,

HALDIA INSTITUTE OF TECHNOLOGY

IoT Smart parking System Using RFID

Group – 11

Dipankar Saha (22/ECE/075)
Diptanshu Laha (22/ECE/076)
Aastha Choudhury (22/ECE/017)
Anindita Das (22/ECE/039)
Arighna Jha (22/ECE/053)
Ashish Kumar Rawani (22/ECE/056)
Ayushi Srivastava (22/ECE/065)
Barnik Pal (22/ECE/067)
Deb Agni Patra (22/ECE/072)
Debasmita Adak (22/ECE/073)

Acknowledgement

I would like to express my heartfelt gratitude to everyone who contributed to the successful completion of this project, "IoT Smart Parking Using RFID."

First and foremost, I extend my sincere thanks to my project guide, **Dr. Dibyendu Chowdhury**, **Dr. Avisankar Roy, prof. Surajit Mukharjee, prof. Tirthadip Sinha** , for their valuable insights, continuous support, and guidance throughout the development of this project. Their expert advice and encouragement have been instrumental in shaping the project and overcoming challenges.

I would also like to thank **Haldia Institute Of Technology**, particularly **the Department of Electronics and Communication Engineering**, for providing the necessary resources and a conducive environment for learning and development. The knowledge and skills I gained during my studies have been pivotal in bringing this project to life.

A special thanks to my peers and team members who collaborated with me during various phases of the project. Their teamwork and dedication helped ensure the timely and efficient execution of tasks.

Lastly, I would like to express my gratitude to my family and friends for their constant encouragement, patience, and support throughout this journey.

Thank you all for your invaluable contributions.

Contents

Aim

The aim of an IoT Smart Parking system using RFID is to automate and optimize the process of finding and managing parking spaces. This system uses RFID technology to identify vehicles and allocate parking spots, reducing the time spent searching for parking, improving space utilization, and enhancing the overall parking experience. It enables real-time monitoring of parking space availability and provides users with accurate parking information, which can be accessed remotely via an IoT platform. This approach also helps reduce traffic congestion and improves security by automating access control.

Bill of the components

Sl no.	Components	Quantity	Rate	Amount
1	ESP32 (Microcontroller)	1	450	450
2	RC522 (RFID Module)	1	130	130
3	16 x 2 LCD	1	130	130
4	I2C Module	1	55	55
5	MG90 (Servo Motor)	2	120	240
6	RFID card	4	30	120
7	Wire	3	10	30
8	Vero Board	1	40	40
9	Buzzer	1	15	15
10	Pinholes	6	10	60
11	330E (¼ w)	8	0.4	2
12	IR Proximity	4	30	120
13	Jumper wire	30	2	60
14	Sand board	5	60	300
15	Wood bits	8	4	32
16	LEDs	5	5	25
17	Tablet LEDS	15	3	45
18	Card board	1	60	60
19	Art paper	6	12	72
20	Glue gun	1	250	250
21	Soldering wire	1	120	120
22	Glue	1	60	60
23	Wax	1	20	20
24	Battery	2	25	50
			Total	**2486**

Introduction

In smart cities, the demand for innovative and efficient technologies is increasing, especially to address both surface-level and deeper issues, as well as to reduce urban congestion. One of the most frustrating challenges for drivers is finding a parking spot, particularly in public spaces such as shopping malls, luxury hotels, and movie theaters. Even within parking lots, drivers often waste of time and fuel for searching an open space, which not only leads to frustration but also contributes to environmental pollution.

This study proposes the development and design of a smart parking system aimed at effectively tackling these issues. Over recent years, numerous studies have sought to mitigate parking problems, making the process more convenient and user-friendly. A review of smart parking systems has been conducted, focusing on the practical application of such technologies, which rely on wireless sensor networks and real-time data from sensors. However, existing systems are often complex, lack robust access technology, and fail to provide guidance for locating parking spots.

In this project, an ESP 32 is used to develop a smart car parking system. The system employs infrared (IR) sensors installed in parking slots to detect vacant spaces and assist drivers, especially in unfamiliar locations. While the system lacks a payment feature and automated guidance for finding available spots, the primary goal is to simplify the parking process. This initiative helps drivers save time by offering accurate information on space availability. The setup includes IR sensors, servo motors, and an LCD display, all connected to an ESP 32 Uno microcontroller. The LCD indicates the number of available spaces, while the IR sensors track the entry and exit of vehicles, detecting whether a parking space is occupied or vacant.

Objective

The Smart Parking IoT System addresses a critical issue faced by drivers: the difficulty of quickly locating vacant parking spaces, especially in multilevel garages during peak times such as weekends or public events. Data shows that around 66% of customers spend over 10 minutes searching for a spot during these busy periods. This inefficiency, particularly in places like stadiums and shopping centers, leads to driver frustration and stifles activities due to inadequate parking space availability.

One solution is to employ a Smart Parking Assistance system. With rising fuel costs, drivers are more motivated than ever to conserve energy. However, long wait times at entrance gates during peak hours and the extended search for available spots contribute to unnecessary energy and time wastage.

The root issue often lies in poor planning by developers who fail to allocate sufficient parking spaces. Additionally, the lack of real-time, user-friendly information exacerbates the problem. During busy times, users are forced to circle parking areas multiple times, often with little to no guidance on space availability. Car park operators frequently neglect to update the status of the parking lot, leaving drivers unaware when an area is full.

To address these concerns, the Smart Parking Lot System should incorporate real-time monitoring and clear guidance, possibly through mobile apps or digital signage, to direct drivers efficiently. Furthermore, integrating reservation capabilities and automated notifications could significantly reduce wait times and improve overall customer satisfaction.

Brief Overview

Smart parking systems leverage IoT technology to enhance parking management, optimize space utilization, and improve the overall parking experience for both drivers and city residents.

Real-Time Data Availability:

- An IoT-based smart parking system provides real-time information about parking space availability.
- Drivers can access this data through mobile apps, websites, or digital displays, allowing them to make informed decisions before arriving at a parking spot.

Benefits of IoT Smart Parking:

Reduced Search Time: Drivers spend less time circling around looking for parking spots.

Efficient Space Utilization: Cities can optimize parking resources, reducing congestion and pollution.

Revenue Generation: Smart parking systems can introduce paid parking models, generating revenue for municipalities.

Improved Safety: Real-time data minimizes the risk of accidents caused by sudden stops while searching for parking.

Future Scopes and Trends

Autonomous Vehicles (AVs) Integration:

- As AVs become more prevalent, smart parking systems will need to accommodate their unique requirements.
- AVs may require charging stations, specialized parking zones, and adherence to specific parking rules.

Dynamic Pricing and Demand-Based Models:

Smart parking can evolve beyond fixed pricing models.

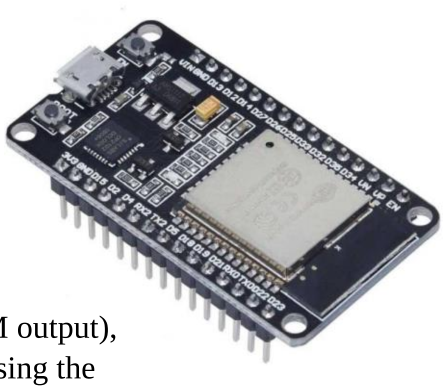
Dynamic pricing based on demand (peak hours, events, etc.) can optimize revenue and encourage efficient space utilization.

1. **Predictive Analytics**:
    - By analyzing historical data, smart parking systems can predict peak demand times, helping cities plan infrastructure accordingly.
2. **Integration with Navigation Apps**:
    - Imagine your navigation app guiding you not only to your destination but also to an available parking spot nearby.
3. **Eco-Friendly Solutions**:
    - Integrating EV (electric vehicle) charging stations into smart parking systems aligns with sustainability goals.
4. **Multi-Modal Transportation Hubs**:Smart parking can be part of larger transportation hubs that seamlessly connect buses, trains, and other modes of transit

Components of Smart parking

ESP32 (Microcontroller)-

The ESP32 is an open-source microcontroller board designed by ESP 32.cc and based on the Microchip ATmega328P microprocessor.



The board includes digital and analogue input/output (1/0) pins that can be used to connect to expansion boards (shields) and other circuits.

The board features 14 digital VO pins (six of which are capable of P WM output), 6 analogue 1/0 pins, and is programmable through a type B USB cable using the Arduino IDE (Integrated Development Environment).
It can be powered by a USB cable or an external 9-volt battery, with voltages ranging from 7 to 20 volts.

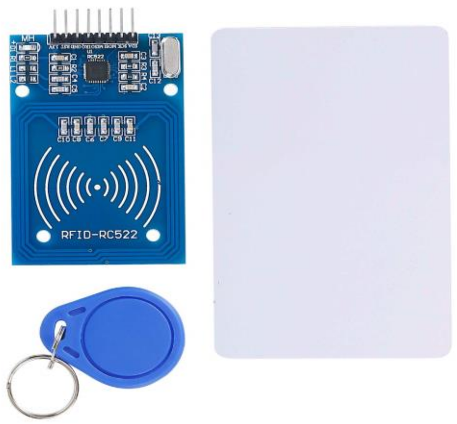
RC522 (RFID Module)-

The RC522 is a popular RFID (Radio Frequency Identification) module used to read and write RFID tags. It operates at 13.56 MHz and uses the SPI (Serial Peripheral Interface) protocol to communicate with microcontrollers like ESP 32 or Raspberry Pi.

Key features:

- **Low power consumption**: Ideal for embedded systems.
- **Easy to use**: With libraries available for microcontrollers, integrating RC522 into projects is simple.
- **Multiple card support**: Can read multiple RFID card types, including MIFARE cards.
- **Security features**: Supports authentication mechanisms to protect data on the tags.

Applications include access control systems, payment systems, and inventory tracking.



RFID Card –

An RFID card is a type of smart card embedded with a small radio frequency identification (RFID) chip and antenna. It is used for wireless data transmission and can be read by RFID readers without physical contact.

Key features:

- **Wireless communication**: Uses radio waves to communicate with RFID readers.
- **No need for power**: Most RFID cards are passive, meaning they don't need a battery.
- **Short-range**: Typically operates within a range of a few centimeters to a meter.
- **Security**: Can store data securely, often used for access control or payment systems.

RFID cards are widely used in applications like keyless entry, public transportation, and contactless payment systems.

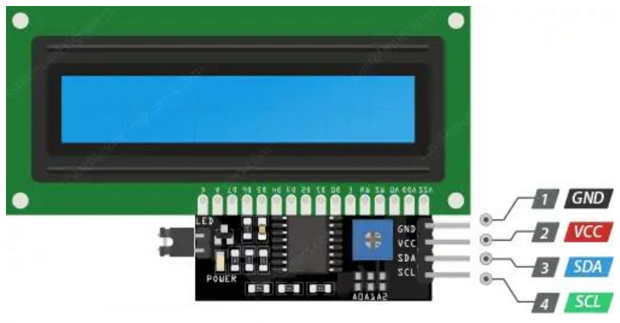
16*2 LCD Display -

An LCD (Liquid Crystal Display) is a flat-panel display technology commonly used in devices like smartphones, calculators, and TVs. It works by manipulating liquid crystals using electric currents to block or allow light to pass through, creating an image on the screen.

Key features:

- **Low power consumption**: Efficient for battery-powered devices.
- **Compact and lightweight**: Ideal for portable electronics.
- **High resolution**: Displays sharp and clear images.
- **Versatile**: Can show text, numbers, and graphics.

LCDs are widely used in various applications, from digital clocks to complex graphical interfaces.



<u>I2C Module</u> –

The I2C (Inter-Integrated Circuit) module is a communication protocol that allows multiple devices, like sensors and microcontrollers, to communicate with each other using just two wires: **SDA** (data) and **SCL** (clock). It is commonly used in embedded systems due to its simplicity and ability to connect multiple devices on the same bus.

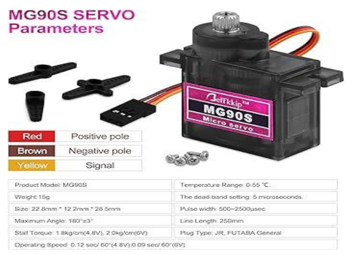
Key features:

- **Two-wire communication**: Reduces pin usage on microcontrollers.
- **Master-slave structure**: One device controls communication (master), while others respond (slaves).
- **Supports multiple devices**: Up to 128 devices can be connected to a single bus.
- **Synchronous**: Data transfer is synchronized with the clock signal.

I2C is widely used in applications like sensors, LCDs, and EEPROMs in projects that require communication between multiple components.

<u>MG90 (Servo Motors)</u> –

The MG90 is a metal gear micro servo motor commonly used in robotics and DIY electronics projects. It provides precise control of angular motion, making it ideal for small projects requiring rotational movement, such as robotic arms or remote-controlled vehicles.



Red	Positive pole
Brown	Negative pole
Yellow	Signal

Product Model: MG90S	Temperature Range: 0-55 ℃.
Weight: 15g	The dead-band setting: 5 microseconds.
Size: 22.8mm * 12.2mm * 28.5mm	Pulse width: 500~2500μsec
Maximum Angle: 180°±3°	Line Length: 250mm
Stall Torque: 1.8kg/cm(4.8V), 2.0kg/cm(6V)	Plug Type: JR, FUTABA General
Operating Speed: 0.12 sec/ 60°(4.8V) 0.09 sec/ 60°(6V)	

Key features:

- **Metal gears**: Durable and strong, capable of handling higher loads compared to plastic gear servos.
- **Compact size**: Small and lightweight, perfect for tight spaces.
- **180-degree rotation**: Can move from 0 to 180 degrees, providing accurate positioning.
- **Torque**: Offers good torque for its size, typically around 2 kg-cm.

The MG90 is widely used for tasks requiring precise movement, such as controlling levers, doors, or sensors in various applications.

<u>Wires</u> –

A wire is a conductor, typically made of metal like copper or aluminum, that is used to transmit electrical signals or power between components in a circuit.

Key features:

- **Electrical conductor**: Carries electric current with minimal resistance.
- **Various types**: Includes solid, stranded, and insulated wires, each suited for different applications.
- **Flexible or rigid**: Depending on the material and design, some wires are highly flexible, while others are rigid.
- **Insulation**: Often covered with plastic or rubber to prevent short circuits and protect users.

Wires are essential in electronics and electrical systems, connecting components in devices ranging from simple circuits to complex machinery.

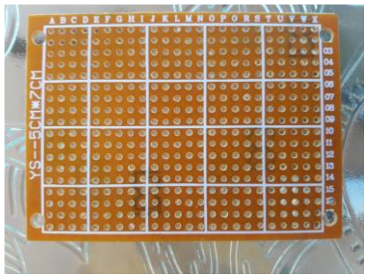
Vero Board –

A Vero board, also known as a stripboard, is a type of prototyping board used to build electronic circuits without soldering custom printed circuit boards (PCBs). It consists of a grid of holes with copper strips running along one side.

Key features:

- **Perforated board**: Has a grid of holes where components are inserted.
- **Copper strips**: Provide electrical connections, eliminating the need for extra wiring.
- **Flexible prototyping**: Ideal for creating and testing circuit designs before finalizing them.
- **No pre-made connections**: Users manually cut or modify strips to shape the circuit.

Vero boards are widely used by hobbyists and engineers for small to medium-sized projects, making prototyping faster and easier.



Buzzer –

A buzzer is a small audio device that produces sound, typically used to provide alerts, alarms, or feedback in electronic circuits.

Key features:

- **Simple sound output**: Generates a buzzing or beeping noise.
- **Types**: Can be active (built-in sound generation) or passive (requires an external signal).
- **Low power consumption**: Efficient, making it ideal for battery-powered devices.
- **Compact**: Small in size, suitable for embedding in a variety of electronics.

Buzzers are commonly found in alarms, timers, and devices like microwaves and computers for indicating notifications or errors.

Pin holes –

Pin holes are small, pre-drilled openings on circuit boards, like breadboards or Vero boards, used to insert the leads of electronic components for prototyping or building circuits.

Key features:

- **Component insertion**: Designed for easily placing components like resistors, capacitors, or ICs.
- **Standard spacing**: Typically follow a 0.1-inch grid pattern to accommodate standard-sized components.
- **Electrical connection**: Pin holes on some boards (e.g., breadboards or Vero boards) are connected by conductive traces or strips.
- **Reusable**: Often used for temporary or semi-permanent setups in prototyping.

Pin holes help in building and testing circuits efficiently, making them a fundamental part of electronics development.

330E(1/4 W) –

A 330E (1/4W) is a resistor with a resistance value of **330 ohms** (denoted by "330E") and a power rating of **1/4 watt**. Resistors like these are essential components in electronic circuits, used to limit current, divide voltages, or protect other components.

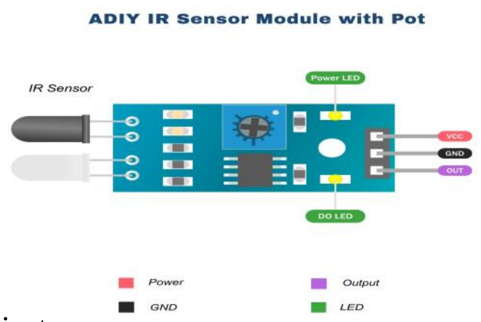
Key features:

- **Resistance value**: 330 ohms, controlling current flow based on Ohm's law.
- **Power rating**: 1/4 watt, meaning it can safely dissipate up to 0.25 watts of power.
- **Standard size**: Small and suitable for most low-power applications.
- **Color code**: Typically marked with colored bands for easy identification.

This type of resistor is commonly used in basic electronic circuits for current limiting or protection.

IR Proximity –

An IR (infrared) proximity sensor is a device that uses infrared light to detect the presence or distance of objects nearby. It emits an infrared beam and measures the reflection or interruption of this beam to determine how close an object is.



Key features:

- **Infrared light**: Emits and detects infrared light to sense objects.
- **Non-contact**: Detects objects without physical contact, making it useful for various applications.
- **Adjustable range**: Can be set to detect objects at different distances.
- **Common applications**: Used in automation, robotics, and obstacle detection.

IR proximity sensors are widely used in applications like touchless switches, automated systems, and obstacle avoidance in robots.

Jumper Wire –

Jumper wires are flexible wires used to connect different parts of a circuit or electronic components on a breadboard or other prototyping setup.

Key features:

- **Flexible and insulated**: Easy to handle and connect without soldering.
- **Pre-stripped ends**: Often come with exposed metal tips that fit into breadboard holes or connectors.
- **Variety of lengths**: Available in various lengths to suit different circuit designs.
- **Color-coded**: Usually in different colors to help identify connections.

Jumper wires are essential for building and modifying circuits during prototyping and testing phases.

Sand Board –

A sand board, also known as a sandpaper board or sanding board, is a tool used for smoothing or shaping surfaces. It consists of a flat board covered with sandpaper or abrasive material.

Key features:

- **Surface smoothing**: Helps in sanding wood, metal, or other materials to create a smooth finish.
- **Variety of grits**: Comes in different abrasive grits for various levels of roughness.

- **Flat surface**: Provides a consistent surface for even sanding.

  Sand boards are commonly used in woodworking, metalworking, and crafting to prepare surfaces for finishing or painting.

Wood bits -Wood bits are specialized drill bits designed for drilling into wood. They come in various shapes and sizes to make clean and precise holes.

Key features:

- **Sharp cutting edges**: Designed to cut through wood efficiently.
- **Various types**: Includes spade bits, auger bits, and Forstner bits, each suited for different drilling tasks.
- **Smooth holes**: Helps in creating clean and accurate holes without splintering the wood.

Wood bits are commonly used in woodworking and carpentry for tasks like creating holes for screws, dowels, or fittings.

Pin configuration

7a. MFRC522(RFID Module) Pin to ESP32 Pin:

SDA → GPIO 5 (Can be any GPIO, but it needs to match in the code)
SCK → GPIO 18 (Can be any GPIO, but it needs to match in the code)
MOSI → GPIO 23 (Can be any GPIO, but it needs to match in the code)
MISO → GPIO 19 (Can be any GPIO, but it needs to match in the code)
IRQ → Not connected
RST → GPIO 22 (Can be any GPIO, but it needs to match in the code)
VCC → 3.3V (Make sure not to use 5V, as the ESP32 operates at 3.3V)
GND → GND

7b. 16*2 LCD Display Pin to ESP32 Pin:

SDA - GPIO 21	I2C Data Line
SCL - GPIO 22	I2C Clock Line
VCC - 3.3V	Power Supply
GND - GND	Ground

7c. MG90 (Servo Motors) pin to ESP32 Pin:

Servo Entry – D26	(This pin is use to send signals to entry gate's servo motor)
Servo Exit – D27	(This pin is use to send signals to exit gate's servo motor)
VCC - 3.3V	Power Supply
GND - GND	Ground

7d. IR Sensor Pin to ESP32 :

1st IR output pin to- D32

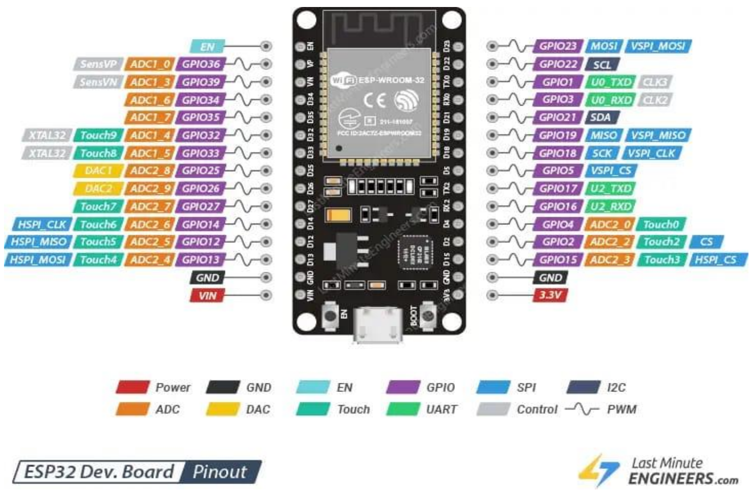
2nd IR output pin to -D33

3rd IR output pin to – D34

4th IR output pin to -D35

VCC pin to – 3.3v power supply

GND - GND



Programing and logic

```
#define BLYNK_TEMPLATE_ID "TMPL3TzWGS14D"
#define BLYNK_TEMPLATE_NAME "Smart Car Parking"
#define BLYNK_AUTH_TOKEN "oRAPLHAAG6waObAjfMp01MpXE2XhZKEn"

#include <Wire.h>
#include <LiquidCrystal_I2C.h>
#include <ESP32Servo.h>
#include <WiFi.h>
#include <BlynkSimpleEsp32.h>
#include <MFRC522.h>  // RFID Library
#include <HTTPClient.h>
// Pin Definitions
#define SS_PIN 5      // RFID SS pin for RFID
#define RST_PIN 22    // RFID RST pin
#define SERVO_ENTRY_PIN 26  // Servo motor pin for Entry gate
#define SERVO_EXIT_PIN 27   // Servo motor pin for Exit gate
#define IR_SENSOR_1_PIN 32  // IR Sensor for parking slot 1
#define IR_SENSOR_2_PIN 33  // IR Sensor for parking slot 2
#define IR_SENSOR_3_PIN 34  // IR Sensor for parking slot 3
#define IR_SENSOR_4_PIN 35  // IR Sensor for parking slot 4
#define BUZZER_PIN 14  // Buzzer pin
#define LED_ENTRY_PIN 16  // Entry gate LED pin
#define LED_EXIT_PIN 17   // Exit gate LED pin
// Servo and LCD setup
Servo entryServo;
Servo exitServo;
LiquidCrystal_I2C lcd(0x27, 16, 2);  // Set LCD I2C address (0x27)
// RFID Reader
MFRC522 rfid(SS_PIN, RST_PIN);  // Create MFRC522 instance


// WiFi credentials
char ssid[] = "dipankar";
char pass[] = "dipankar1";
```

```
// Blynk auth token
char auth[] = "oRAPLHAAG6waObAjfMp01MpXE2XhZKEn";
// Variables
String car[4] = {"", "", "", ""};  // Simulating four parking spots
String record = "";
String authorizedUIDs[] = {"41C37627", "13A65AA7", "83233025", "51E86109", "83C82925"};  // Example
RFID UIDs for authorized cars
BlynkTimer timer;
// Function to manually clear a specific line on the 16x2 LCD
void clearLCDLine(int line) {
  lcd.setCursor(0, line);   // Move the cursor to the beginning of the line
  lcd.print("                ");  // Print 16 spaces to clear the line (for a 16x2 display)
}
void setup() {
  Serial.begin(115200);
  // Initialize WiFi and Blynk
  Blynk.begin(auth, ssid, pass);
  Serial.println("Connecting to WiFi...");
  // Wait for Blynk connection
  while (Blynk.connect() == false) {
    delay(100);
  }
  Serial.println("Connected to WiFi and Blynk");
  // Initialize LCD
  lcd.init();
  lcd.backlight();
  lcd.setCursor(0, 0);
  lcd.print("Smart Parking");
  lcd.setCursor(0, 1);
  lcd.print("System");
  Serial.println("LCD Initialized");
  // Show initialization message for 2 seconds
  delay(2000);  // Wait for 2 seconds to let the message appear
  // Immediately proceed to the parking status display (no need to clear)
  // Initialize Servos
  entryServo.attach(SERVO_ENTRY_PIN);
  exitServo.attach(SERVO_EXIT_PIN);
  // Initialize RFID
  SPI.begin();
  rfid.PCD_Init();
  Serial.println("RFID Initialized");
  // Set the initial servo position (closed)
  entryServo.write(0);
  exitServo.write(0);
  // Initialize IR Sensors
  pinMode(IR_SENSOR_1_PIN, INPUT);
  pinMode(IR_SENSOR_2_PIN, INPUT);
  pinMode(IR_SENSOR_3_PIN, INPUT);
  pinMode(IR_SENSOR_4_PIN, INPUT);
   // Initialize Buzzer and LEDs
  pinMode(BUZZER_PIN, OUTPUT);
  pinMode(LED_ENTRY_PIN, OUTPUT);
  pinMode(LED_EXIT_PIN, OUTPUT);
  // Set up the Blynk timer to update parking status every second
  timer.setInterval(1000L, updateParkingStatus);  // Check parking status every second
}
void loop() {
  Blynk.run();  // Handle Blynk connection
  timer.run();  // Handle Blynk Timer to update the parking status regularly
  handleRFID(); // Handle RFID for entry/exit
}
// Function to handle RFID car entry/exit logic
void handleRFID() {
  if (rfid.PICC_IsNewCardPresent() && rfid.PICC_ReadCardSerial()) {
    String uid = getRFIDUid();
    if (isCarParked(uid)) {  // Check if the car is already parked (exit scenario)
      clearLCDLine(0);
      lcd.setCursor(0, 0);
      lcd.print("Car Exiting");
      openExitGate();
      delay(1000);
      carExit(uid);  // Car exiting
    } else if (isAuthorized(uid)) {  // If the car is authorized, check for space and allow entry
      if (isParkingAvailable()) {   // Check if there's an available spot
        clearLCDLine(0);
        lcd.setCursor(0, 0);
        lcd.print("Access Granted");
        openEntryGate();
        delay(1000);
        carEntry(uid);  // Car entering
      } else {
        // Parking is full, do not open entry gate
        clearLCDLine(0);
        lcd.setCursor(0, 0);
        lcd.print("Parking Full");
        Serial.println("Parking Full - No space available");
        // Entry gate should NOT open when parking is full
        // Buzzer response for parking full
        digitalWrite(BUZZER_PIN, HIGH);  // Turn on the buzzer
        delay(3000);  // Buzz for 1.5 seconds
        digitalWrite(BUZZER_PIN, LOW);  // Turn off the buzzer
      }
    } else {  // Unauthorized car
      clearLCDLine(0);
      lcd.setCursor(0, 0);
      lcd.print("Access Denied");
      delay(2000);
    }
    rfid.PICC_HaltA();  // Halt the RFID card
  }
}
// Function to get the RFID UID
String getRFIDUid() {
  String uid = "";
  for (byte i = 0; i < rfid.uid.size; i++) {
    uid.concat(String(rfid.uid.uidByte[i] < 0x10 ? "0" : ""));
    uid.concat(String(rfid.uid.uidByte[i], HEX));
  }
  uid.toUpperCase();  // Ensure the UID is uppercase
  Serial.println("UID: " + uid);
  return uid;
}
// Function to check if the UID is authorized
bool isAuthorized(String uid) {
  for (int i = 0; i < sizeof(authorizedUIDs) / sizeof(authorizedUIDs[0]); i++) {
    if (uid == authorizedUIDs[i]) {
      return true;  // UID is authorized
    }
  }
  return false;  // UID is not authorized
}
// Function to check if a car is already parked
bool isCarParked(String uid) {
  for (int i = 0; i < 4; i++) {  // Check all four parking spots
    if (car[i] == uid) {
      return true;  // Car with this UID is already parked
    }
  }
  return false;
}
// Function to check if parking is available (if any parking spot is empty)
bool isParkingAvailable() {
  for (int i = 0;  i < 4; i++) {
    if (car[i] == "") {
      return true;  // There is an empty parking spot
    }
  }
  return false;  // No empty spots, parking is full
}
// Function for entering a car (after authentication)
```

```
void carEntry(String carX) {
  for (int i = 0; i < 4; i++) {  // Limiting to 4 parking slots
    if (car[i] == "") {
      car[i] = carX;
      record += "\n" + carX + " Entered at " + String(millis() / 1000) + " seconds";
      Blynk.virtualWrite(V3, record);  // Displaying car entry record on Blynk
    // Buzzer and LED response for entry
      digitalWrite(BUZZER_PIN, HIGH);  // Turn on the buzzer
      digitalWrite(LED_ENTRY_PIN, HIGH);  // Turn on entry LED
      delay(1000);  // Buzz for 1 second
      digitalWrite(BUZZER_PIN, LOW);  // Turn off the buzzer
      digitalWrite(LED_ENTRY_PIN, LOW);  // Turn off entry LED

      // Send entry data to Google Sheets
      sendToGoogleSheets(carX, "Entered");
      closeEntryGate();  // Close the gate after entry
      return;
    }
  }
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("Parking Full");
  Serial.println("Parking Full");
}
// Function for exiting a car
void carExit(String carX) {
  for (int i = 0; i < 4; i++) {  // Limiting to 4 parking slots
    if (car[i] == carX) {
      car[i] = "";
      record += "\n" + carX + " Exited at " + String(millis() / 1000) + " seconds";
      Blynk.virtualWrite(V3, record);  // Displaying car exit record on Blynk
      // Buzzer and LED response for exit
      digitalWrite(BUZZER_PIN, HIGH);  // Turn on the buzzer
      digitalWrite(LED_EXIT_PIN, HIGH);  // Turn on exit LED
      delay(1000);  // Buzz for 1 second
      digitalWrite(BUZZER_PIN, LOW);  // Turn off the buzzer
      digitalWrite(LED_EXIT_PIN, LOW);  // Turn off exit LED

       // Send exit data to Google Sheets
      sendToGoogleSheets(carX, "Exited");
      closeExitGate();  // Close the exit gate after the car leaves
      return;
    }
  }
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("Car not found");
  Serial.println("Car not found.");
}
// Functions to handle servos for entry/exit gates
void openEntryGate() {
  entryServo.write(90);  // Open the entry gate
  delay(3000);  // Hold open for 3 seconds
}
void closeEntryGate() {
  entryServo.write(0);  // Close the entry gate
}
void openExitGate() {
  exitServo.write(90);  // Open the exit gate
  delay(3000);  // Hold open for 3 seconds
}
void closeExitGate() {
  exitServo.write(0);  // Close the exit gate
}
// Function to update parking status based on IR sensor readings
void updateParkingStatus() {
  int sensor1Status = digitalRead(IR_SENSOR_1_PIN);  // Read IR sensor 1
  int sensor2Status = digitalRead(IR_SENSOR_2_PIN);  // Read IR sensor 2
  int sensor3Status = digitalRead(IR_SENSOR_3_PIN);  // Read IR sensor 3
  int sensor4Status = digitalRead(IR_SENSOR_4_PIN);  // Read IR sensor 4
  // Display sensor statuses on Blynk
  Blynk.virtualWrite(V5, sensor1Status == LOW ? "Occupied" : "Empty");
  Blynk.virtualWrite(V6, sensor2Status == LOW ? "Occupied" : "Empty");
  Blynk.virtualWrite(V7, sensor3Status == LOW ? "Occupied" : "Empty");
  Blynk.virtualWrite(V8, sensor4Status == LOW ? "Occupied" : "Empty");
  // Send the number of full slots to the gauge
  int fullSlots = (sensor1Status == LOW) + (sensor2Status == LOW) + (sensor3Status == LOW) +
(sensor4Status == LOW);
  Blynk.virtualWrite(V10, fullSlots);
  // Display on LCD in a compact form (2 slots per line)
  lcd.setCursor(0, 0);
  lcd.print(sensor1Status == LOW ? "S1:F " : "S1:E ");  // Slot 1 status
  lcd.print(sensor2Status == LOW ? "S2:F" : "S2:E");    // Slot 2 status
  lcd.setCursor(0, 1);
  lcd.print(sensor3Status == LOW ? "S3:F " : "S3:E ");  // Slot 3 status
  lcd.print(sensor4Status == LOW ? "S4:F" : "S4:E");    // Slot 4 status
}
// Google sheet response
void sendToGoogleSheets(String uid, String action) {
  if (WiFi.status() == WL_CONNECTED) {  // Check Wi-Fi connection status
    HTTPClient http;

http.begin("https://script.google.com/macros/s/AKfycbzeQxAIJp06mjvC5ylqd1PCMewlkURaSWxagZoGx
W-i-VnaGRZtfWLAhZL9oTs8FKR7Lw/exec");  // Your Google Apps Script URL
    http.addHeader("Content-Type", "application/json");


    // Create the JSON payload
    String payload = "{\"uid\":\"" + uid + "\", \"action\":\"" + action + "\", \"timestamp\":\"" + String(millis() /
1000) + "\"}";
    // Send HTTP POST request
    int httpResponseCode = http.POST(payload);
    // Check the response code
    if (httpResponseCode > 0) {
      String response = http.getString();  // Get the response
      Serial.println(httpResponseCode);
      Serial.println(response);
    } else {
      Serial.println("Error in sending POST request");
      Serial.println(httpResponseCode);
    }
    http.end();  // End the connection
  } else {
    Serial.println("WiFi Disconnected");}}
```

Gantt chart: -

Task /Days	09 sept 2024	10 sept 2024	11 sept 2024	12 sept 2024	13 sept 2024	14 sept 2024	15 sept 2024	16sept 2024	17 sept 2024	18 sept 2024	19 sept 2024
Planning	🟦	🟦									
Implementation		🟦	🟦	🟦	🟦	🟦	🟦				
Designing							🟦	🟦	🟦		
Finishing Up									🟦	🟦	🟦

Challenges and difficulties

1. **Integration of IoT Components:** One of the primary challenges was integrating different IoT components, such as RFID sensors, microcontrollers, and communication modules, to ensure seamless data exchange. Ensuring that the hardware and software components worked together harmoniously required significant troubleshooting and testing.
2. **RFID Signal Interference:** During the implementation phase, managing RFID signal interference was a challenge. Ensuring reliable and accurate detection of vehicles, especially in environments with multiple RFID readers and tags, required fine-tuning of signal sensitivity and calibration.
3. **Real-Time Data Processing:** Handling and processing real-time data from RFID sensors presented another challenge. Developing a system that could quickly and efficiently process and relay parking availability in real time demanded optimizing both the hardware setup and the software algorithms.
4. **Network Connectivity Issues:** Ensuring consistent and reliable internet connectivity for IoT devices was crucial for real-time monitoring. However, dealing with network latency and connectivity issues at various stages sometimes led to delays in data transmission and system response.
5. **Power Management:** Managing the power consumption of the IoT components was another technical challenge. The project required energy-efficient hardware to ensure prolonged operation of devices, particularly in areas where constant power supply might not be guaranteed.
6. **Security Concerns:** Since RFID technology involves the transmission of data wirelessly, ensuring the security of vehicle and parking data was a concern. Addressing potential vulnerabilities and securing the communication between RFID devices and the central server required additional encryption and authentication mechanisms.
7. **Hardware Availability and Configuration:** Procuring the right components and configuring the hardware accurately posed logistical challenges. Some devices required additional research and time to be fully integrated into the system.

Conclusion

The "IoT Smart Parking Using RFID" project successfully demonstrates the potential of integrating IoT technologies with RFID systems to create a more efficient and automated parking management solution. By leveraging real-time data collection and processing, the system enables users to monitor parking availability, ensuring a smoother parking experience while reducing time and traffic congestion.

Through this project, the implementation of RFID for vehicle identification and the IoT infrastructure for communication proved to be effective in solving common parking-related challenges. Despite encountering issues with hardware integration, network reliability, and signal interference, these challenges were overcome through careful calibration, optimization, and troubleshooting.

This project not only highlights the importance of IoT in modern smart city initiatives but also provides a scalable solution that can be expanded further to include additional features such as mobile app integration, payment systems, and enhanced security protocols.

In conclusion, the "IoT Smart Parking Using RFID" system offers a practical and innovative solution to modern parking problems and stands as a testament to the growing influence of IoT and automation in daily life.