Project Title
Smart Parking System with Real-Time Monitoring and Control

Introduction
With the growing number of vehicles, managing parking spaces efficiently has become a significant challenge. The proposed Smart Parking System leverages modern IoT technology to provide a user-friendly and efficient parking management solution. This system utilizes an ESP32 microcontroller, IR sensors, an LCD display, and the Blynk app to monitor and control parking operations in real time.

Objective
The main objective of this project is to design and implement a smart parking system that:

- Automatically detects vehicle entry and exit.
- Monitors the availability of parking slots in real time.
- Displays slot status on an LCD screen and updates the Blynk app.
- Provides a user-friendly interface for both administrators and users.

System Overview
The Smart Parking System consists of the following key components:

1. **ESP32 Microcontroller:**
    - Serves as the central control unit for the system.
    - Handles sensor data and communicates with the Blynk app.
2. **IR Sensors:**
    - Two IR sensors placed at the entrance and exit to detect vehicle movement.

3. **LCD Display:**
    - Provides real-time information about available slots at the parking lot.
4. **Blynk App:**
    - Displays real-time slot availability on a smartphone.
    - Notifies users of parking slot availability.

Features:

1. **Real-Time Monitoring:**
    - Automatic detection of vehicle entry and exit.
    - Updates the number of available slots on the LCD and the Blynk app.
2. **User-Friendly Interface:**
    - Easy-to-use mobile app with live updates.
    - Clear and concise slot availability displayed on the LCD.
3. **Efficient Parking Management:**
    - Tracks vehicle movement and prevents over-parking.
    - Provides accurate, real-time information to users and administrators.

Components Required:

1. ESP32 microcontroller.
2. IR sensors (x2) for entrance and exit detection.
3. 16x2 LCD display.
4. Jumper wires, and a breadboard.
5. Power supply.
6. Smartphone with the Blynk app.
7. Internet connection for IoT functionality.

Implementation Plan:

- **System Design:**
  - Design the circuit and layout for the parking system.
  - Develop the flowchart and algorithm for the system.
- **Hardware Setup:**
  - Connect the ESP32 microcontroller with sensors and the LCD.
  - Assemble the components on a breadboard or PCB.
- **Software Development:**
  - Write the code for the ESP32 to process sensor data and control the LCD and Blynk app.
  - Develop the Blynk app interface to display slot availability.
- **Testing:**
  - Test individual components for accuracy and reliability.
  - Perform system integration testing to ensure seamless operation.
- **Deployment:**
  - Install the system in a prototype parking lot setup.
  - Demonstrate the system's functionality.

Expected Outcomes:

1. A fully functional smart parking system prototype.
2. Real-time monitoring and control of parking slots via an LCD screen and the Blynk app.

Conclusion

The proposed Smart Parking System aims to address parking challenges by providing an efficient, automated solution. This project will not only enhance the parking experience for users but also serve as a stepping stone for implementing larger-scale IoT-based systems.

ESP32 SOURCE CODE

```
#include <LiquidCrystal_I2C.h>
#include <ESP32Servo.h>
#include <WiFi.h>
#include <BlynkSimpleEsp32.h>
#include <WiFiClient.h>

// Wi-Fi credentials
char ssid[] = "Sufyan@Cyber0333-3370979";  // Replace with your Wi-Fi network name
char pass[] = "abcd1234";                  // Replace with your Wi-Fi password

// Blynk Auth Token and Template
#define BLYNK_TEMPLATE_ID "TMPL64S2CCxZT"
#define BLYNK_TEMPLATE_NAME "Park Arena"
#define BLYNK_AUTH_TOKEN "eiLM6MCcM7dzk5uNo_tqUKqMaIK0CIaJ"

// Pin definitions
const int IR1 = 2;    // Entry IR sensor
const int IR2 = 4;    // Exit IR sensor
const int SERVO_PIN = 5; // Servo motor pin

// Parking slot configurations
const int MAX_SLOTS = 5;
int Slot = MAX_SLOTS;  // Available slots

// Servo motor positions
const int GATE_OPEN_ANGLE = 0;
const int GATE_CLOSED_ANGLE = 100;

// Debounce variables
unsigned long lastActionTime = 0;
const unsigned long debounceDelay = 1000;  // 1 second debounce for IR sensors

// LCD and Servo objects
LiquidCrystal_I2C lcd(0x27, 16, 2);
```

```
Servo myservo;
BlynkTimer timer;
// Function Declarations
void updateLCD();
void openGate();
void closeGate();
void handleEntry();
void handleExit();
void myTimerEvent();

void setup()
{
   Serial.begin(115200);
   Serial.println("ESP32 is starting...");

   // Connect to Wi-Fi and Blynk
   Blynk.begin(BLYNK_AUTH_TOKEN, ssid, pass);

   // Initialize LCD
   lcd.init();
   lcd.backlight();
   lcd.setCursor(0, 0);
   lcd.print("      SMART      ");
   lcd.setCursor(0, 1);
   lcd.print(" PARKING SYSTEM ");
   delay(5000);  // Display for 5 seconds

   lcd.clear();
   updateLCD();  // Show "WELCOME!" and available slots after
initialization

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

   // Attach servo motor to the designated pin
   myservo.attach(SERVO_PIN);
   myservo.write(GATE_CLOSED_ANGLE); // Ensure gate starts as
closed
```

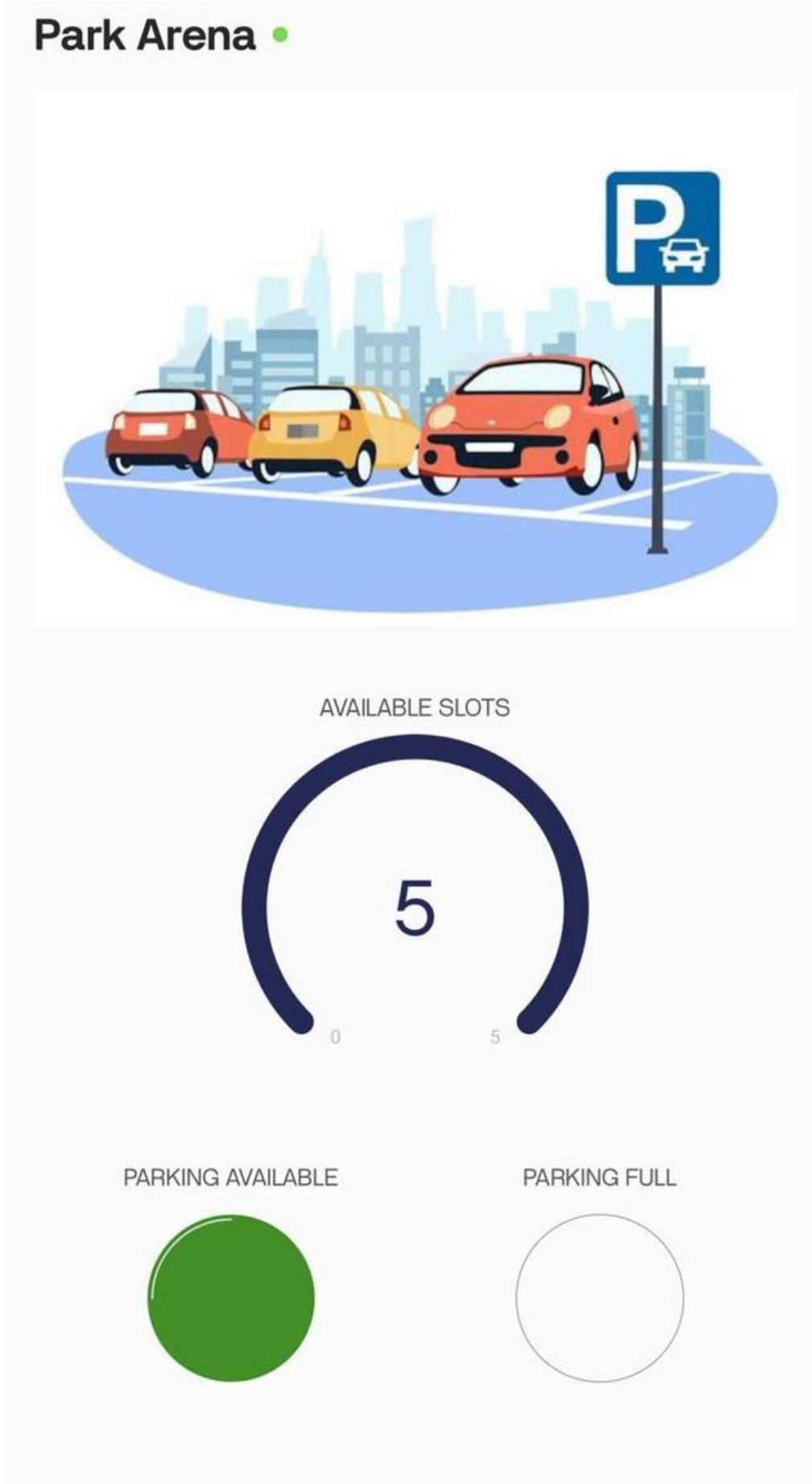
```
    // Initialize Blynk Gauge correctly
    Blynk.virtualWrite(V1, Slot);  // Start with all slots available

    // Set the timer to send data to Blynk every second
    timer.setInterval(1000L, myTimerEvent);
}

void loop()
{
    unsigned long currentMillis = millis();

    // Handle car entry (only if IR1 goes from HIGH to LOW)
    if (digitalRead(IR1) == LOW && (currentMillis - lastActionTime) >
debounceDelay)
    {
        lastActionTime = currentMillis;
        handleEntry();
    }

    // Handle car exit (only if IR2 goes from HIGH to LOW)
    if (digitalRead(IR2) == LOW && (currentMillis - lastActionTime) >
debounceDelay)
    {
        lastActionTime = currentMillis;
        handleExit();
    }

    Blynk.run();
    timer.run();
}

// Function to open the gate
void openGate()
{
    myservo.write(GATE_OPEN_ANGLE);  // Move to 0 degrees (fully
open)
    delay(5000);  // Keep the gate open for 5 seconds
}
```

```
// Function to close the gate
void closeGate()
{
  myservo.write(GATE_CLOSED_ANGLE); // Move to 100 degrees (fully closed)
  delay(1000);  // Allow time for the servo to fully close
}

// Handle car entry
void handleEntry()
{
  if (Slot > 0)
  {
    Serial.println("Entry Gate Opening... Slot Remaining: " + String(Slot - 1));
    openGate();
    Slot--;
    updateLCD();

    // Update Blynk Gauge to show occupied slots
    Blynk.virtualWrite(V1, Slot);



    // Check if parking is full
    if (Slot == 0) {
      Blynk.virtualWrite(V3, 255);  // Turn ON Red LED (Parking Full)
      Blynk.virtualWrite(V5, 0);    // Turn OFF Green LED
    } else {
      Blynk.virtualWrite(V3, 0);    // Turn OFF Red LED
      Blynk.virtualWrite(V5, 255);  // Turn ON Green LED (Slots Available)
    }

    Serial.println("Entry Gate Closing...");
    closeGate();
  }
  else
  {
```

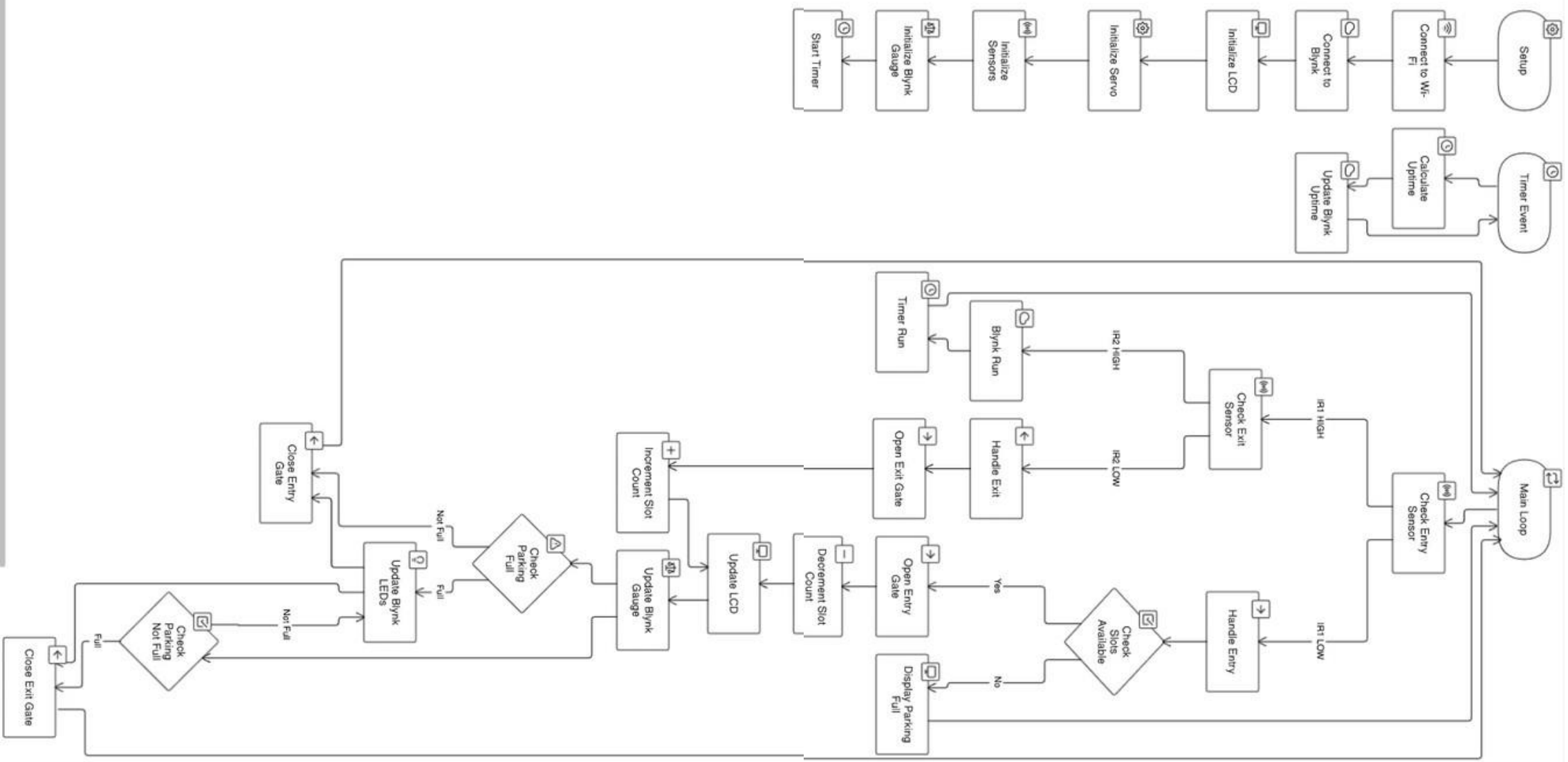
```
        lcd.clear();
        lcd.setCursor(0, 0);
        lcd.print("    SORRY :(    ");
        lcd.setCursor(0, 1);
        lcd.print("  Parking Full  ");
        Blynk.virtualWrite(V4, "Parking Full");  // Send message to LCD
widget
    }
}

// Handle car exit
void handleExit()
{
    Serial.println("Exit Gate Opening...");
    openGate();

    if (Slot < MAX_SLOTS) // Prevent slot overflow
    {
        Slot++;
    }

    updateLCD();

    // Update Blynk Gauge to show occupied slots
    Blynk.virtualWrite(V1, Slot);



    // Check if parking is no longer full
    if (Slot > 0) {
        Blynk.virtualWrite(V3, 0);    // Turn OFF Red LED (Parking Full)
        Blynk.virtualWrite(V5, 255);  // Turn ON Green LED (Slots Available)
    } else {
        Blynk.virtualWrite(V5, 0);    // Turn OFF Green LED
    }

    Blynk.virtualWrite(V4, "Car Exited. Slots Left: " + String(Slot)); // Send
update to Blynk LCD
```

```
    Serial.println("Exit Gate Closing...");
    closeGate();
}

// Update LCD display
void updateLCD()
{
    lcd.clear();
    lcd.setCursor(0, 0);
    lcd.print("    WELCOME!     ");
    lcd.setCursor(0, 1);
    lcd.print("Slot Left: ");
    lcd.print(Slot);
}

// Blynk Timer Event
void myTimerEvent()
{
    long uptime = millis() / 1000;
    Blynk.virtualWrite(V2, uptime);
}
```

IOT APP INTERFACE



FLOWCHART


Setup
Connect to Wi-Fi
Connect to Blynk
Initialize LCD
Initialize Servo
Initialize Sensors
Initialize Blynk Gauge
Start Timer
Timer Event
Calculate Uptime
Update Blynk Uptime
Main Loop
Check Entry Sensor
IR1 HIGH
IR1 LOW
Handle Entry
Check Slots Available
Yes
No
Open Entry Gate
Display Parking Full
Decrement Slot Count
Update LCD
Update Blynk Gauge
Check Exit Sensor
IR2 HIGH
IR2 LOW
Handle Exit
Open Exit Gate
Increment Slot Count
Timer Run
Blynk Run
Close Entry Gate
Check Parking Full
Not Full
Full
Update Blynk LEDs
Check Parking Not Full
Not Full
Full
Close Exit Gate