Module Code & Module Title
CS5068NI– Cloud Computing & IoT

Toll Gate Parking System

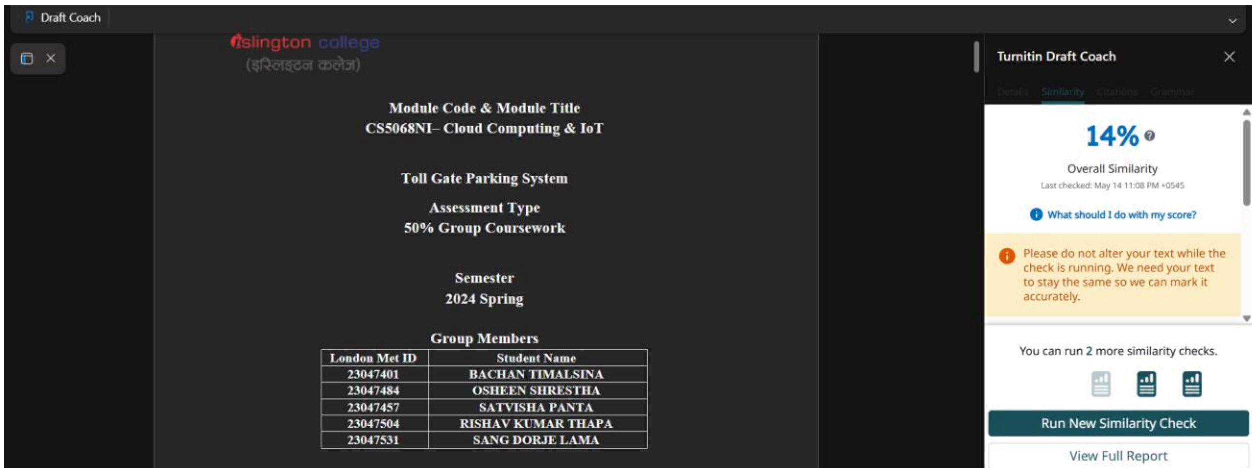
Assessment Type
50% Group Coursework

Semester
2024 Spring

Group Members

London Met ID	Student Name
23047401	BACHAN TIMALSINA
23047484	OSHEEN SHRESTHA
23047457	SATVISHA PANTA
23047504	RISHAV KUMAR THAPA
23047531	SANG DORJE LAMA

Assignment Due Date: 15th May,2025
Assignment Submission Date: 15th May,2025
Submitted to: Mr. Sugat Man Shakya
Word Count: 3857



Module Code & Module Title
CS5068NI– Cloud Computing & IoT

Toll Gate Parking System

Assessment Type
50% Group Coursework

Semester
2024 Spring

Group Members

London Met ID	Student Name
23047401	BACHAN TIMALSINA
23047484	OSHEEN SHRESTHA
23047457	SATVISHA PANTA
23047504	RISHAV KUMAR THAPA
23047531	SANG DORJE LAMA

Turnitin Draft Coach

14%
Overall Similarity
Last checked: May 14 11:08 PM +0545

What should I do with my score?

Please do not alter your text while the check is running. We need your text to stay the same so we can mark it accurately.

You can run 2 more similarity checks.

Acknowledgement

Firstly, we wish to extend our thanks to our module leader Mr. Sugat Man Shakya and our class tutor Mr. Bishnu Pandey sir for granting us this wonderful chance to engage in this IoT project. Additionally, we appreciate our parents for their encouragement and financial support for our project. Lastly, we are extremely grateful to the senior students for providing us with valuable perspectives on effective teamwork.

Abstract

Getting parking space especially in modern fully developed cities is becoming a problem because of the increase in traffic jams and unnecessary expenditure of time. This paper presents a detailed account of implementing a cheapest and quite effective Smart Parking System by employing Arduino Uno and simple electronics parts only. This is by the use of infrared proximity sensors that are able to identify either the existence or the non-existence of automobiles in particular slots. The I2C module of the LCD display also shows the current slots availability therefore indicating whether the gate is open or closed and the use of servo motor hence controlling the opening or closing of the gate is based on the availability of slots. This paper further proposes a system that envisages minimizing the use of personnel, the efficiency of parking space, and the satisfaction of the user. This project shows how complex applications of simple embedded systems could be implemented and how it can be built with scalability for future enhancements like mobile interface, and better ways of cloud data handling.

Table of Contents

Table of Figures

Table of Tables

1. Introduction

The Internet of Things (IoT) transforms industry operations because devices link through a system of sensors and software and networks. The IoT operates via devices that connect using sensors and software linked through networks. These systems operate for solving particular problems. Commercial IoT systems solve everyday problems and produce efficient solutions which enhance safety and make mundane activities simpler to execute. By using the combination of microcontrollers with sensors together with automated systems makes possible IoT solutions (Yansar, 2025).

This technology gets deployed efficiently to enhance transportation systems healthcare institutions along with urban management entities. The “Automated Toll Gate Parking System” project serves to address the parking congestion challenges that exist in urban areas. of parking congestion in urban areas. The parking space exists with few available locations while parking management operates inefficiently. Regular drivers encounter extended delays as well as traffic congestion at numerous driving checkpoints.

This system is a better parking management solution becomes possible by implementing automated controls and minimizing human intervention along with real-time parking spot information provision. This system removes manual interaction so drivers get live notifications about open parking spaces and spots.

1.1. Current scenario

As one of the most important and increasingly complex problems of the society, parking management is one of the most burning issues in Nepal and particularly in the capital city of Kathmandu. It is recorded that there are currently 1.75 million registered vehicles across the Kathmandu valley in the year 2022 and it is a continuously increasing ratio every year as per the records available with Kathmandu Valley Traffic Police Office. This has been sparked by the many vehicles that are on the road, making it hard for drivers to search for a parking space to park. Most drivers become impatient, and therefore, they park their cars on the roadsides or side-walks, which results in congestion and worsens the situation further. These challenges are greatly compounded by the failure of proper management of parking areas and this a pointer to the need to address the issue.

Figure 1 Current parking scenario

1.2. Problem Statement and Project as a solution

Parking management is also common in many cities and urban area, particularly in the Kathmandu valley, where the trend of using automobiles has drastically increased. This is due to the fact that drivers get frustrated and may spend a lot of time seeking for a parking place. And effective parking management is also missing and results to parking of the cars along the edges that eventually comes with other chain repercussions such as traffic jams. All these problems originate from the fact that their solution has not been devised yet, and thus, this project seeks to put an end to all such issues by realizing an Automated Toll Gate Parking System. The system will involve the use of a display screen which gives an update of the available parking space. It will only open when a parking space is available and hence assisting the drivers find a parking space easily.

The developed system does not involve any need for persons, and as such proves to be quite efficient as compared to the other existing parking systems. In this way we shall be able to cut on the traffic incidences, time and arrival of products and services for overall parking management experience.

Figure 2 Project as a solution

1.3. Aim and Objectives

1.3.1 Aim

The purpose of this project is the creation of an Automated Toll Gate System capable of detecting which parking slots are available or not, to grant entry to vehicles when there are parking spaces as well as stimulate a buzzer whenever there is no space.

1.3.2 Objectives

The following are the set goals of the Automates Toll Gate Parking System:

i. The one that will enable monitoring of parking slots available in real time in order to ease identification of the available slots.

ii. Develop technology that will give directions on available parking space in real-time hence span a shorter time searching for a vacant parking lot.

iii. Reduce operational costs by parking space automation and ensuring less intervention of workers.

iv. Increased efficiency of the parking management process and minimization of human error using the software.

2. Background

2.1. System overview:

The Automated Toll Gate Parking System is designed to efficiently manage parking operations with minimal human intervention. It incorporates several hardware components such as IR sensors, an LCD display, servo motors and an Arduino UNO to ensure smooth functionality. The IR sensors play a vital role in detecting vehicles at both entry and exit points. Upon detection, the system updates the availability of parking spaces in real time on the LCD display and automatically operates the gate mechanism accordingly.

In situations where the parking lot reaches full capacity, the system displays a message indicating that parking is full and restricts further entry. By automating these functions, the system minimizes human error, improves time efficiency and enhances the overall user experience. This implementation demonstrates the effective use of IoT and automation technologies in addressing practical challenges in parking management.

2.2 Design Diagram

2.2.1 Block Diagram

A block diagram is a specialized high general flowchart in engineering. It is used for the design of newly developed systems and for the description and improvement of already existing systems. Its structure provides a general view of major engineered system components, while identifying key participants in the process involved and some of the most important working relationships (indeed, 2025).

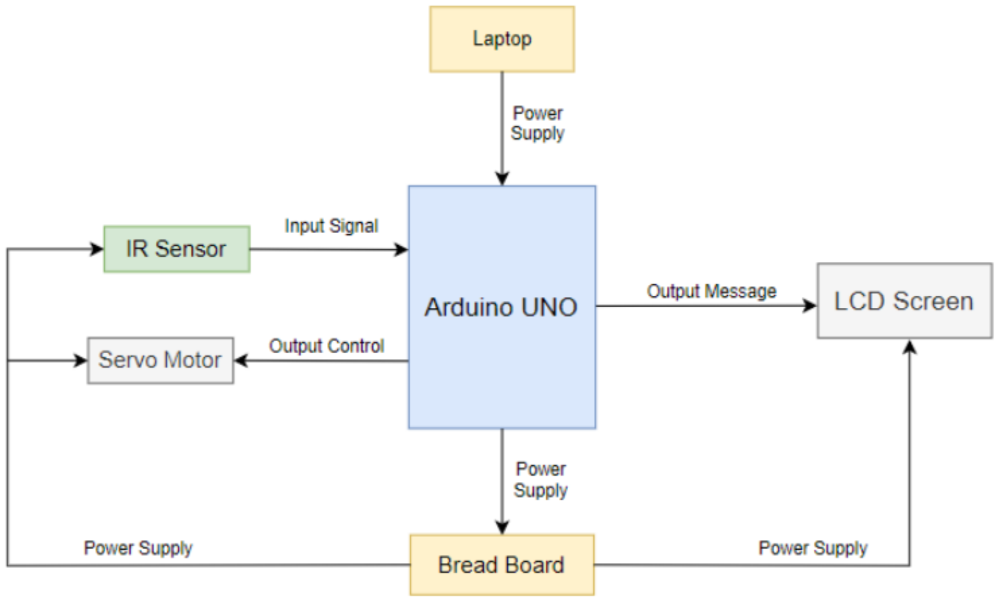


Figure 3 Block Diagram

2.2.2 System Architecture

System architecture is the organization of a given system and how it is put together including its components, their relation and the principles and guidelines associated with the design and evolution of the components of the system. It gives a high-level perspective on the system which describes the key elements of the system, their responsibilities, and the ways they interact with other systems. System architecture pays attention to system's general design philosophy, parcelling out functionality to distinct components and combining the components together (Kang, 2023).

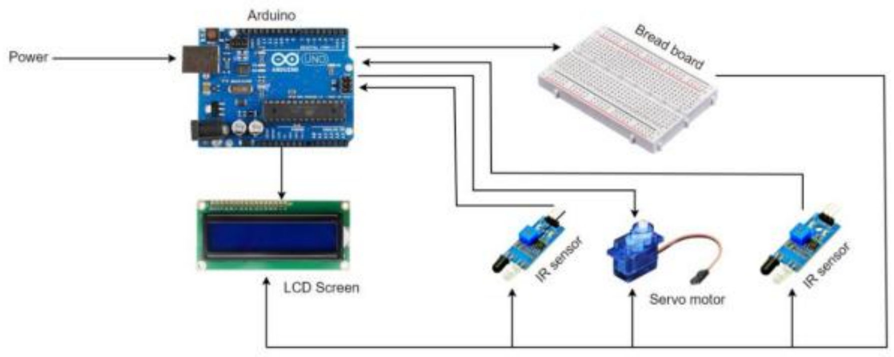


Figure 4 System Architecture

2.2.3 Circuit Diagram

A circuit diagram, electrical schematic or an electronic schematic is a structural picture of an electrical circuit. It is a pictorial illustration of how may electrical components as well as wires are interconnected to create a functional circuit. Circuit diagrams are an important element of electronics and electrical engineering since what they do is allow engineers, technicians, as well as hobbyists to understand, design, and fix electronic circuits (Kohli, 2023).

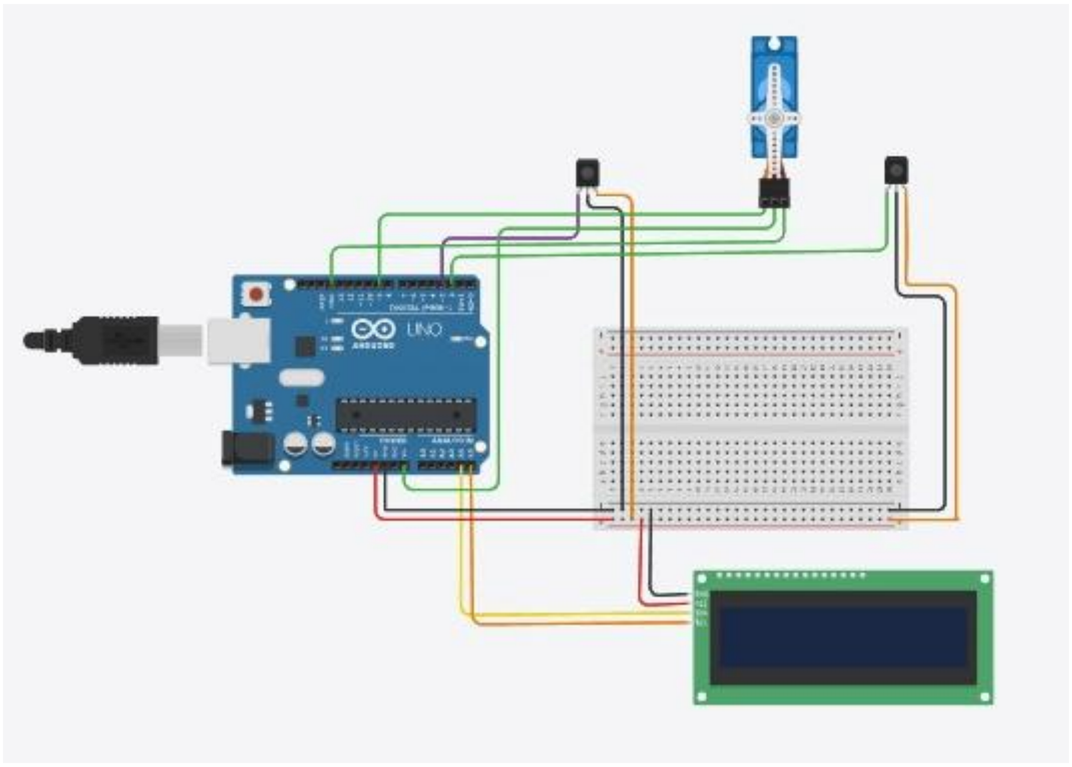


Figure 5 Circuit Diagram

2.2.4 Schematic Diagram

A schematic is a diagram whereby something is depicted in a simple manner through symbols. Schematic diagram, also known as schematic, is a picture representing the parts of a process, device, or other object with abstract, generally standard symbols and lines. Schematic diagrams are only provided, which demonstrates the main elements of a system, whereas some of the elements of the diagram can be also exaggerated or presented to make the system more comprehensible (Lim, 2024).

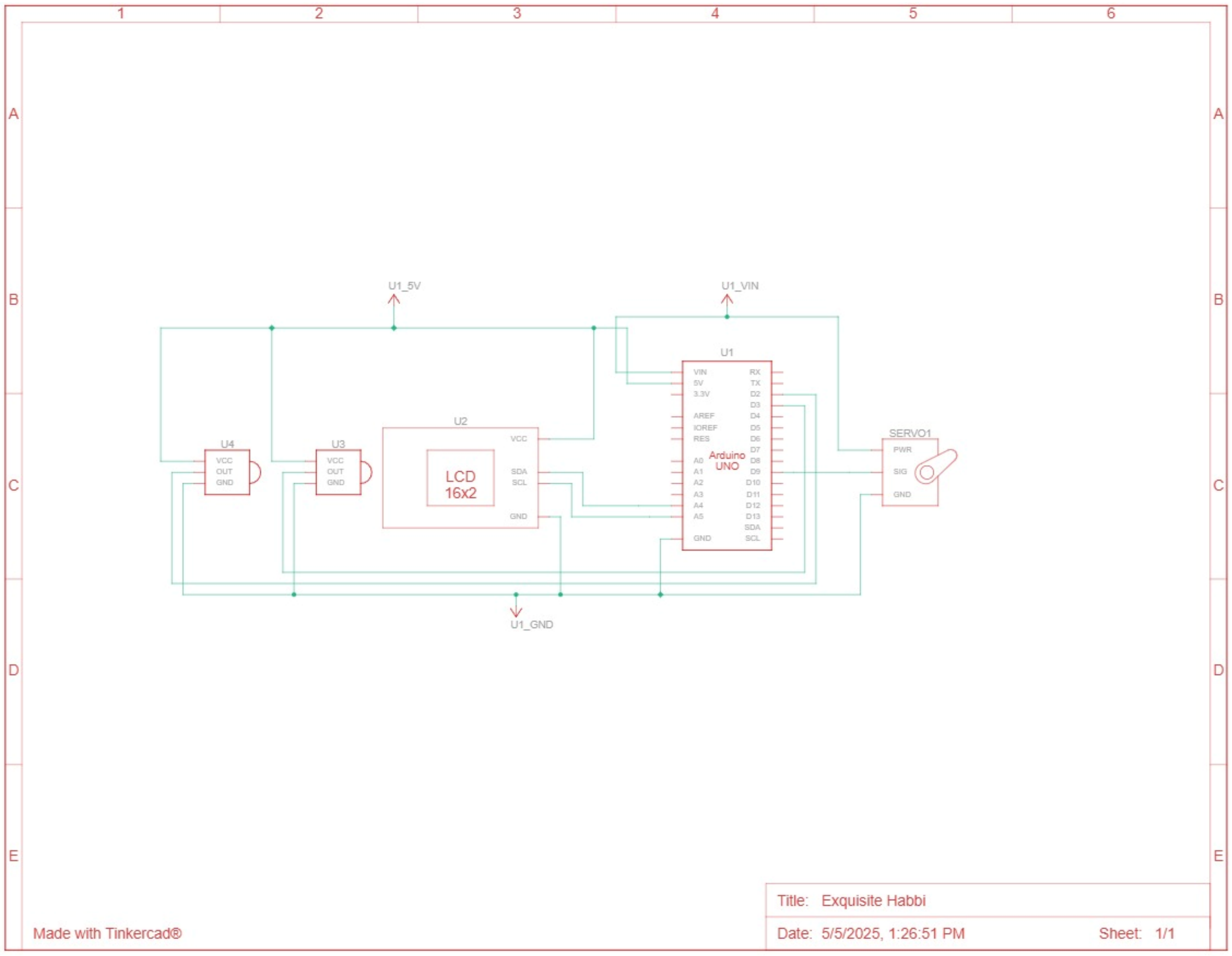


Figure 6 Schematic Diagram

2.2.5 Flowchart

This flowchart demonstrates the sequential process of vehicle entrance, slot availability and exit. This employs symbols like ovals for start and end points, rectangles for processes, parallelograms for input/output, diamonds for decision points, and arrow to represent the flow of activities.

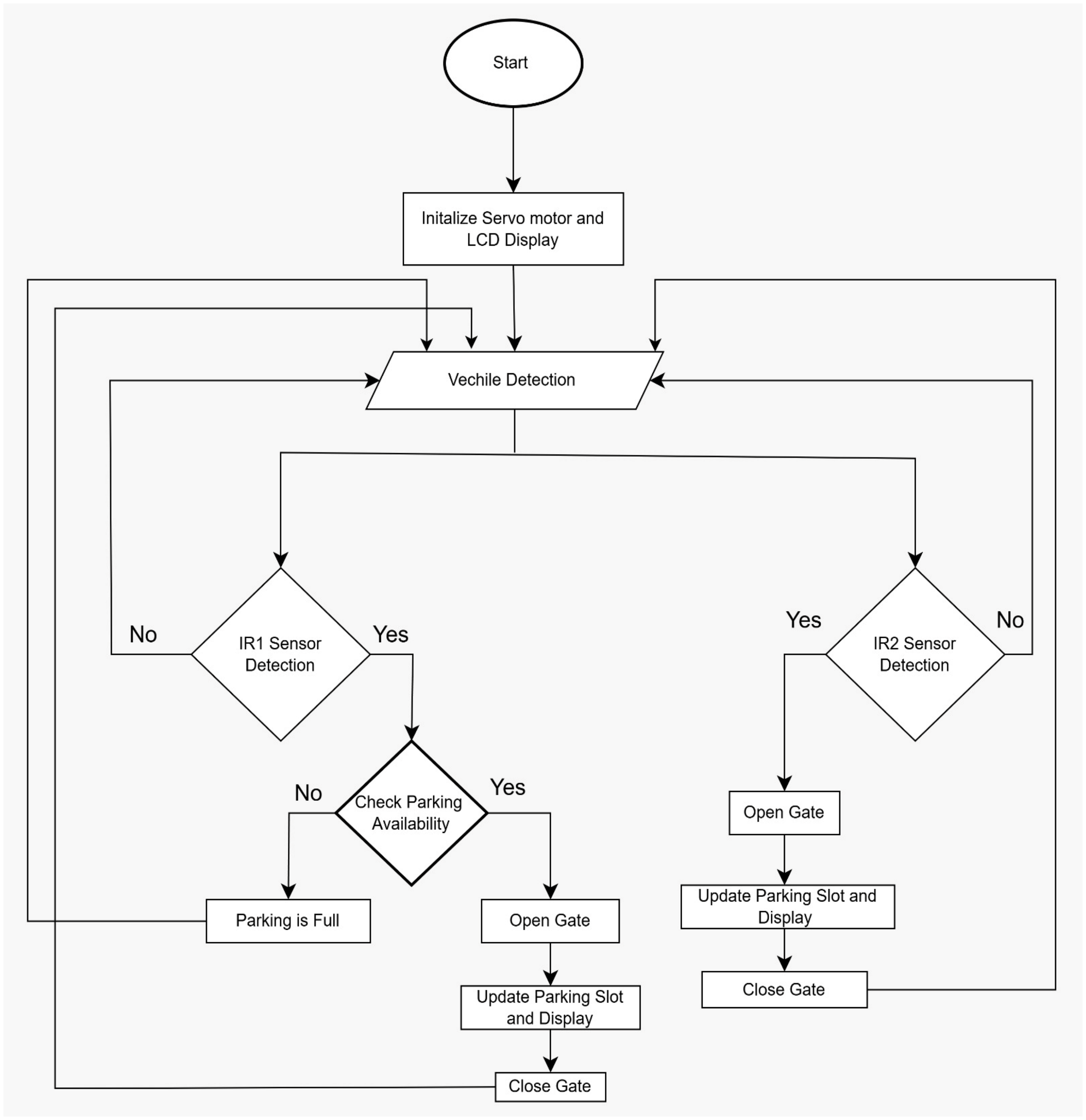


Figure 7 Flowchart

2.3 Requirement Analysis

2.3.1 Hardware Components

1. Arduino Uno

Figure 8 Arduino Uno

The Arduino Uno is one of the microcontrollers that can be used to control the power and the operation of the sensors, as well as the capacity to process information in different structures. In a parking management system, it has interactions with sensors such as IR to help in sensing opening spaces of parking to make parking easier and faster (Arduino, 2018).

2. IR Sensor

Figure 9 IR Sensor

Vehicle sensors are also used because they can efficiently identify open space through the infrared light that they apply on the ground. In parking management systems, they assist to enable the driver to gain an easy way of getting to the free space and therefore make the whole parking more efficient (robu.in, 2020).

3. Jumper Wires

Figure 10 Jumper Wire

Jumper wires are a basic cable that connects two or more points in a circuit through and from one location to another. In the context of parking management system, they assist in integrating the sensors and the controllers and tending to all the purposes.

4. Breadboard

Figure 11 Bread Board

Breadboard is quite useful gadget, which allows creating and trying circuits without having to use solder – it is simply inserted into the breadboard. Thus, in the condition of parking management, it will be convenient to connect a number of sensors and controllers to organize the work, experiment with their placement and connections before finalizing the wiring (Anon., science buddies).

5. LCD Display

Figure 12 LCD Display

LCD display exhibits plain text and is of the 2x16 type, meaning that it can display two lines of 16 characters each, which is convenient if one wants to observe some particular data. In an LPR application for example in a parking management system, it can show messages such as the number of parking spaces available or guide the direction for drivers.

6. Servo Moter

Figure 13 Servo Motor

A servo motor is a motor capable of operating up to an angle to provide precise control in some specific applications. In a parking management system, it is capable of performing an operation such as opening or closing of the gates.

7. USB cable and power supply

Figure 14 USB cable and power supply

Arduino boards can operate satisfactorily on power that is available from the USB port. It provides 5V DC voltage and can be sourced from the port from a PC, wall socket adapter or portable power bank.

2.3.2 Software Components

i. Arduino IDE:

Arduino IDE is such one program which is quite user-friendly to enable the user to write program and upload the same to any of the Arduino boards. Overall, it is useful for first time users and for professionals; this makes it easy for anyone to use to create master pieces.

Figure 15 Arduino IDE

ii. Draw.io

Draw.io is a diagram and chart creating tool. It supports different types of layouts, and users can create their own layout as needed. It consists of a large collection of shapes and graphics, which helps users create their own diagrams. In this coursework, the Draw.io tool is utilized for creating block diagrams and flowcharts (computerhop, 2024).

Figure 16 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, 2025).

Figure 17 MS Word

iv. Tinkercad

Tinker cad 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 (IGI GLOBAL, 2025).

Figure 18 Tinkercad

3. Development

The development section includes the entire process of building the IoT-based Car Parking System from design to implementation.

Step 1: Design and Planning

Electronic design and circuit production occurred through the electronic simulation software Tinker CAD. This project needed main hardware items that consisted of Arduino Uno together with Two IR sensors and Servo motor and LCD display with I2C module and Breadboard and Jumper wires. The main distribution of power (5V and GND) occurred between the Arduino board and the breadboard after which IR Sensor 1 received placement near the gate as a detector of incoming traffic while IR Sensor 2 operated within the parking space to detect vehicles existing. The motor activated by Servo control moved its shaft to achieve 90-degree rotation for gate opening and maintained a 0-degree position for closing the gate. Through the I2C module we linked the LCD Display to reduce the number of pins and show available parking slot counts. The implementation included precise jumper wire connections which established proper electrical contacts for power supply and data signals.

We transferred the Tinker CAD simulation logic into a physical breadboard setup with identical parameters that we had tested before. The system became simpler to program and fix any issues throughout physical implementation.

Step 2: Resource Collection

The resources for this project were obtained from the college. An application was submitted to the **IT Resource Department** to receive available items such as Arduino Uno, Breadboard, Jumper wires, IR Sensors, Servo Motor **LCD Display** and **I2C Module**. Through this stepwise development process, we were able to clearly understand the system's functionality and implement the hardware and software successfully.

Step 3: System Development

Phase 1: Giving power to breadboard from Arduino

During the beginning of system development, we established power distribution connections on the breadboard. The Arduino Uno 5V pin received connection to the breadboard positive rail while its GND (ground) pin received connection to the negative rail. All components attached to the breadboard obtained a reliable power supply from the Arduino through this power distribution setup. Application of the base power distribution system became essential to enable further integration with other elements in the circuit.

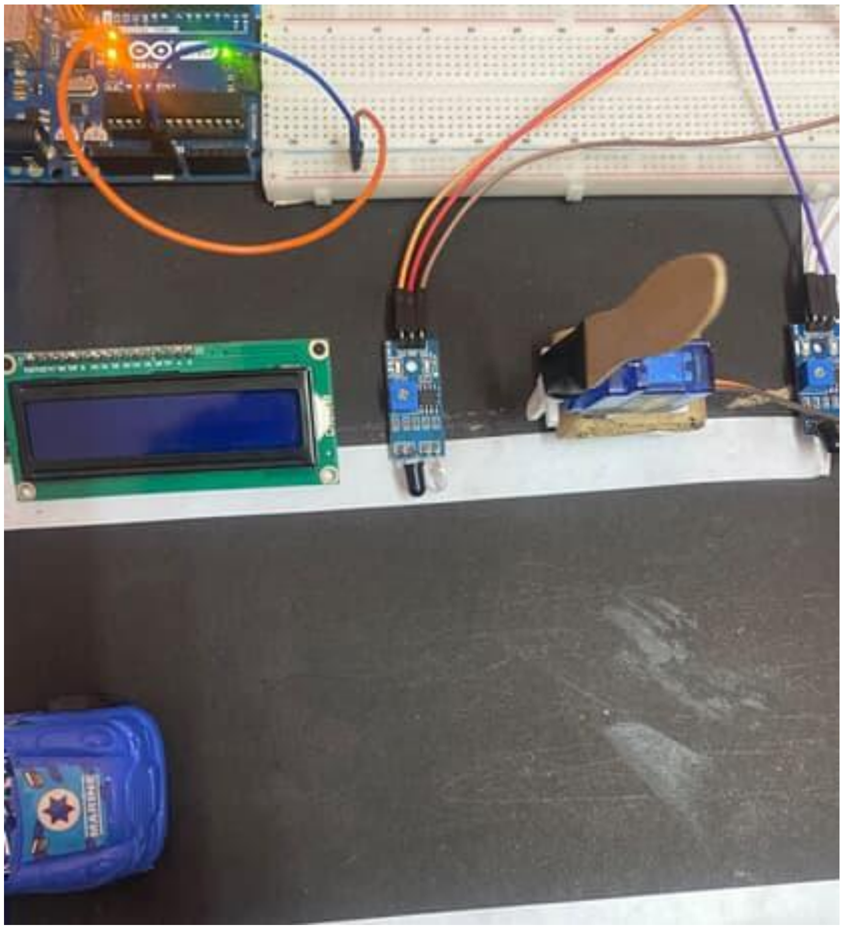


Figure 19 Giving power to breadboard from Arduino

Phase 2: Connecting all input device

The IR proximity sensor was integrated to Arduino through a connection with the breadboard during this phase. The GND pin of the sensor received power from the negative rail of the breadboard through the VCC pin which was connected to the positive rail. An LED signal flowed from the OUT pin to Arduino digital pin D2.

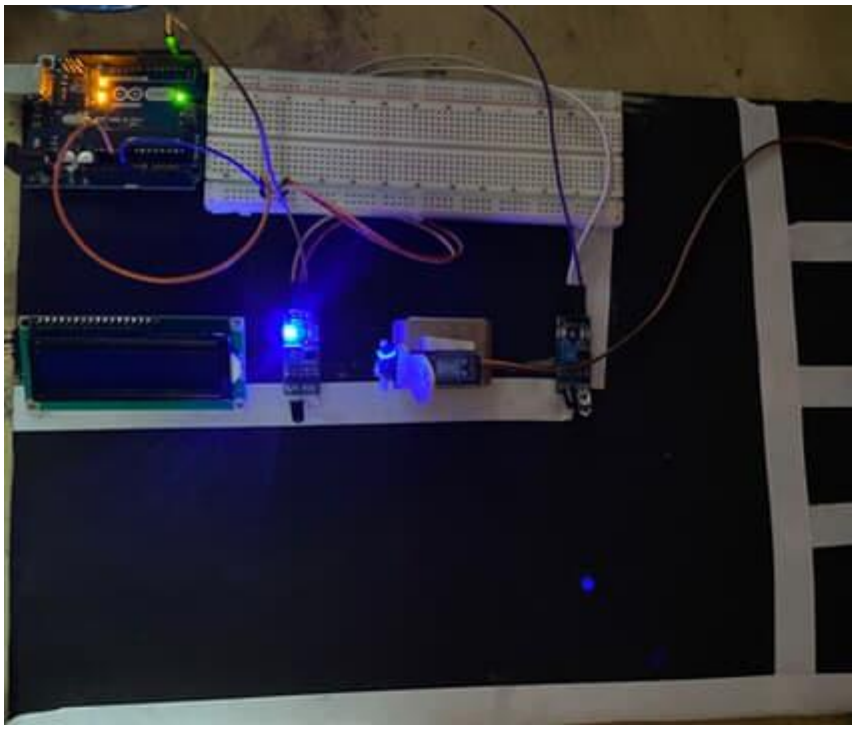


Figure 20 IR sensor setup with Arduino: OUT to D2, VCC and GND via breadboard

We established the second IR sensor in the final phase of the project. The GND and VCC pins of the sensor received their connections to the negative and positive rails similarly to past sensors. A signal port from the OUT pin reached digital pin D3 in the Arduino for monitoring purposes.

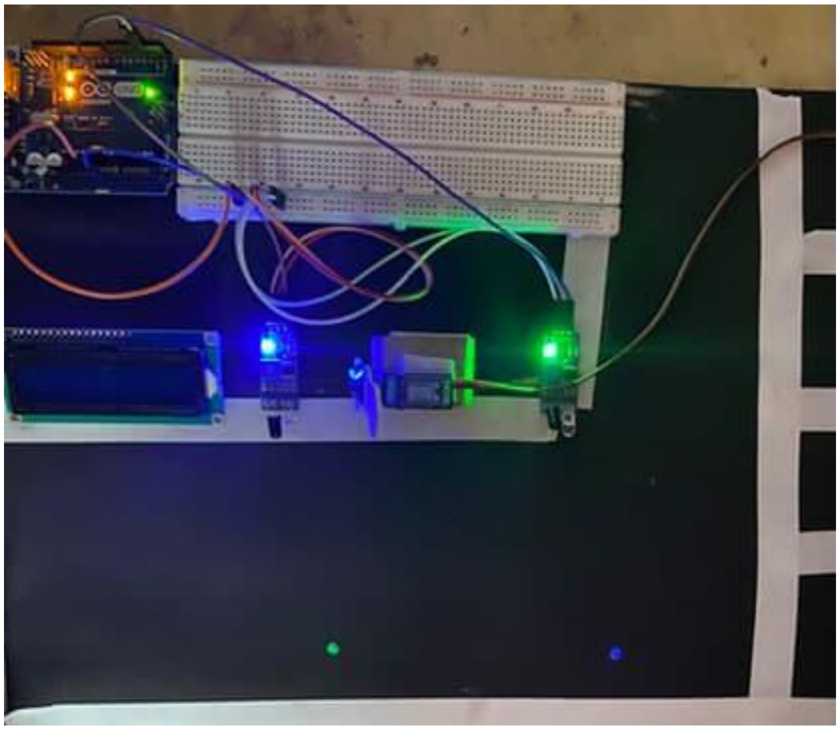


Figure 21 Second IR sensor connected: OUT to D3, powered via breadboard rails

Phase 3: Connecting all output devices

The LCD display received its connection to the Arduino system through the second phase. The LCD required its GND pin connected to the negative rail of the breadboard while the VCC pin required connection to the positive rail for power supply. The A4 pin of the Arduino served as the interface for data communication with the SDA pin of the LCD display and the SCL pin linked to the A5 pin. An effective configuration enabled the LCD to obtain power supply and serial data signals from the Arduino.

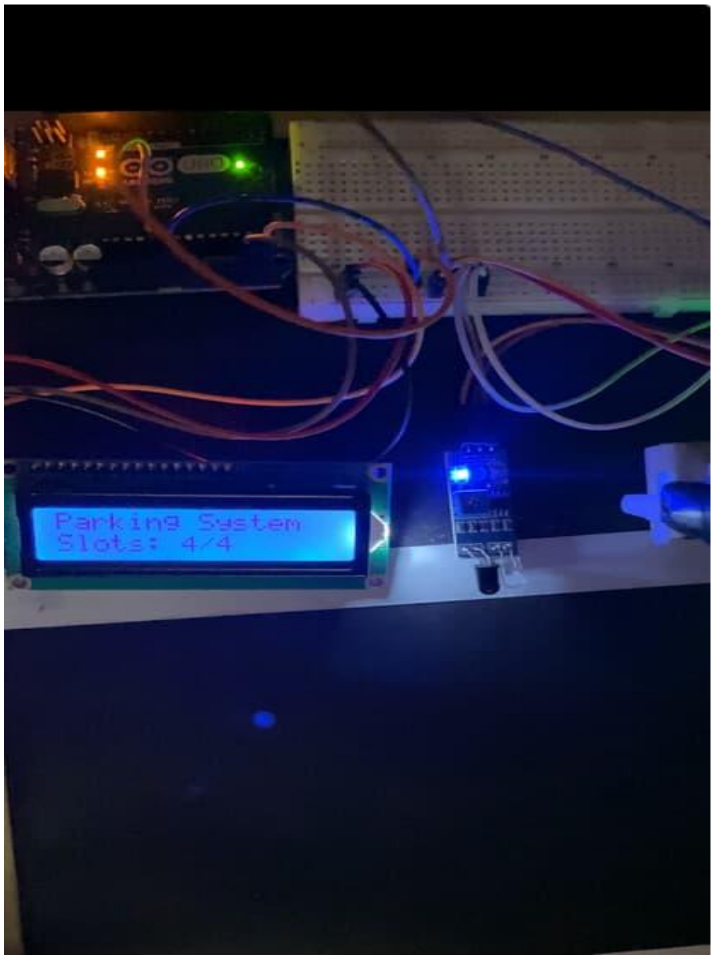


Figure 22 LCD connected: GND/VCC to rails, SDA to A4, SCL to A5 for data and power

We then established a connection for the servo motor. The GND wire of the servo motor attached to the GND in Arduino and the VCC wire to the 5v on the Arduino. The Arduino received control signals through digital pin D9 to which the signal pin was connected.

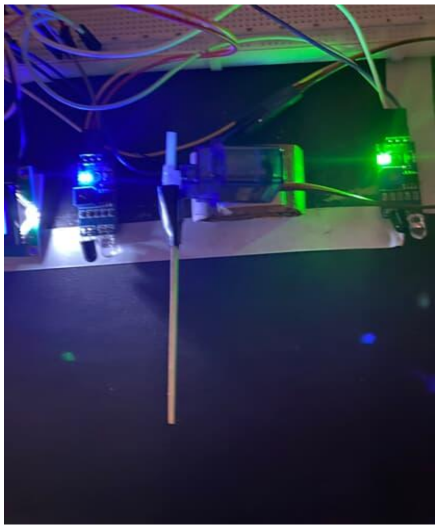
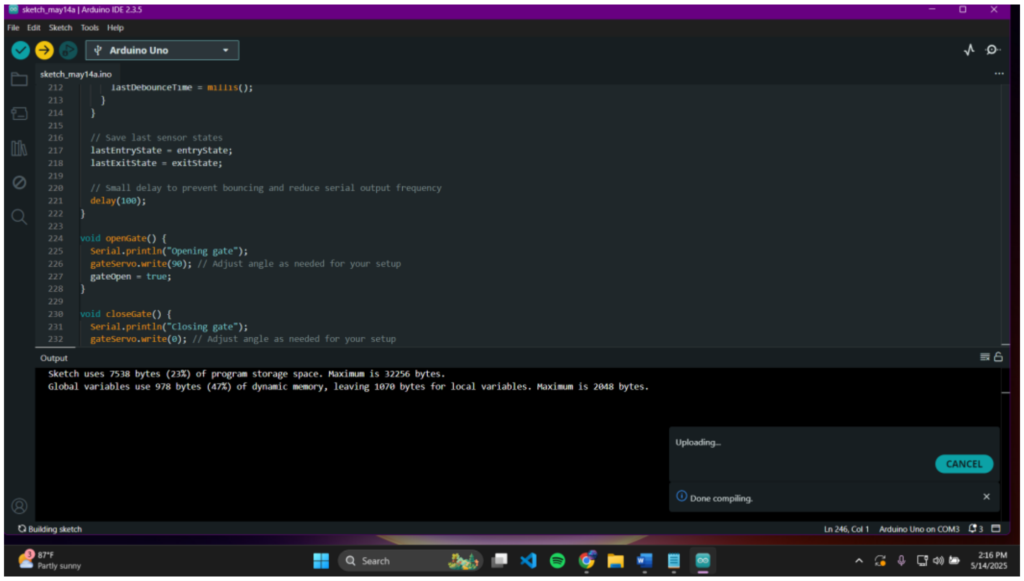


Figure 23 Servo motor wired: GND/VCC to arduino, signal to D9 for control

Phase 4: Executing the code

The final phase is the code compilation and execution. So, the source code was given in linkedin and we have done certain modifications in the code and then it was executed.


sketch_may14a | Arduino IDE 2.3.5
File Edit Sketch Tools Help
Arduino Uno
sketch_may14a.ino

```
        lastDebounceTime = millis();
      }
    }

    // Save last sensor states
    lastEntryState = entryState;
    lastExitState = exitState;

    // Small delay to prevent bouncing and reduce serial output frequency
    delay(100);
}

void openGate() {
  Serial.println("Opening gate");
  gateServo.write(90); // Adjust angle as needed for your setup
  gateOpen = true;
}

void closeGate() {
  Serial.println("Closing gate");
  gateServo.write(0); // Adjust angle as needed for your setup
```

```
Sketch uses 7538 bytes (23%) of program storage space. Maximum is 32256 bytes.
Global variables use 978 bytes (47%) of dynamic memory, leaving 1070 bytes for local variables. Maximum is 2048 bytes.
```

Figure 24 Executing the code

4. Result and Findings

The developed smart parking system fulfilled its main goal of creating a system for simple and efficient parking operations by reaching its implementation target successfully. During testing the system displayed reliable performance by operating with precision under different conditions.

The main components which make up the system include Infrared (IR) sensors together with Arduino UNO microcontroller and a servo motor and LCD display. The Arduino UNO receives a signal from the IR sensor which detects the arrival of a car in the parking zone. The Arduino UNO microcontroller reads the sensor input while verifying the status of open parking spaces. A servo motor receives instructions from the Arduino system when empty slots exist leading to immediate and controlled gate opening for vehicle access.

Through the LCD display drivers receive real-time messages like "Slots Available" "Parking Full" in order to make rapid choices without difficulty. Through its interactive system users achieve better experiences through shorter search times for parking spots as well as reduced traffic inside the area and decreased fuel usage.

System performance tests revealed outstanding results because the servo motor exhibited quick responses throughout its operation and kept synchronized with vehicle detection without delays. The reliable performance of the system design became possible through stable communication links between IR sensors and Arduino UNO in addition to servo motor and LCD.

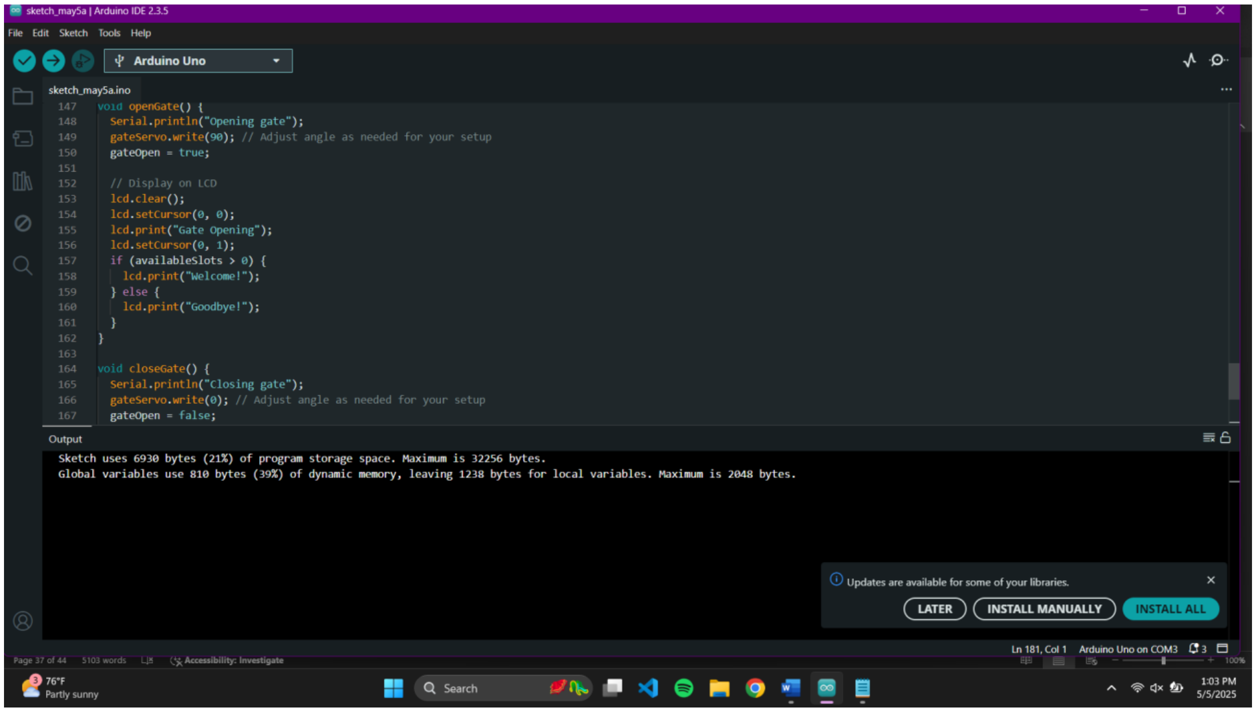
The system exhibits a significant quality by being expandable for additional uses. The open nature of the system architecture supports both the addition of extra sensors and wireless interface integration for distant monitoring functions. The project maintains its long-term viability because of its adaptable features

4.1 Testing

Test 1: To compile the code and execute it.

Test	1
Objective	To compile the code and execute it.
Action	• Firstly, connecting the Arduino with the laptop. • Then, compiling the code and executing it.
Expected Result	The code will be compiled and executed successfully.
Actual Result	The code is compiled and executed successfully.
Conclusion	The test was successful.

Table 1 Test 1: To compile the code and execute it


sketch_may5a | Arduino IDE 2.3.5
File Edit Sketch Tools Help
Arduino Uno
sketch_may5a.ino
Output
Updates are available for some of your libraries.
LATER
INSTALL MANUALLY
INSTALL ALL
Ln 181, Col 1 Arduino Uno on COM3

```
void openGate() {
  Serial.println("Opening gate");
  gateServo.write(90); // Adjust angle as needed for your setup
  gateOpen = true;

  // Display on LCD
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("Gate Opening");
  lcd.setCursor(0, 1);
  if (availableSlots > 0) {
    lcd.print("Welcome!");
  } else {
    lcd.print("Goodbye!");
  }
}

void closeGate() {
  Serial.println("Closing gate");
  gateServo.write(0); // Adjust angle as needed for your setup
  gateOpen = false;
```

```
Sketch uses 6930 bytes (21%) of program storage space. Maximum is 32256 bytes.
Global variables use 810 bytes (39%) of dynamic memory, leaving 1238 bytes for local variables. Maximum is 2048 bytes.
```

Figure 25 Test 1: To compile the code and execute it

Test 2: To check the rotation of the servo motor

Test	2
Objective	To check whether the servo motor rotates or not.
Action	• Firstly, connect the Arduino with the laptop. • Then verify and upload the code.
Expected Result	The servo motor will rotate vertically up to 90° if parking lot is empty otherwise it will not open.
Actual Result	The servo motor rotates vertically up to 90° if the parking lot is empty otherwise not.
Conclusion	The test was successful.

Table 2 Test 2: To check the rotation of the servo motor

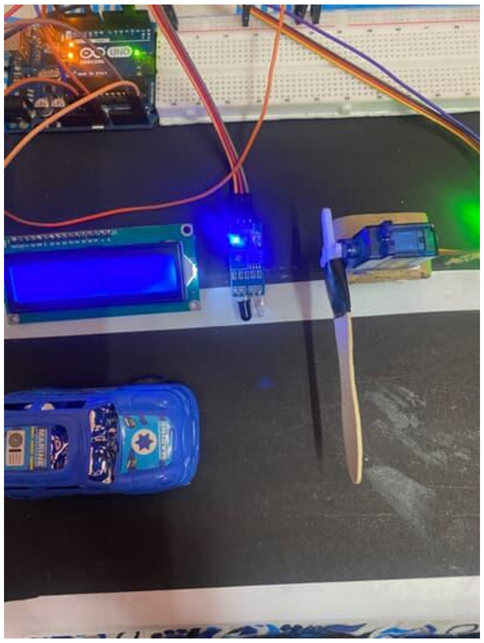


Figure 26 Test 2: Rotation of servo motor (1)

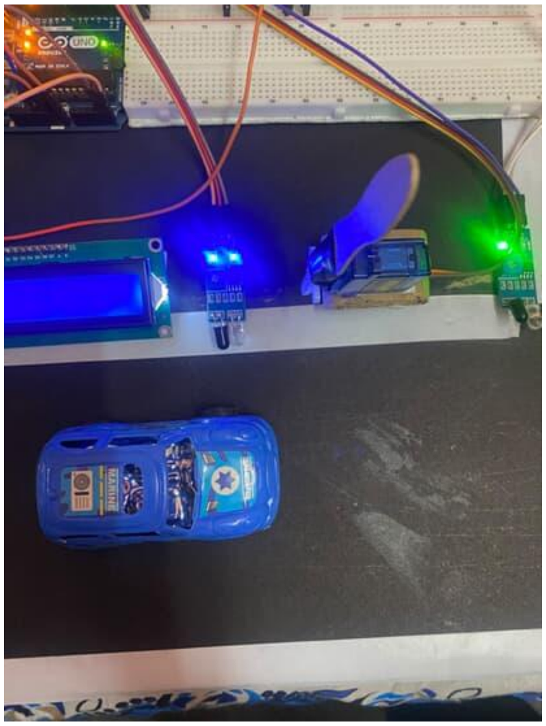


Figure 27 Test 2: Rotation of servo motor (2)

Test 3: Checking if the LCD display is working.

Test	3
Objective	To check if the LCD display is working.
Action	• Firstly, connect the Arduino with the laptop. • Execute any code and show that the LCD is working.
Expected Result	The LCD should display what is written in the code.
Actual Result	The LCD displays the thing written in the code.
Conclusion	The test was successful.

Table 3 Test 3: To check if the LCD display is working

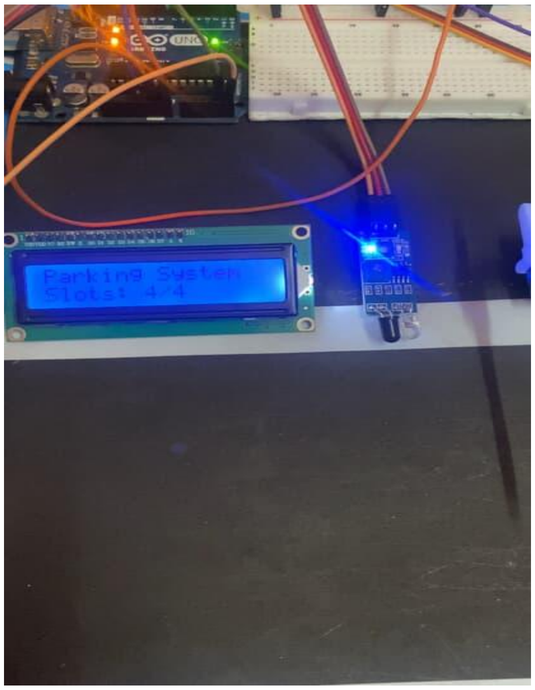


Figure 28 Test 3: Checking if the LCD display is working

Test 4: To check if the IR sensor detects the vehicles or not.

Test	4
Objective	To check if the IR sensor detects the vehicles or not.
Action	• Firstly, connect the Arduino with the laptop and run code with Arduino ide. • Place the object to see if the object is sensed.
Expected Result	The IR sensors should detect the presence of the vehicles.
Actual Result	The IR sensors will detect the presence of the vehicles.
Conclusion	The test was successful.

Table 4 Test 4: To check if the IR sensor detects the vehicles or not

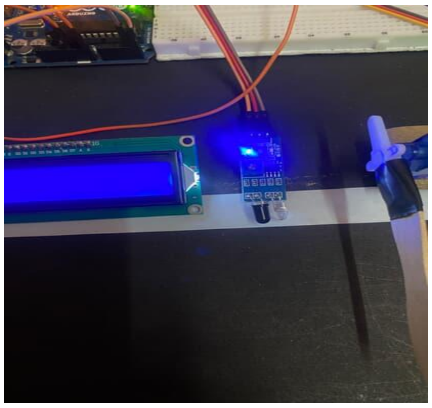


Figure 29 Test 4: IR sensor before it detects vehicle

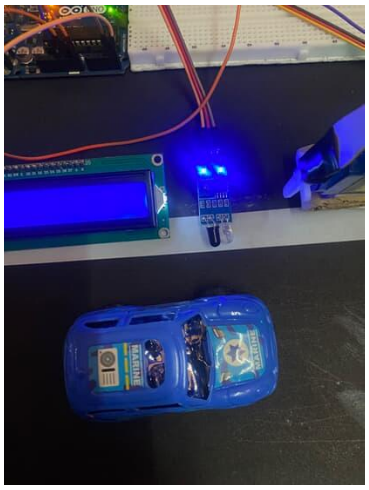


Figure 30 Test 4: IR sensor after it detects vehicle

Test 5: To check if the LCD display after code execution.

Test	5
Objective	To check if the LCD display after code execution.
Action	• Firstly, connect the Arduino with the laptop. • Execute the code and see if the LCD display updates or not.
Expected Result	The LCD display should update the status of availability of parking slots.
Actual Result	The LCD doesn’t display the status of availability of parking slots.
Conclusion	The test was unsuccessful

Table 5 Test 5: To check if the LCD display after code execution

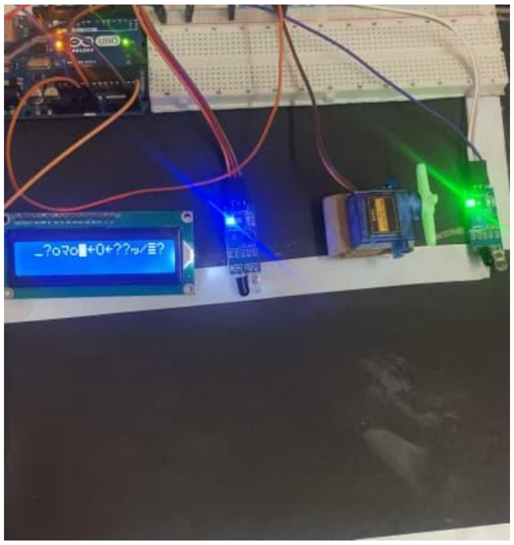


Figure 31 Test 5: To check if the LCD display after code execution

5. Future Works

The system operates as an integrated compact design serving as a basis for actual smart parking systems. blueprint for a real-life smart car parking system. This prototype has a lot of Future improvements in the project hold multiple opportunities because designers can enhance its features and make it more expansive. Additional development of this system at scale will enhance its features together with scalability and integration capabilities. The future development requires the creation of a mobile application that exhibits parking spot availability with reservation functionality. The system presents open parking slots for customer viewing while allowing reservations on specific locations.

The system displays spots for customers to allow better parking efficiency. For further energy efficiency and the system's energy needs can be sustained through solar panel power that would supply electricity to sensors and Arduino units as well as display screens.

LED screen. The user experience will benefit from visual feedback signals along with audio indicators. The system will provide audio directions about available parking spaces closest to customers and an automated billing system can complement these features. Guests should have automatic billing through the payment gateway system when their parking duration exceeds a specific threshold. An additional benefit of the future work advances the operational capabilities of smart parking. The smart parking system functions to address current issues including congestion as well as inefficiencies and unavailability of parking spots. The system suffers from parking inefficiency along with empty parking locations and other similar issues.

6. Conclusion

The system created during this project for the automated tollgate parking is an effective solution of the existing serious problem of poor management of parking in the urban centers like Kathmandu. Utilizing the underlying IoT technologies which are; Arduino Uno, IR sensors; an LCD display and a servo motor, the system automatically monitors the detection and assignment of parking spaces. This prototype sets the chances of receiving human intervention to zero by giving real-time information regarding availability of parking space and guiding access allowed to gates. While implementing and testing the system it had shown a stable performance. IR sensors did capture the presence of vehicle accurately, servo motor worked as stipulated and updates on LCD were clear as well, all translating into smooth and intelligent parking solution.

The system has modular and scalable architecture, which makes it not only viable as a prototype but also capable for making it deployable in real-world conditions. With a reduced manual operation, it provides better management of time, traffic, and users' satisfaction. The integration of hardware and software with success touches on the fact that microcontrollers and simple components can address actual urban-related issues efficiently.

Besides, the project would serve as a good start for the future improvements like the mobile app integration, solar powered functioning, automated billing machines and audio visual guidance, all of which might revolutionize the smart parking scene. This project is one of the examples when IoT proves how it could help to improve the everyday infrastructure and foster the smart cities and decrease congestion-related inefficiencies. It is a powerful example of the way through which technological innovations can be used to solve chronic problems in society through available and affordable engineering solutions.

7. References

Anon., science buddies. *science buddies.* [Online]
Available at: https://www.sciencebuddies.org/science-fair-projects/references/how-to-use-a-breadboard
[Accessed 7 04 2025].

Arduino, 2018. *arduino.* [Online]
Available at: https://www.arduino.cc/en/Guide/Introduction
[Accessed 7 04 2025].

computerhop, 2024. *computerhope.* [Online]
Available at: https://www.computerhope.com/jargon/d/drawio.htm
[Accessed 2 May 2025].

geeksforgeeks, 2025. *geeksforgeeks.* [Online]
Available at: https://www.geeksforgeeks.org/introduction-to-microsoft-word/
[Accessed 2 May 2025].

IGI GLOBAL, 2025. *IGI GLOBAL.* [Online]
Available at: https://www.igi-global.com/dictionary/examining-the-links-between-affect-toward-3d-printing-technology-and-interest-in-stem-careers-among-female-elementary-students/69479
[Accessed 2 May 2025].

indeed, 2025. *indeed.* [Online]
Available at: https://www.indeed.com/career-advice/career-development/what-is-block-diagram
[Accessed 2 May 2025].

Kang, B., 2023. *Medium.* [Online]
Available at: https://medium.com/design-bootcamp/system-design-and-system-architecture-e963d030bc7b
[Accessed 2 May 2025].

Kohli, K., 2023. *aakash.* [Online]
Available at: https://www.aakash.ac.in/blog/what-is-a-circuit-diagram-meaning-components-and-

importance/
[Accessed 2 May 2025].

Lim, A., 2024. *ThoughtCo..* [Online]
Available at: https://www.thoughtco.com/what-is-a-schematic-diagram-4584811
[Accessed 2 May 2025].

robu.in, 2020. *robu.in.* [Online]
Available at: https://robu.in/ir-sensor-working/
[Accessed 7 04 2025].

Yansar, K., 2025. *tech targets.* [Online]
Available at: https://www.techtarget.com/iotagenda/definition/Internet-of-Things-IoT
[Accessed 7 april 2025].

8.Appendix

8.1 Source Code

```
#include <Wire.h>              // I2C communication library

#include <LiquidCrystal_I2C.h> // I2C LCD library

#include <Servo.h>             // Servo library



// Pin definitions

const int entryIRPin = 2;      // IR sensor at entry

const int exitIRPin = 3;       // IR sensor at exit

const int servoPin = 9;        // Servo motor for gate



// Initialize objects

Servo gateServo;

// Set the LCD address to 0x27 for a 16 chars and 2 line display

// NOTE: Your I2C address might be different (common addresses are 0x27, 0x3F)

LiquidCrystal_I2C lcd(0x27, 16, 2);



// System variables

int totalSlots = 4;            // Total parking slots

int availableSlots = 4;        // Initially all slots are available

bool gateOpen = false;         // Gate status

unsigned long gateTimer = 0;   // Timer for auto-closing gate
```

```
const int gateOpenTime = 5000; // Time to keep gate open (5 seconds)


// Sensor state variables

int lastEntryState = HIGH;     // Assume sensors start HIGH (no detection)

int lastExitState = HIGH;

unsigned long lastDebounceTime = 0;

const int debounceDelay = 50;  // Debounce time in milliseconds


void setup() {

  // Initialize serial communication for debugging

  Serial.begin(9600);

  Serial.println("Initializing Parking System...");


  // Initialize IR sensors as inputs

  pinMode(entryIRPin, INPUT);

  pinMode(exitIRPin, INPUT);


  // Initialize servo

  gateServo.attach(servoPin);

  closeGate(); // Start with gate closed


  // Initialize LCD with I2C
  lcd.init();

  lcd.backlight();



  // Display welcome message

  lcd.clear();

  lcd.setCursor(0, 0);

  lcd.print("Parking System");

  lcd.setCursor(0, 1);

  lcd.print("Initializing...");

  delay(2000);



  updateLCD();



  Serial.println("Parking System Initialized");

  Serial.println("IR Sensor readings (LOW = detected, HIGH = not detected):");

}



void loop() {

  // Read current sensor states

  int entryState = digitalRead(entryIRPin);

  int exitState = digitalRead(exitIRPin);
  // Debug output - print sensor values to Serial Monitor
  Serial.print("Entry IR: ");
  Serial.print(entryState);
  Serial.print(" | Exit IR: ");
  Serial.print(exitState);
  Serial.print(" | Available Slots: ");
  Serial.print(availableSlots);
  Serial.print("/");
  Serial.println(totalSlots);

  // Check entry sensor (LOW when object detected for most IR sensors)
  // NOTE: You may need to change LOW to HIGH depending on your sensor behavior
  if (entryState == LOW && !gateOpen && entryState != lastEntryState) {
    // Debounce
    if ((millis() - lastDebounceTime) > debounceDelay) {
      Serial.println("Entry sensor triggered");

      if (availableSlots > 0) {
        // Open gate for entry
        openGate();
        // Decrement available slots
        availableSlots--;
        updateLCD();
        gateTimer = millis();
      } else {
        // No slots available
        lcd.clear();
        lcd.setCursor(0, 0);
        lcd.print("PARKING FULL");
        lcd.setCursor(0, 1);
        lcd.print("No Entry");
        delay(2000);
        updateLCD();
      }

      lastDebounceTime = millis();
    }
  }

  // Check exit sensor
  // NOTE: You may need to change LOW to HIGH depending on your sensor behavior
  if (exitState == LOW && !gateOpen && exitState != lastExitState) {
    // Debounce
    if ((millis() - lastDebounceTime) > debounceDelay) {
      Serial.println("Exit sensor triggered");

      if (availableSlots < totalSlots) {
        // Open gate for exit
        openGate();
        // Increment available slots
        availableSlots++;
        updateLCD();
        gateTimer = millis();
      }

      lastDebounceTime = millis();
    }
  }

  // Auto-close gate after timeout
  if (gateOpen && (millis() - gateTimer > gateOpenTime)) {
    closeGate();
  }

  // Save last sensor states
  lastEntryState = entryState;
  lastExitState = exitState;


  // Small delay to prevent bouncing and reduce serial output frequency
  delay(100);
}


void openGate() {
  Serial.println("Opening gate");
  gateServo.write(90); // Adjust angle as needed for your setup
  gateOpen = true;


  // Display on LCD
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("Gate Opening");
  lcd.setCursor(0, 1);
  if (availableSlots > 0) {
    lcd.print("Welcome!");
  } else {
    lcd.print("Goodbye!");
  }
}
```

```
void closeGate() {
  Serial.println("Closing gate");
  gateServo.write(0); // Adjust angle as needed for your setup
  gateOpen = false;
  updateLCD();
}

void updateLCD() {
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("Parking System");
  lcd.setCursor(0, 1);
  lcd.print("Slots: ");
  lcd.print(availableSlots);
  lcd.print("/");
  lcd.print(totalSlots);
}
```

8.2 List of Components Used

Arduino Uno

Breadboard

IR Proximity Sensors (x2)

Servo Motor

LCD Display with I2C Module

Jumper Wires

Usb cable and power supply

Click here for explainations

8.3 Individual contribution plan

Task are divided in 5 members of my group:

Student Name	Roles	Contribution
Bachan Timalsina	**Proposal:** Introduction, expected outcomes and current scenarios **System Development Report:** Procedure and working mechanism of the project. **Application Implementation:** Code to execute the connection between different components. **Presentation:** Working mechanism of the code.	20%

Osheen Shrestha	**Proposal:** Problem statement and Solutions of the proposal. **System Development Report:** Design and Diagram of the flowchart, circuit. **Application Implementation:** Setting up Arduino IDE and making connection between different components. **Presentation:** Expectation of the project.	20%
Rishav Kumar Thapa	**Proposal:** Abstract, Aims and Objective of the proposal **System Development Report:** Test cases. **Application implementation:** Reviewing the code and improving it. **Presentation:** Working mechanism of the project	20%
Sang Dorje Lama	**Proposal:** Conclusion and the acknowledgement of the proposal **System Development Report**: Prose of the IOT device, Hardware and Software evaluation **Application Implementation:** Setting up Arduino IDE and making connection between different components. **Presentation:** Purpose and objective of the project.	20%
Satvisha Panta	**Proposal:** Formatting of report and appendix **System Development Report**: Prose of the IOT device, Hardware and Software evaluation, analysis of component performance, compatibility and suitability.	20%

	Application Implementation: Setting up Arduino IDE, testing and troubleshooting hardware and making connection between different components. **Presentation:** Purpose, objective and explanation of project workflow.	