parking System are as follows:

i. Monitor parking space available in real-time, making it easier to identify available parking slots.

ii. Reduce the time spent searching for empty parking spaces by providing real-time updates on availability.

iii. Reduce the operational cost by implementing automated parking management and minimizing human involvement.

iv. Enhance efficiency by automating the parking management process and reducing human error.

v. Maintain an accurate count of parked vehicles and helps in reducing queues while making operations easier.

vi. Reduce traffic congestion in parking areas by ensuring efficient flow and management.

3. Background

3.1. System Overview

The Automated Toll Gate Parking System is designed to manage parking operations efficiently and with minimal human involvement. The system integrates several hardware components, including IR sensors, an LCD display, servo motors, and an Arduino UNO, to ensure seamless operation. IR sensor plays a critical role in detecting vehicles at both entry and exit points. When a vehicle is detected, the system updates the availability of parking slots in real-time on the LCD display and trigger the gate to open or close accordingly.

The system also handles scenarios like full parking capacity, displaying a “Sorry Parking Full” message and preventing entry when no slots are available. By automating these processes, the system aims to reduce errors, save time, and provide a smooth experience for users. This solution highlights the application of IoT and automation in solving real-world challenges like parking management.

3.2. Design Diagram

i. Block diagram

A block diagram is a simple visual representation of a system that uses blocks to depict major components and lines to show their connections or relationships. It focuses on the overall structure and flow of information rather than detailed internal workings, making it easy to understand complex systems. (geeksforgeeks, 2024)

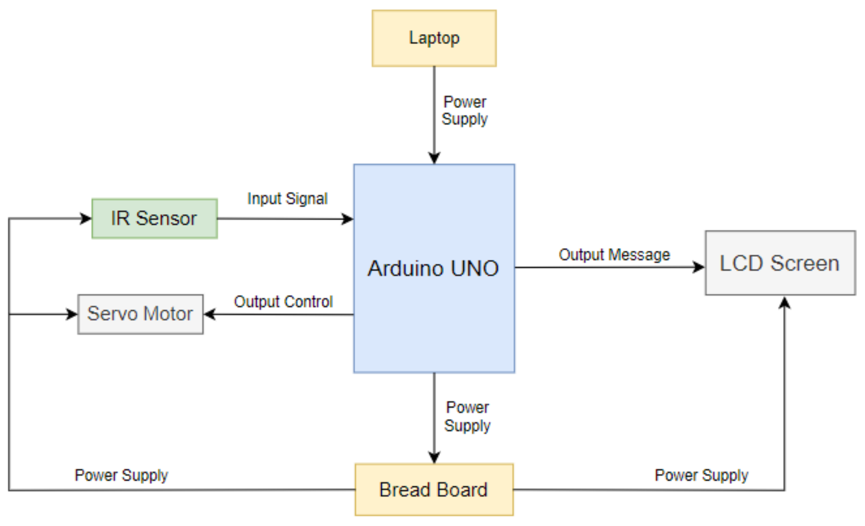


Figure 4: Block diagram

ii. System architecture

A system architecture is a visual representation of the structure and components of a system, showcasing their relationships and interactions. It provides an abstract view of how the system functions and communicates, serving as a blueprint for design and development. (edrawsoft, 2025)

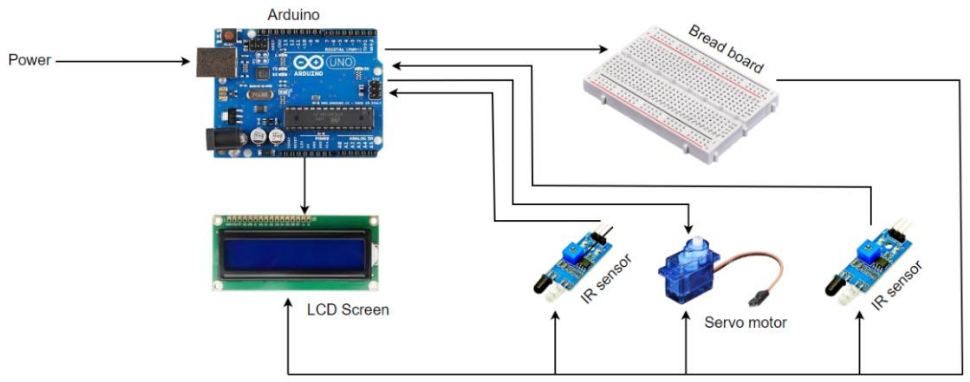


Figure 5: System Architecture

iii. Circuit diagram

A circuit diagram is a visual representation of an electrical circuit. It uses standardized symbols to illustrate electrical components such as resistors, capacitors, transistors, and wires, showing how they are connected to each other. Circuit diagrams are essential for understanding analysing, designing, and troubleshooting electrical circuits. (testbook, 2023)

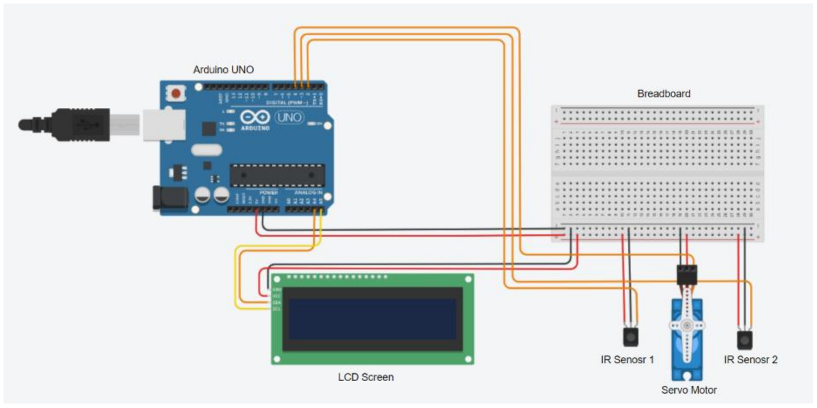


Figure 6: Circuit diagram

iv. Schematic diagram

A schematic diagram is a simplified representation of an electrical circuit, showing the components and their connections using standardized symbols. It does not include physical details such as the size or arrangement of components but focuses on the functional relationships between them. Schematic diagrams are used to design, analyze, and troubleshoot electrical systems by providing a clear, abstract view of the circuit's layout. (Yogendrappa, 2021)

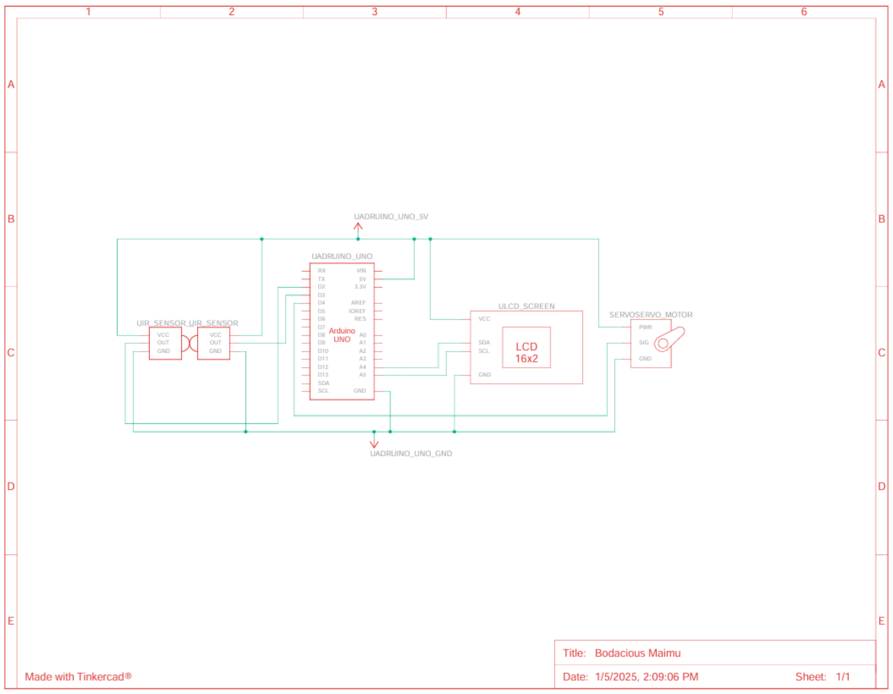


Figure 7: Schematic diagram

v. Flowchart showing how the system works in parking system

This flowchart the step-by-step process of vehicle entry, slot availability and exit. It uses symbols like ovals for start and end points, rectangles for processes, parallelograms for input/output, diamonds for decision points, and arrow to show the flow of actions.

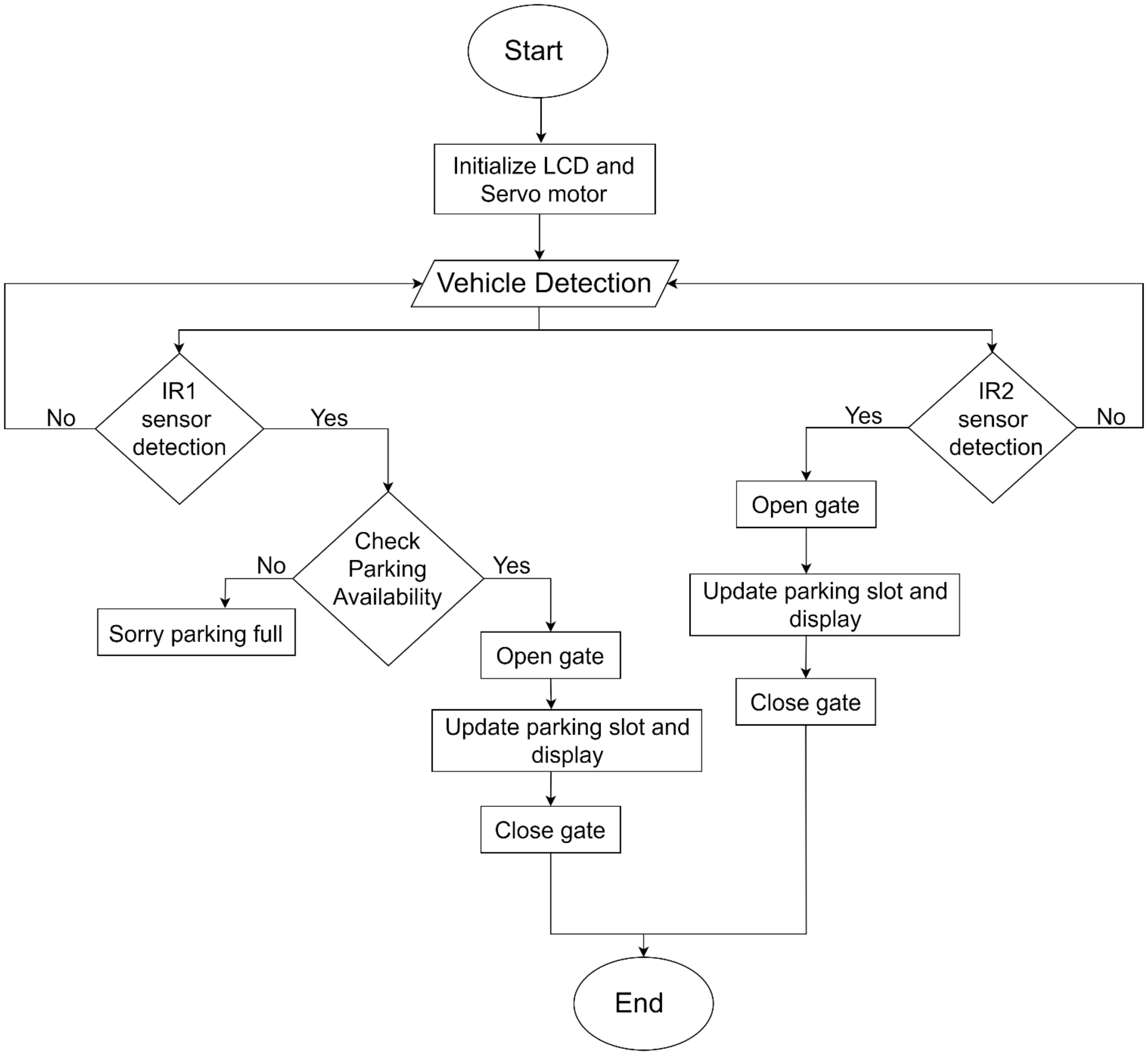


Figure 8: Flowchart showing how the system works

vi. Flowchart showing how the code works in parking system

A flowchart showing how the code works in parking system represents the logical sequence of programming steps involved in managing the system. It uses symbols such as ovals for start and end, rectangles for code execution or functions, diamonds for decisions-making, parallelograms for input/output and arrows indicates the flow of control throughout the program operations.

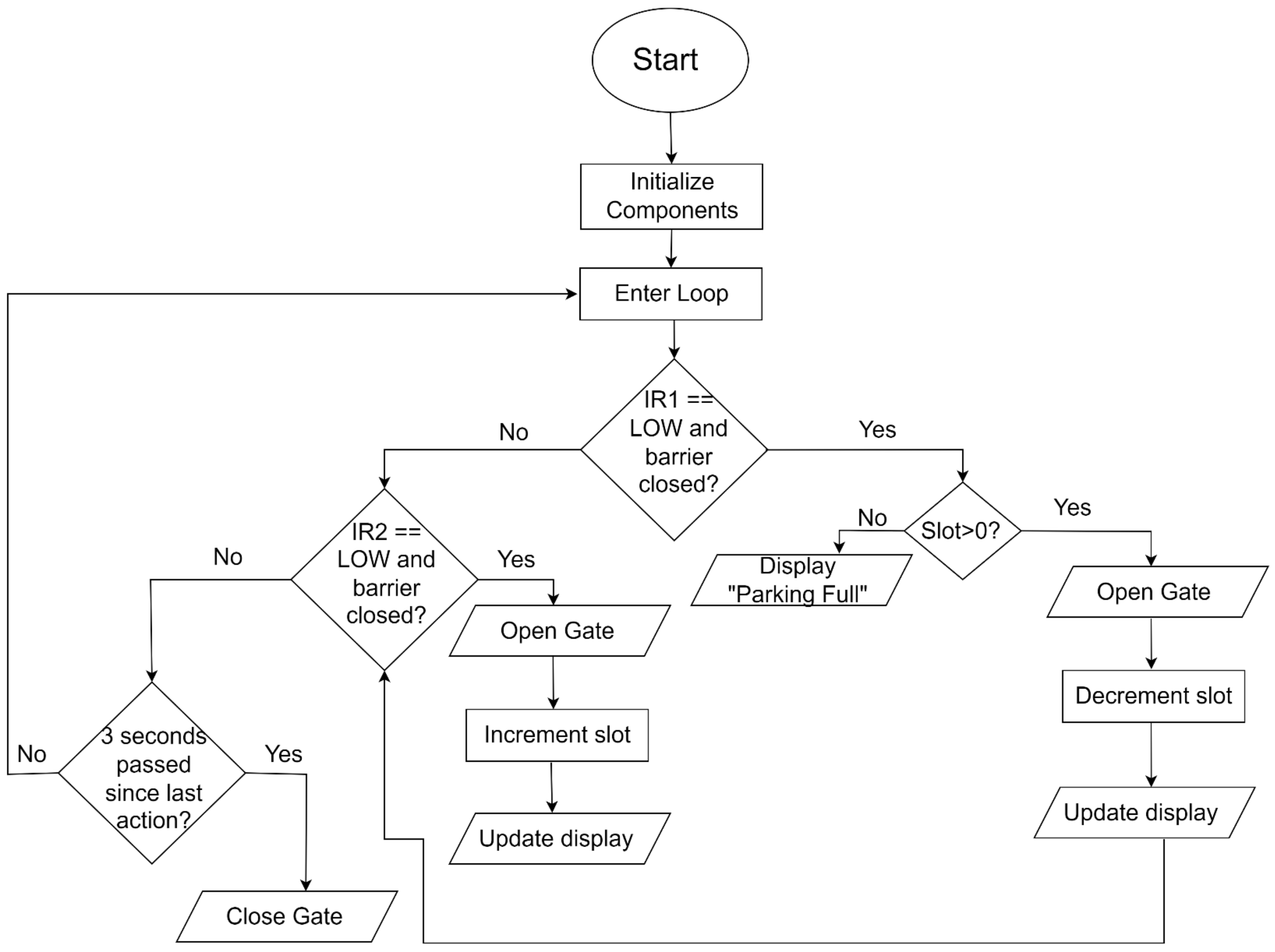


Figure 9: Flowchart showing how the code works

3.3. Requirement Analysis

3.3.1. Hardware Requirements

i. Arduino UNO

ii. Breadboard

iii. IR Proximity Sensor

iv. Servo Motor

v. Jumper Wires

vi. LCD display with I2C Module

3.2.2. Software requirements

i. Arduino IDE

ii. Draw.io

iii. MS Word

iv. Tinkercad:

4. Development

Step by step development process:

Phase 1: Planning and Design

The planning and design phase of the Automated Toll Gate Parking System involves identifying the required components, including Arduino UNO, IR sensors, servo motor, LCD display, jumper wires, and breadboard. Various diagrams, such as the block diagram, system architecture, circuit diagram and schematic diagram, were created using Tinkercad to visualize the system's structure and electrical connections. Additionally, flowcharts were developed to outline the workflow, detailing processes like vehicle detection, gate control, and parking slot updates. This phase ensures a well-defined blueprint for the system's development and minimizes potential errors during implementation.

Phase 2: Collecting components

The development of the Automated Toll Gate Parking System begins with gathering the essential components, including an Arduino UNO, LCD display, IR sensors, servo motor, jumper wires and a breadboard. These components are the foundation of the system and are required to assemble the circuit, ensuring proper functionality and integration of the hardware.

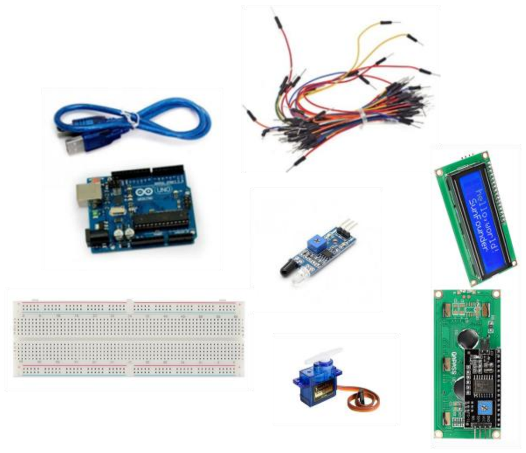


Figure 10: Phase 2 Collecting components

Phase 3: Powering the Initial Connections

This phase sets up the power supply for the system. The 5V and GND pins of the Arduino are connected to the positive and negative rails of the breadboard. Next, attach the GND and VCC pins of the LCD to the negative and positive rails of the breadboard. At last, connect the SDA and SCL pins of the LCD to the A4 and A5 pins of the Arduino.

- Connection Between Arduino UNO and breadboard.

  5V → Positive Rails

  GND → Negative Rails

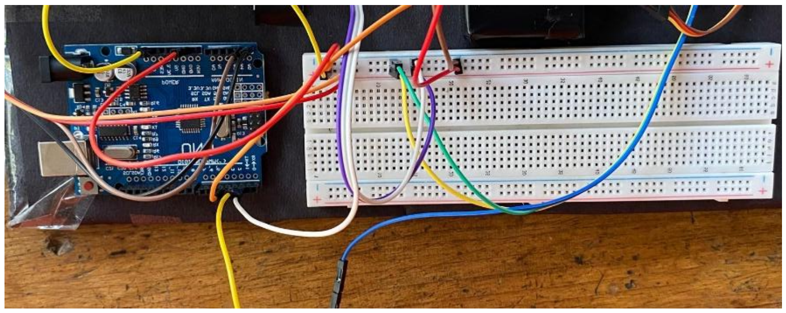


Figure 11: Phase 3 Powering the Initial Connections

- Connection of LCD display in Arduino and breadboard.

  GND → Negative Rails
  VCC → Positive Rails
  SDA → A4
  SCL → A5

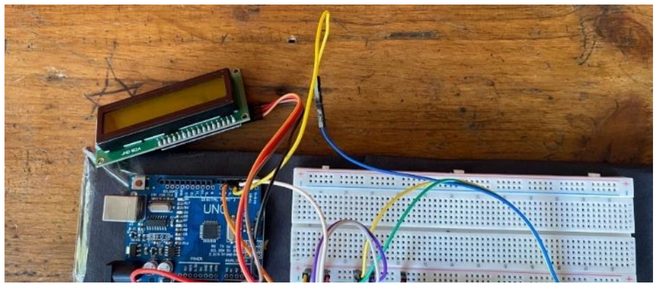


Figure 12: Phase 3 LCD display in Arduino and breadboard

Phase 4: Adding IR Sensors

IR sensors are installed at the entry and exit points to detect vehicles. The GND and VCC pins of both the IR1 and IR2 sensors are connected to the breadboard's negative and positive rails. The OUT pin of the IR1 sensor is connected to D2, and the OUT pin of the IR2 sensor is connected to D3 on the Arduino. This enables the system to detect vehicle movements at entry and exit points.

- Connection of IR1 and IR2 sensors in Arduino UNO and breadboard.
    - IR1

        GND → Negative Rails

        VCC → Positive Rails

        OUT → D2
    - IR 2

        GND → Negative Rails

        VCC → Positive Rails

        OUT → D3

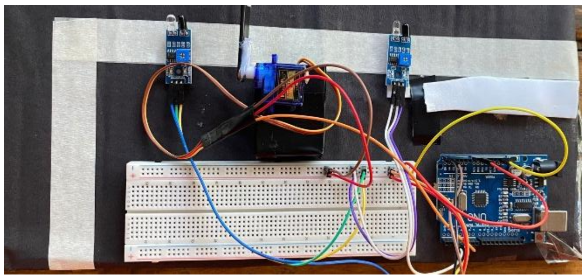


Figure 13: Phase 4 Connection of IR1 and IR2

Phase 5: Connecting the Servo Motor

The servo motor is responsible for controlling the gate movement. Connect the servo motor's GND and VCC pins to the breadboard's negative and positive rails. Then, attach the servo motor's signal pin to D4 on the Arduino. This allows the Arduino to command the servo motor for opening and closing the gate.

- Connection of servo motor in Arduino UNO and breadboard.

  GND (Brown) → Negative Rails
  VCC (Red) → Positive Rails
  Signal (Yellow) → D4

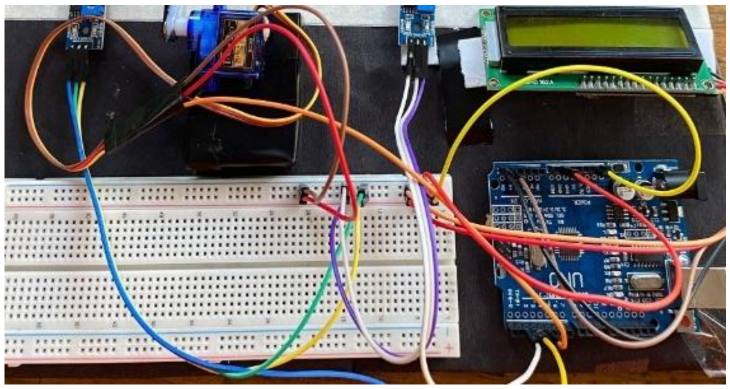


Figure 14: Phase 5 Connecting the servo motor

Phase 6: Powering and Testing the System

The system is powered by connecting the Arduino to a laptop via a USB cable, which also facilitates uploading the developed code using the Arduino IDE. This code manages the operations of the IR sensors, servo motor, and LCD display, enabling the system to handle vehicle detection, gate control, and parking availability display. After uploading, the system is tested to ensure all components function seamlessly, such as detecting vehicles, controlling the gate, and displaying the available parking slots. This phase confirms the system's overall performance and reliability.

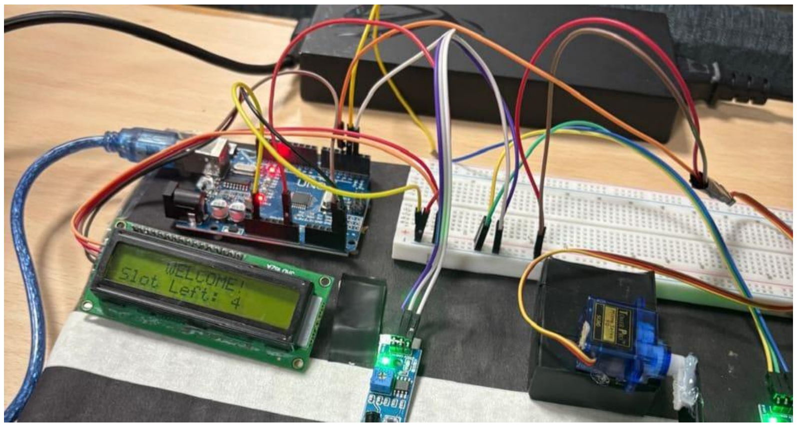


Figure 15: Phase 6 Powering and Testing the System

5. Result and Findings

5.1. Testing:

5.1.1. Real-Time Slot Availability Testing.

Table 1: Test 1 (Real-Time Slot Availability Testing)

Test	1 (Real-Time Slot Availability Testing)
Objective	Verify if the system updates and displays the correct number of available and occupied parking slots on the LCD display.
Activity	Simulate vehicles entering and exiting the parking lot and observe the LCD display for changes in slot availability.
Expected Outcomes	The system should detect vehicles entering and exiting the parking space and accurately update slot count on the LCD display.
Actual Result	The system accurately updates the slot count and reflects it on the LCD display.
Conclusion	The test is successful, and the system works as intended, ensuring accurate real-time updates of parking slot availability.

- While Entering:
  - Here total number of slot left is 3.

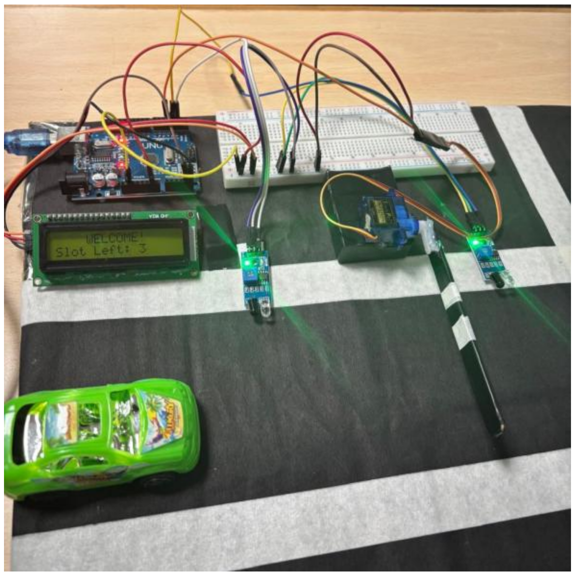


Figure 16: Number of slots left 3

- When a car approaches the sensor, the IR sensor detects its presence, triggering the system to reduce the count of available parking slot. If slots are available, the gate opens automatically to allow the car to enter.

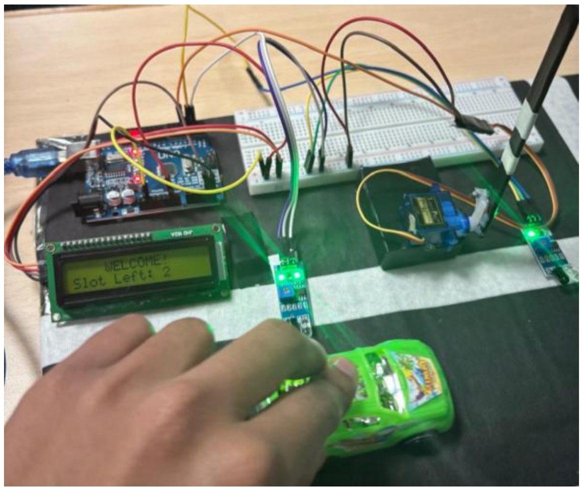


Figure 17: reduce the count of available parking slot

- Here, the total slot left decreases and becomes 2.

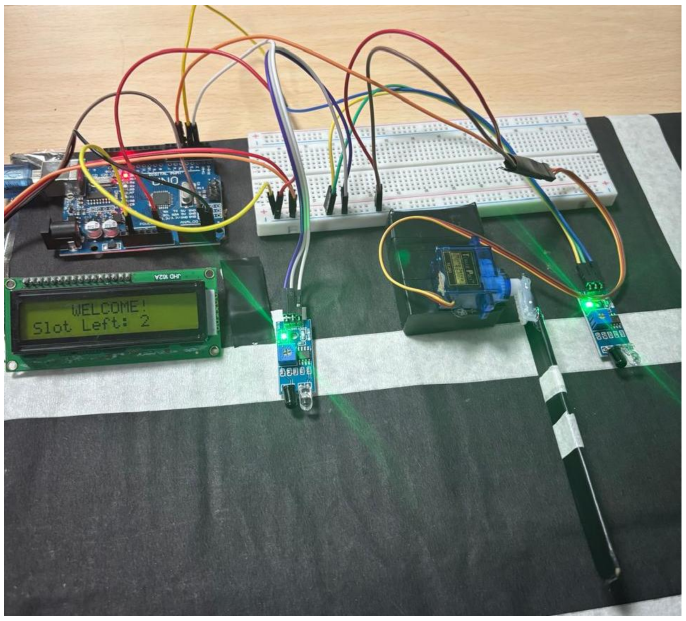


Figure 18: Slot decreases

- While Exiting:
  - Here total number of slot left is 2.

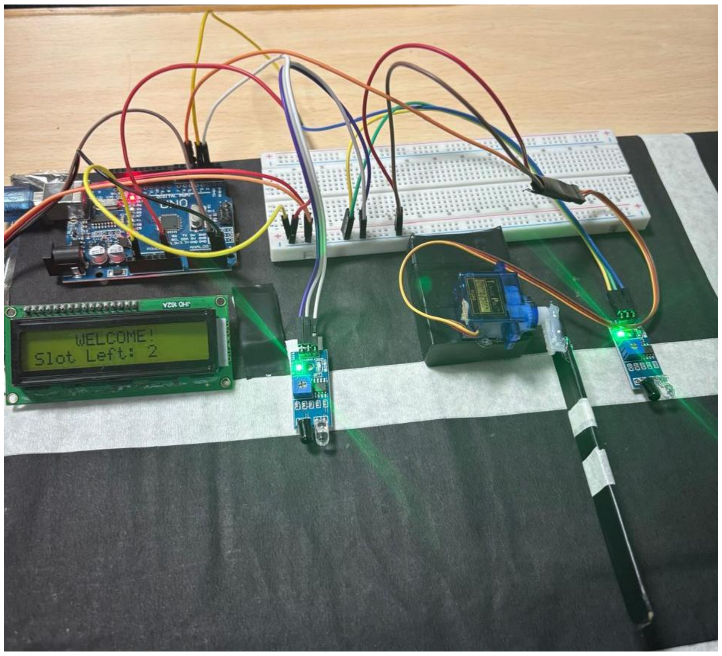


Figure 19: Slot left 2

-

- When a car approaches the sensor, the IR sensor detects it presence, triggering the system to increase the count of available parking slots.

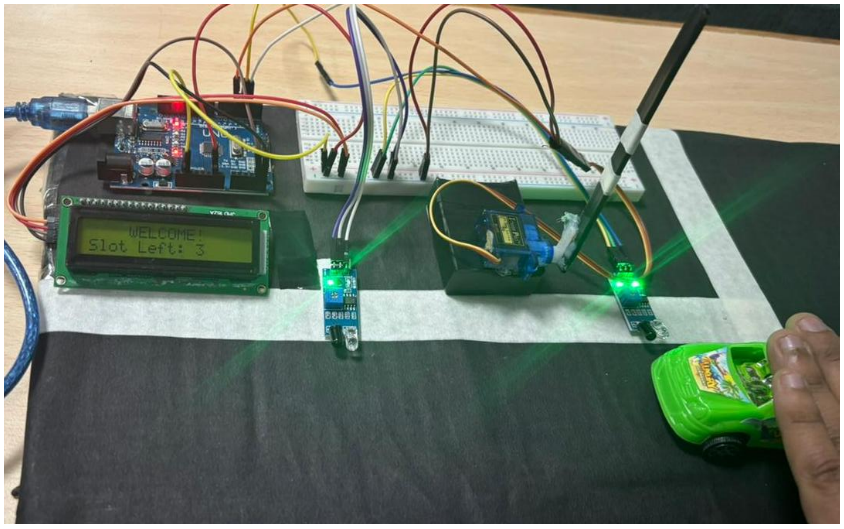


Figure 20: Increase the count of available parking slots.

- Here, the total number of slots increase to 3.

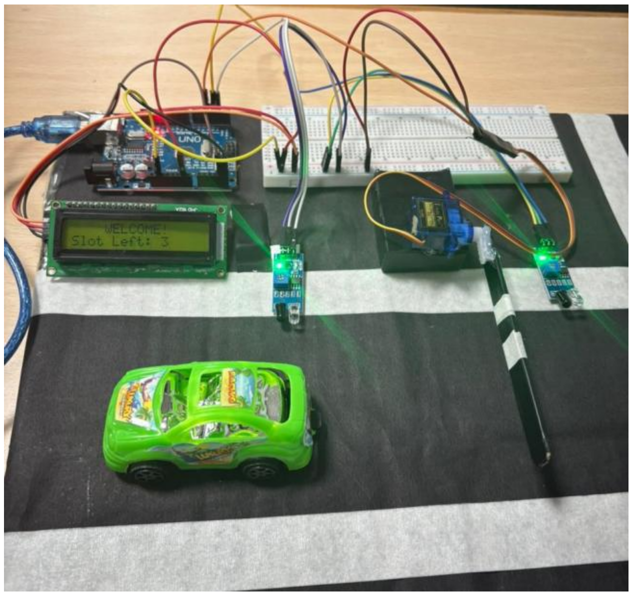


Figure 21: Slot increases to 3

5.1.2. Vehicle Detection Testing at entry point.

Table 2: Test 2 (Vehicle Detection Testing at entry point)

Test	2 (Vehicle detection testing at entry point.)
Objective	Verify if the IR sensors accurately detect the presence of a vehicle at the entry point.
Activity	Position a vehicle near the entry sensor and check whether the system responds by attempting to open the gate.
Expected Outcomes	The IR sensor should detect the presence of a vehicle near the entry point and send a signal to the system to open gate automatically.
Actual Result	The sensor correctly detects the vehicle and triggers the gate to open.
Conclusion	The test is successful, indicating that the IR sensors function correctly in detecting vehicles.

- When vehicles pass through IR sensor at the entry point, the gate responds.

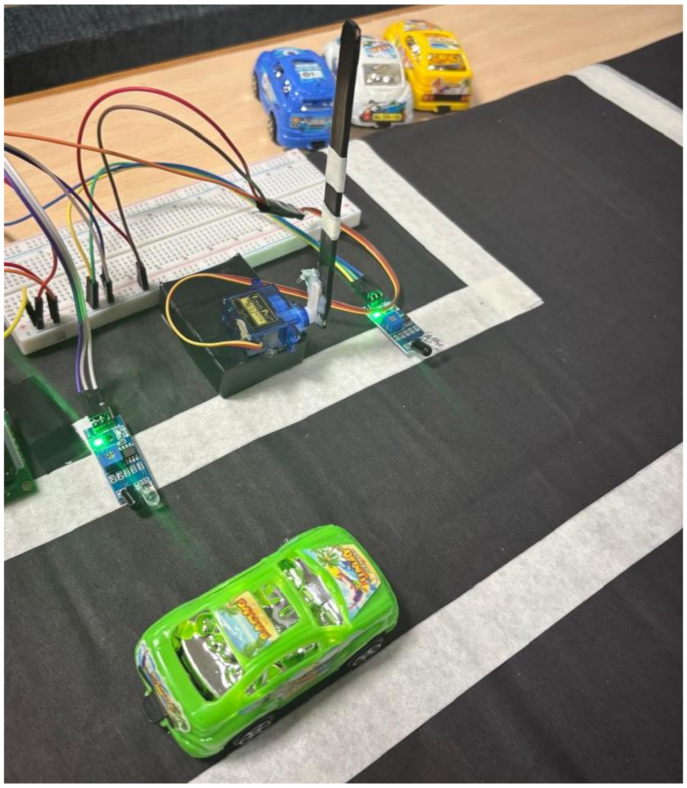


Figure 22: Vehicles pass through IR sensor at the entry point, the gate responds.

5.1.3. Vehicle Detection Testing at exit point.

Table 3: Test 3 (Vehicle Detection Testing at exit point)

Test	3 (Vehicle detection testing at exit point.)
Objective	Verify if the IR sensors accurately detect the presence of a vehicle at the exit point.
Activity	Position a vehicle near the exit sensor and check whether the system responds by attempting to open the gate.
Expected Outcomes	The IR sensor should detect the vehicle accurately, triggering the gate to open. The process should function smoothly and without errors.
Actual Result	The sensor correctly detects the vehicle and triggers the gate to open.

Conclusion	The test is successful, indicating that the IR sensors function correctly in detecting vehicles.

- When vehicles pass through IR sensor at the exit point, the gate responds.

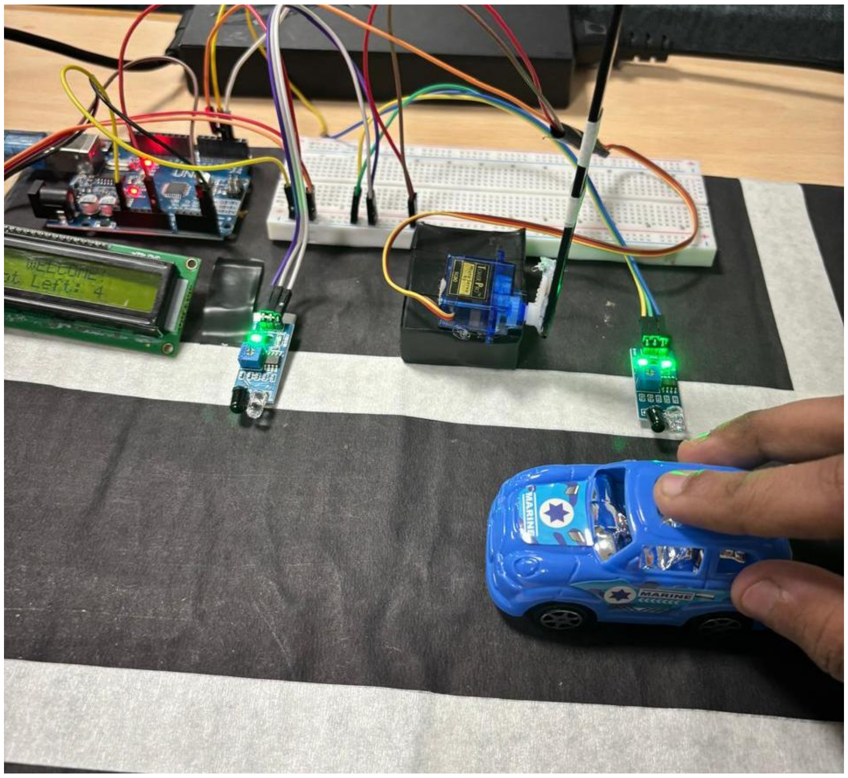


Figure 23: IR sensor at the exit point.

5.1.4. Vehicle detection when parking slots are full (0 Slots left).

Table 4: Test 4 (Vehicle detection when parking slots are full (0 slots left)

Test	4 (Vehicle detection when parking slots are full (0 Slots left).)
Objective	Verify that the system prevents the vehicle from entering when all parking slots are occupied.
Activity	Simulating the scenario where all parking slots are occupied, leaving 0 available slots. A vehicle approaches the entry gate, and the IR sensor detects its presence.

Expected Result	The system checks the parking availability and finds that no slots are left. The gate should remain closed, and the system should display a “Sorry Parking Full” message. The vehicle is not allowed to enter.
Actual Result	The IR sensor detects the vehicle, but the gate closed as no parking slots are available, and a “Sorry Parking Full” message is displayed.
Conclusion	The test is successful, confirming that the system correctly prevents entry when parking slots are full, ensuring the gate remains closed and the appropriate message is displayed.

- The IR sensor detects the vehicle, but the gate closed as no parking slots are available, and a “Sorry Parking Full” message is displayed.

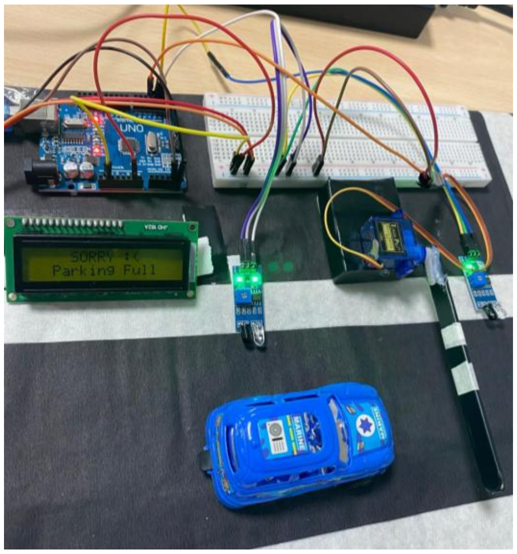


Figure 24: Slots left 0

5.1.5. IR sensor Malfunction During Vehicle Entry.

Table 5: Test 5 (IR sensor Malfunction During Vehicle Entry)

Test	5 (IR sensor Malfunction During Vehicle Entry)
Objective	To verify if the IR sensor and system function as expected during vehicle entry by accurately detecting the vehicle and opening the gate.

Activity	Simulating a scenario where a vehicle approaches the entry point. Check if the IR sensor detects the vehicle and triggers the gate to open.
Expected Result	The IR sensor should detect the vehicle accurately, send a signal to the system, and trigger the gate to open without error. The parking slot count should update correctly on the LCD display.
Actual Result	During testing, the IR sensor did not work properly and failed to detect the vehicle in some cases. As a result, the gate did not open, and the system failed to update the slot count on the LCD display.
Conclusion	The test showed that the IR sensor is not working as expected, leading to issues in detecting vehicles. This fault affects the system ability to operate the gate correctly and update the slot count. To resolve this problem, the faulty IR sensor needs to be replaced with a functional one.

- The IR sensor is not working as expected, leading to issues in detecting vehicles. This fault affects the system ability to operate the gate correctly and update the slot count.

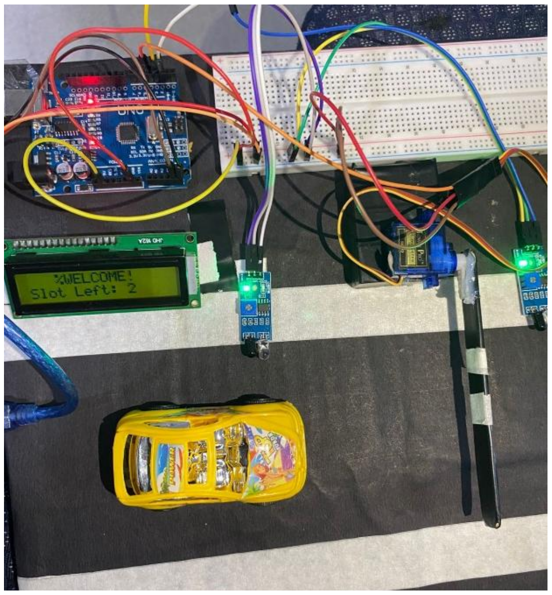


Figure 25: IR issue

6. Future Works

There are several exciting ways to improve and expand the Automated Toll Gate Parking System in the future to make it more efficient, sustainable, and user-friendly. Here are some few ideas:

i. Mobile App Integration:

In the future, a mobile app can be developed that allows users to check real-time parking availability, reserve spots and interact with the system directly from their phones. The app could provide real-time updates on available parking spaces, allow for spot reservations.

ii. Automatic and online Payment System

Currently, the parking system requires manual payment at the gate. In the future, an automated payment system will be implemented to enable seamless and cashless transactions. This system will automatically process payments as users enter or exit the parking space, using methods like digital wallets or OR codes. Additionally, user will be able to pay for parking in advance through an online platform, either by a website or mobile app.

iii. Energy efficiency improvements

The parking system can be made more sustainable by using solar energy. Solar panels could power key components of the system, including sensors and barriers, reducing the need for external power sources and lowering overall energy consumption.

7. Conclusion

This report highlights the development of an Automated Toll Gate Parking System as a solution to parking challenges in busy cities. By minimizing human involvement, the system saves times and reduce errors in managing parking spaces. Automation ensures a more efficient, reliable, and user-friendly experience for drivers and operators, making parking management significantly smoother.

Modern technology like Arduino UNO, IR sensors, servo motors, and LCD displays are integrated to automate the process effectively. Vehicle detection at entry and exit points is handled by IR sensors, the servo motor automates gate operations, and the LCD display shows real-time parking availability. This ensures efficient management and reduces the need of human involvement.

In conclusion, this project demonstrates the practical application of IoT technologies to solve real world problems. It provides a foundation for future enhancements, including mobile app integration, online payment functionality, and improved energy efficiency. The Automated Toll Gate Parking System addresses current parking challenges while opening new possibilities for advancement in automation in other areas.

8. References

Arduino, 2018. *Arduino.* [Online]
Available at: https://www.arduino.cc/en/Guide/Introduction
[Accessed 6 1 2025].

CircuitBread, 2019. *CircuitBread.* [Online]
Available at: https://www.circuitbread.com/ee-faq/what-is-a-breadboard
[Accessed 6 1 2025].

Edirisinghe, H., 2023. *Medium.* [Online]
Available at: https://medium.com/@hasaliedirisinghe/user-guide-to-draw-io-6a1a4d7b5d33
[Accessed 6 1 2025].

Electrical4U, 2024. *Electrical4U.* [Online]
Available at: https://www.electrical4u.com/what-is-servo-motor/
[Accessed 6 1 2025].

geeksforgeeks, 2021. *geeksforgeeks.* [Online]
Available at: https://www.geeksforgeeks.org/introduction-to-microsoft-word/
[Accessed 6 1 2025].

MAKERLAB, 2025. *MAKERLAB.* [Online]
Available at: https://www.makerlab-electronics.com/products/ir-proximity-sensor
[Accessed 6 1 2025].

Store, 2024. *Arduino UNO.* [Online]
Available at: https://store.arduino.cc/products/arduino-uno-rev3?srsltid=AfmBOopsJW4gf45jeH0taa7t_3bkwXxmHteS3aDIhNHzLb0QSg_PfxmN

sunfounder, 2024. *sunfounder.* [Online]
Available at: https://docs.sunfounder.com/projects/ultimate-sensor-kit/en/latest/components_basic/21-component_i2c_lcd1602.html
[Accessed 6 1 2025].

Wiltronics, 2022. *Wiltronics.* [Online]
Available at: https://www.wiltronics.com.au/wiltronics-knowledge-base/what-are-jumper-wires/
[Accessed 6 1 2025].

9. Appendix

9.1. Requirement Analysis

9.1.1 Hardware Requirements

i. Arduino UNO:

The Arduino UNO is an easy-to-use microcontroller board built on the ATmega328P, renowned for its simplicity and adaptability. It offers 14 digital I/O pins, 6 analog inputs, USB connectivity, and is programmed through the Arduino IDE, making it perfect for various projects and prototypes. (Arduino, 2018)

Figure 26: Arduino UNO

ii. Breadboard:

A breadboard is a simple tool used for testing and building circuits without needing to solder. It has many tiny holes that hold components and wires securely in place. Inside the breadboard, metal strips connect these holes, allowing electricity to flow through the circuits. It's commonly used for quick and easy prototyping of circuit designs. (CircuitBread, 2019)

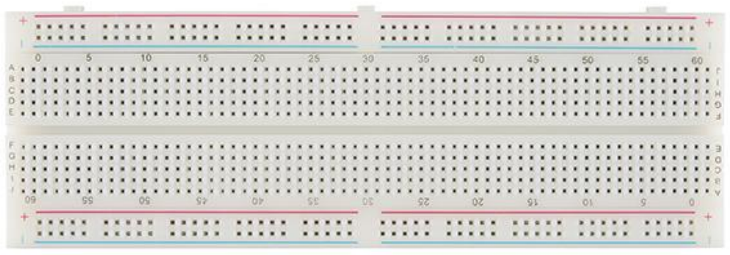


Figure 27: Breadboard

iii. IR Proximity Sensor:

The IR Proximity Sensor is a flexible infrared sensor suitable for various applications such as obstacle detection, color detection, fire detection, line sensing, and more. It has an indoor detection range of 40-50 cm and an outdoor detection range of 15-20 cm. (MAKERLAB, 2025)

Figure 28: IR Proximity Sensor

iv. Servo Motor:

A servomotor is a type of motor that can control the position, speed, and movement precisely, whether it's moving in a straight line or rotating. To operate a servomotor, it needs a special controller, which is often a separate piece of equipment specifically designed for this purpose, known as a servomotor controller. (Electrical4U, 2024)

Figure 29: Servo Motor

v. Jumper Wires:

Jumper wires are small metal wires used to connect different parts of a circuit. They allow you to complete or break a connection between two points on a circuit board without needing to solder. These wires are also called two-wire cables because they typically have two ends that you can use to link components together. (Wiltronics, 2022)

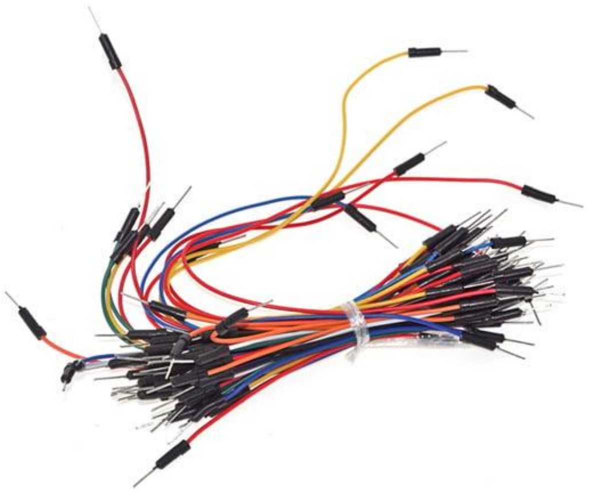


Figure 30: Jumper Wires

vi. LCD Display with IC2 Module:

In LCD screen I2C module is attached to the back. This module expands the Arduino’s input/output ports and allows the LCD communication using the I2C protocol. (sunfounder, 2024)

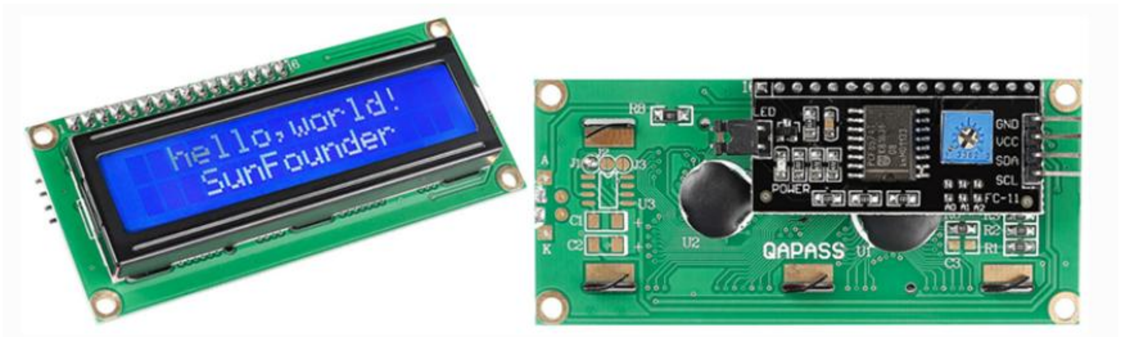
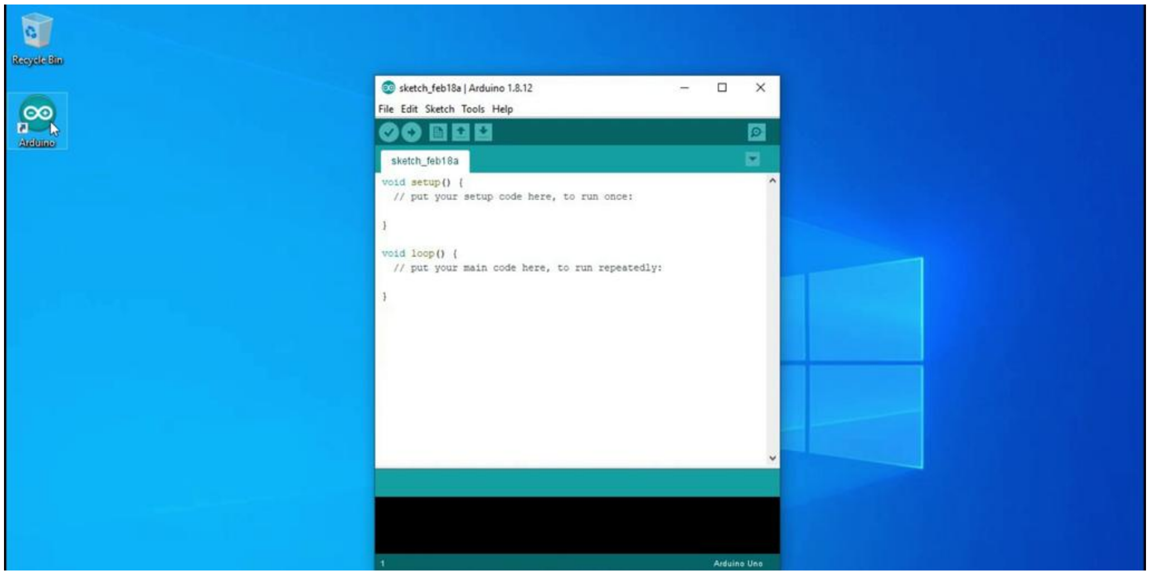


Figure 31: LCD display with IC2 Module

9.2.2. Software requirements

i. Arduino IDE:

The Arduino IDE (Integrated Development Environment) is an open-source software platform designed to simplify building electronic projects. It works seamlessly with Arduino hardware, offering an accessible way for users, including beginners, to create and program interactive devices. The IDE allows users to write code in the Arduino programming language (based on C/C++) and upload it to an Arduino board.



```
void setup() {
  // put your setup code here, to run once:

}

void loop() {
  // put your main code here, to run repeatedly:

}
```

Figure 32: Arduino IDE

ii. Draw.io:

Draw.io is a tool used for creating diagrams and charts. It provides various layout options, allowing users to design custom layouts according to their requirements. The tool includes a wide range of shapes and graphics, enabling users to create unique diagrams. In this coursework, Draw.io is used for developing block diagrams and flowcharts. (Edirisinghe, 2023)

Figure 33: Draw.io

iii. MS Word:

MS Word is a word processing software developed by Microsoft in 1983. It offers a huge number of features, making it ideal for creating professional-quality documents such as letters, reports, and resumes. It also allows users to edit and modify documents easily. The latest version is MS Word 2019. In this coursework, MS Word is utilized for documentation and report writing. (geeksforgeeks, 2021)

Figure 34: MS word

iv. Tinkercad:

Tinkercad is an easy-to-use online platform for designing and simulating electronic circuits. It allows users to create virtual circuits with components like Arduino boards, sensors, and motors, and test their functionality through simulations. This makes it ideal for identifying and fixing issues before physical implementation. (tinkercad, 2025)

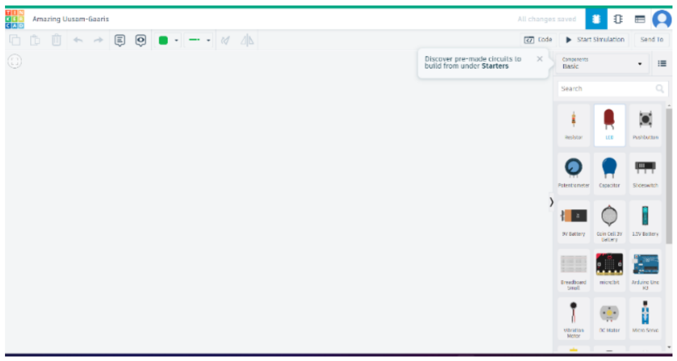


Figure 35: Tinkercad

9.3. Code:

```
        // Created by Simple Circuits


#include <Wire.h>

#include <LiquidCrystal_I2C.h>

#include <Servo.h>


LiquidCrystal_I2C lcd(0x27, 16, 2);

Servo myservo;


int IR1 = 2;  // IR sensor for car entry

int IR2 = 3;  // IR sensor for car exit

int Slot = 4; // Total number of parking slots


void setup() {

  Serial.begin(9600);

  lcd.init();

  lcd.backlight();

  pinMode(IR1, INPUT);

  pinMode(IR2, INPUT);

  myservo.attach(4);

  myservo.write(100); // Close the barrier initially
  // Display welcome message
  lcd.setCursor(0, 0);
  lcd.print("     ARDUINO    ");
  lcd.setCursor(0, 1);
  lcd.print(" PARKING SYSTEM ");
  delay(2000);
  lcd.clear();
}

void loop() {
  static unsigned long lastActionTime = 0; // To track time for actions
  static bool barrierOpen = false;         // Track barrier state

  // Entry of a car
  if (digitalRead(IR1) == LOW && !barrierOpen) {
    if (Slot > 0) {
      myservo.write(0);    // Open the barrier
      barrierOpen = true;  // Mark the barrier as open
      Slot--;              // Decrement available slots
      updateDisplay();
      lastActionTime = millis(); // Record the action time
    } else {
      lcd.setCursor(0, 0);
      lcd.print("    SORRY :(    ");
      lcd.setCursor(0, 1);
      lcd.print("  Parking Full  ");
      delay(2000); // Wait before clearing
      lcd.clear();
      updateDisplay();
    }
  }

  // Exit of a car
  if (digitalRead(IR2) == LOW && !barrierOpen) {
    myservo.write(0);    // Open the barrier
    barrierOpen = true;  // Mark the barrier as open
    Slot++;              // Increment available slots
    updateDisplay();
    lastActionTime = millis(); // Record the action time
  }

  // Close the barrier after a delay
  if (barrierOpen && millis() - lastActionTime >= 3000) {
    myservo.write(100);  // Close the barrier
    barrierOpen = false; // Mark the barrier as closed
  }


  // Update the LCD display

  updateDisplay();

}


void updateDisplay() {

  lcd.setCursor(0, 0);

  lcd.print("    WELCOME!    ");

  lcd.setCursor(0, 1);

  lcd.print("Slot Left: ");

  lcd.print(Slot);

  lcd.print("   "); // Clear leftover characters

}
```