SMART PARKING

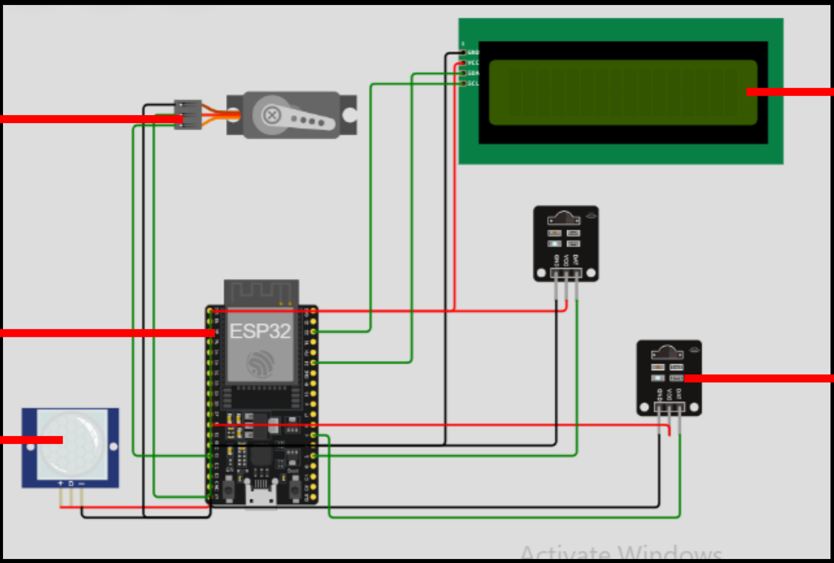
OBJECTIVE:

This Smart Parking project aims to provide real-time parking availability information to drivers , alleviating parking issues and making the parking experience more efficient. This is achieved by using IoT sensors connected to an ESP32 microcontroller to collect data about parking space occupancy, and then displaying this information on a mobile app developed using the Blynk framework.

IOT SENSORS SETUP

Sensors and microcontrollers used:

- ESP32 microcontroller
- IR sensor
- PIR sensor
- Servo motor
- LCD Display



LCD DISPLAY

SERVO MOTOR

ESP32 MICROCONTROLLER

IR SENSOR

PIR SENSOR

ESP32 Microcontroller: Controls the system and communicates with sensors and the mobile app.

IR Sensors: Detect vehicle presence in parking spaces.

Smoke Detector: Detects smoke and fire hazards.

Buzzer: Sounds an alarm when smoke is detected.

LCD Display: Shows real-time parking availability and fire alerts.

Servo Motor: Manages parking space access.

MOBILE APP DEVELOPMENT

BLYNK:
Blynk is an IoT platform that helps in creating mobile apps for controlling and monitoring hardware devices. It offers a user-friendly interface, cloud connectivity, and a wide range of compatible hardware.

CODE FOR CONNECTING WOKWI WITH BLYNK:

```
#define BLYNK_TEMPLATE_ID "TMPL3Sg1thgeA"
#define BLYNK_TEMPLATE_NAME "smartparking"
#define BLYNK_AUTH_TOKEN "***********"
#include <Wire.h>
#include <BlynkSimpleEsp2>
#include <LiquidCrystal_I2C.h>
LiquidCrystal_I2C lcd(0x27, 16, 2);  // Change the HEX address

#include <ESP32Servo.h>
Servo myservo1;

int IR1 = 2;
int IR2 = 4;
int SmokeDetectorPin = 5;  // Digital pin for the smoke detector
int BuzzerPin = 7;         // Digital pin for the buzzer

int Slot = 4;  // Enter Total number of parking Slots

bool flag1 = false;
bool flag2 = false;

unsigned long lastLcdUpdate = 0;  // Variable to track the time of the last LCD update
unsigned long lcdUpdateInterval = 1000;  // Update the LCD every 1000 milliseconds (1
second)
```

```
char auth[] = "*************";
char ssid[] = "temp";
char pass[] = "temp12345";

void setup() {
  lcd.init();
  lcd.begin(16, 2);
  lcd.backlight();
  pinMode(IR1, INPUT);
  pinMode(IR2, INPUT);
  pinMode(SmokeDetectorPin, INPUT);
  pinMode(BuzzerPin, OUTPUT);

  myservo1.attach(13);
  myservo1.write(100);

  lcd.setCursor(0, 0);
  lcd.print("     ESP32   ");
  lcd.setCursor(0, 1);
  lcd.print(" PARKING SYSTEM ");
  delay(2000);
  lcd.clear();

  Serial.begin(9600);   }

void loop() {
  if (digitalRead(IR1) == LOW && !flag1) {
    if (Slot > 0) {
      flag1 = true;
      if (!flag2) {
        myservo1.write(0);
        Slot--;
      }
    } else {
     Blynk.virtualwrite ("    SORRY :(    ", "  Parking Full  ");
    }
  }

  if (digitalRead(IR2) == LOW && !flag2) {
    flag2 = true;
    if (!flag1) {
      myservo1.write(0);
      Slot++;
    }
  }

  if (flag1 && flag2) {
    delay(1000);
    myservo1.write(100);
    Serial.println("Servo returned to the initial position.");
    flag1 = false;
    flag2 = false;
  }

  if (millis() - lastLcdUpdate >= lcdUpdateInterval) {
    updateLcdDisplay();
    lastLcdUpdate = millis();
  }
  Blynk.virtualWrite(V1, Slot);
}

void updateLcdDisplay() {
  if (digitalRead(SmokeDetectorPin) == HIGH) {
   Blynk.virtualwrite ("   WARNING!   ", " Smoke Detected ");
    digitalWrite(BuzzerPin, HIGH);  // Turn on the buzzer
  } else {
   Blynk.virtualwrite ("    WELCOME!    ", "Slot Left: " + String(Slot));
    digitalWrite(BuzzerPin, LOW);   // Turn off the buzzer
  }
}
void Blynk.virtualwrite (const char *line1, const String &line2) {
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print(line1);
  lcd.setCursor(0, 1);
  lcd.print(line2);
Blynk.run()
}
```

EXPLANATION:

Setup:

This code initializes the ESP32, LCD, sensors, Buzzer, and the Blynk communication.

Loop:

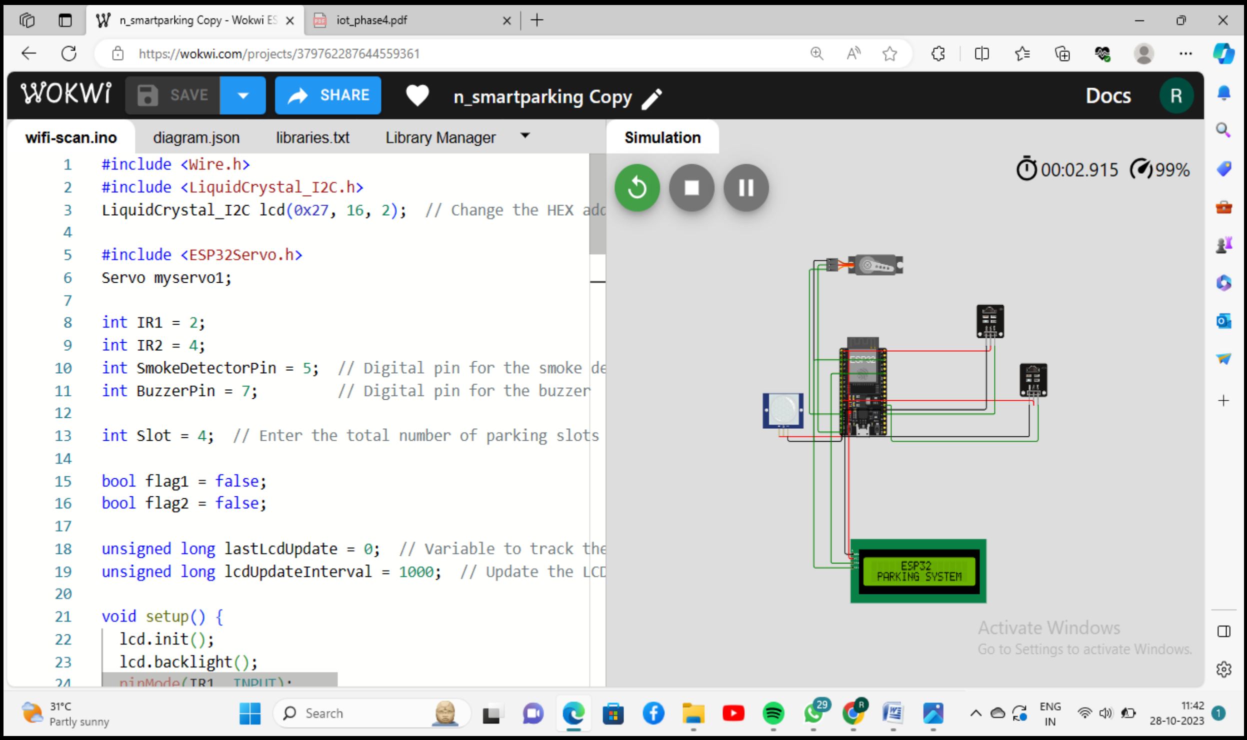
- The main loop checks the status of IR sensors, Smoke Detector, and manages parking availability.
- If a vehicle is detected by IR1 and a parking space is available (Slot > 0), it decreases the Slot count and displays available spaces.
- If a vehicle is detected by IR2, it increases the Slot count.
- It periodically updates the LCD display and checks the Smoke Detector for fire hazards.

LCD Display:

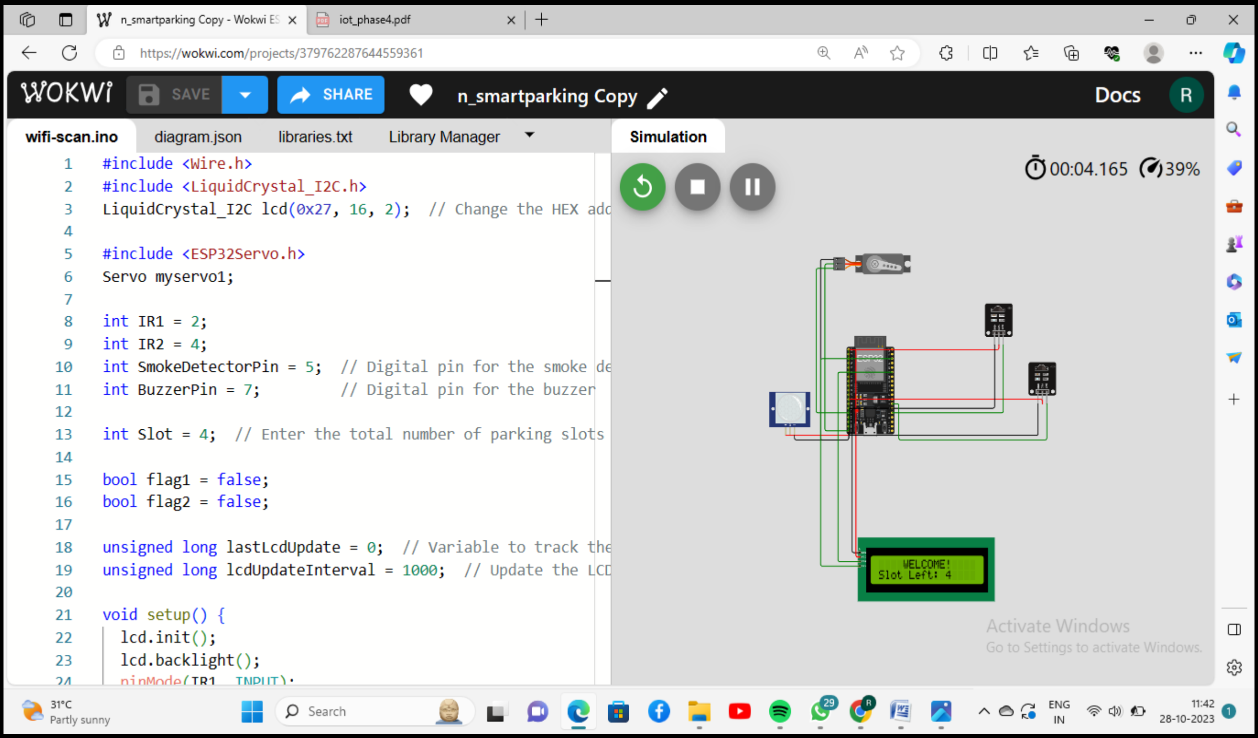
This function updates the LCD display with one of two messages:

Display Messages:

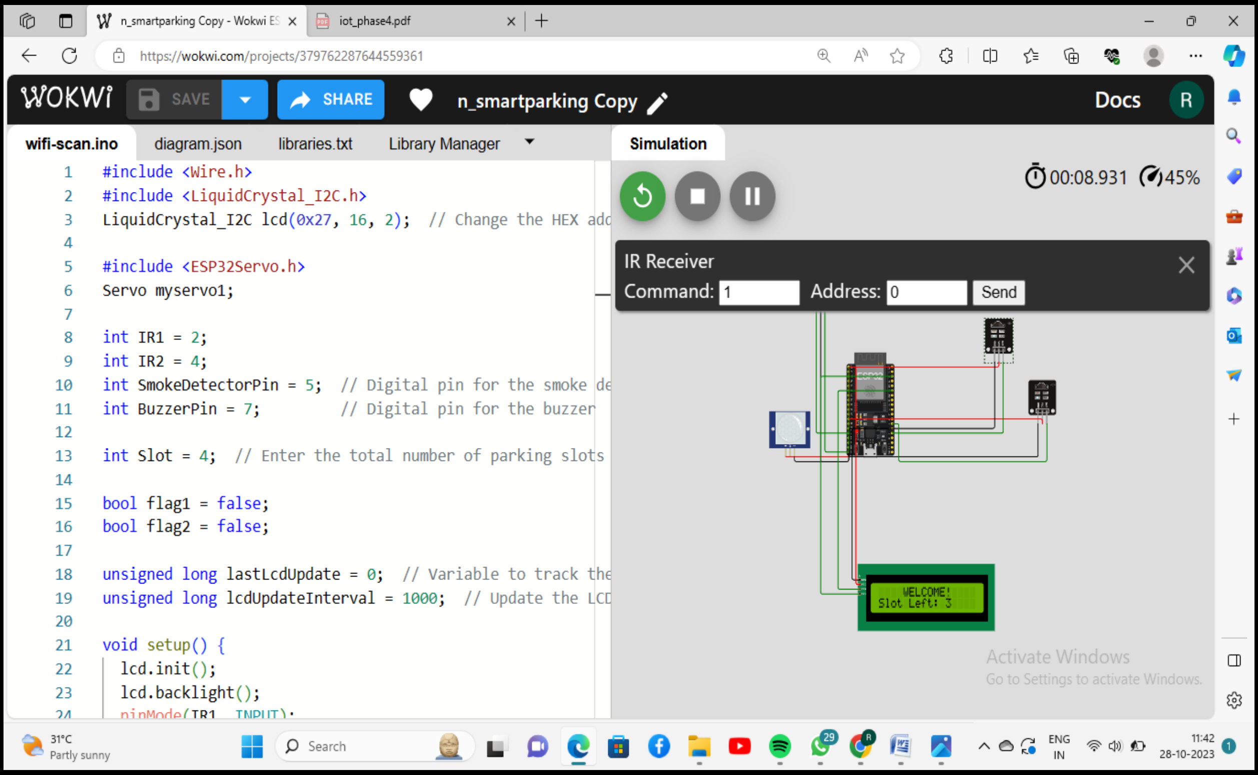
- "WELCOME!": Displayed when parking is available, showing the number of available slots.
- "SORRY :(": Displayed when all parking slots are occupied, indicating that parking is full.
- If the Smoke Detector detects smoke, it displays "WARNING! Smoke Detected" and activates the Buzzer.
- If no smoke is detected, it displays "WELCOME!" and the number of available parking slots.
- The code helps manage parking availability and safety by using IoT sensors and a mobile app for real-time updates, enhancing the overall parking .



```
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
LiquidCrystal_I2C lcd(0x27, 16, 2);  // Change the HEX add

#include <ESP32Servo.h>
Servo myservo1;

int IR1 = 2;
int IR2 = 4;
int SmokeDetectorPin = 5;  // Digital pin for the smoke de
int BuzzerPin = 7;         // Digital pin for the buzzer

int Slot = 4;  // Enter the total number of parking slots

bool flag1 = false;
bool flag2 = false;

unsigned long lastLcdUpdate = 0;  // Variable to track the
unsigned long lcdUpdateInterval = 1000;  // Update the LCD

void setup() {
  lcd.init();
  lcd.backlight();
  pinMode(IR1, INPUT);
```

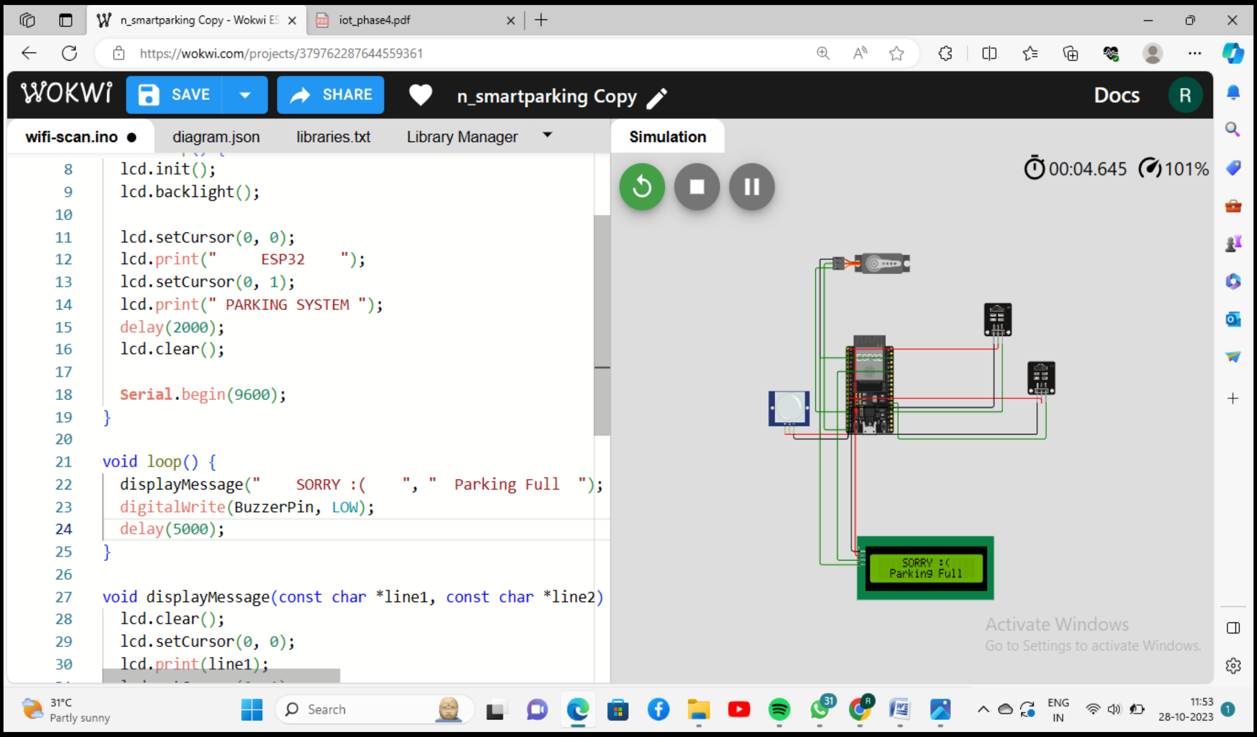


```
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
LiquidCrystal_I2C lcd(0x27, 16, 2);  // Change the HEX add

#include <ESP32Servo.h>
Servo myservo1;

int IR1 = 2;
int IR2 = 4;
int SmokeDetectorPin = 5;  // Digital pin for the smoke de
int BuzzerPin = 7;         // Digital pin for the buzzer

int Slot = 4;  // Enter the total number of parking slots

bool flag1 = false;
bool flag2 = false;

unsigned long lastLcdUpdate = 0;  // Variable to track the
unsigned long lcdUpdateInterval = 1000;  // Update the LCD

void setup() {
  lcd.init();
  lcd.backlight();
  pinMode(IR1, INPUT);
```



Command: 1 Address: 0

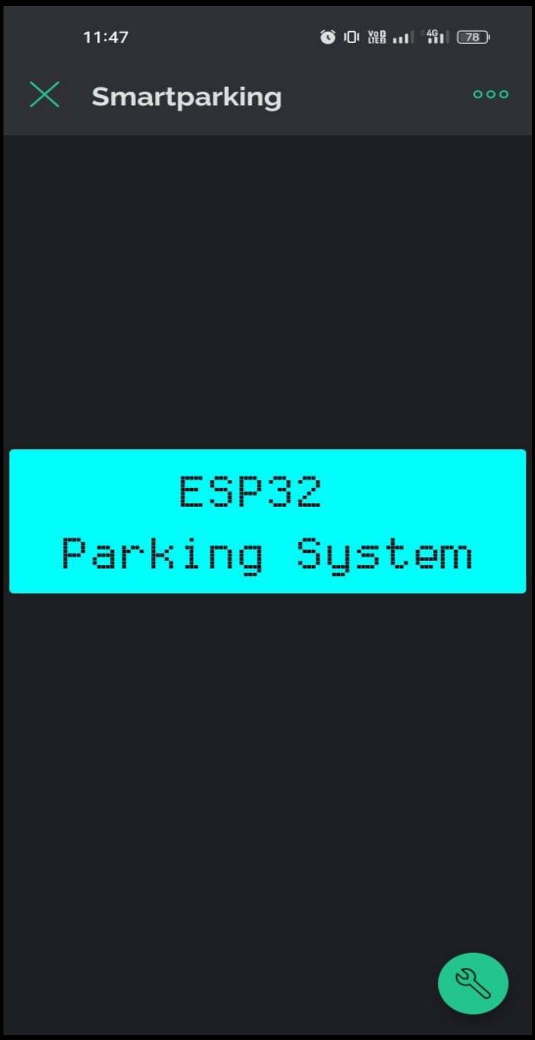
```
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
LiquidCrystal_I2C lcd(0x27, 16, 2);  // Change the HEX adc

#include <ESP32Servo.h>
Servo myservo1;

int IR1 = 2;
int IR2 = 4;
int SmokeDetectorPin = 5;  // Digital pin for the smoke de
int BuzzerPin = 7;         // Digital pin for the buzzer

int Slot = 4;  // Enter the total number of parking slots

bool flag1 = false;
bool flag2 = false;

unsigned long lastLcdUpdate = 0;  // Variable to track the
unsigned long lcdUpdateInterval = 1000;  // Update the LCD

void setup() {
  lcd.init();
  lcd.backlight();
  pinMode(IR1, INPUT);
```



```
  lcd.init();
  lcd.backlight();

  lcd.setCursor(0, 0);
  lcd.print("     ESP32     ");
  lcd.setCursor(0, 1);
  lcd.print(" PARKING SYSTEM ");
  delay(2000);
  lcd.clear();

  Serial.begin(9600);
}

void loop() {
  displayMessage("    SORRY :(    ", "  Parking Full  ");
  digitalWrite(BuzzerPin, LOW);
  delay(5000);
}

void displayMessage(const char *line1, const char *line2)
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print(line1);
```

MOBILE APPLICATION:

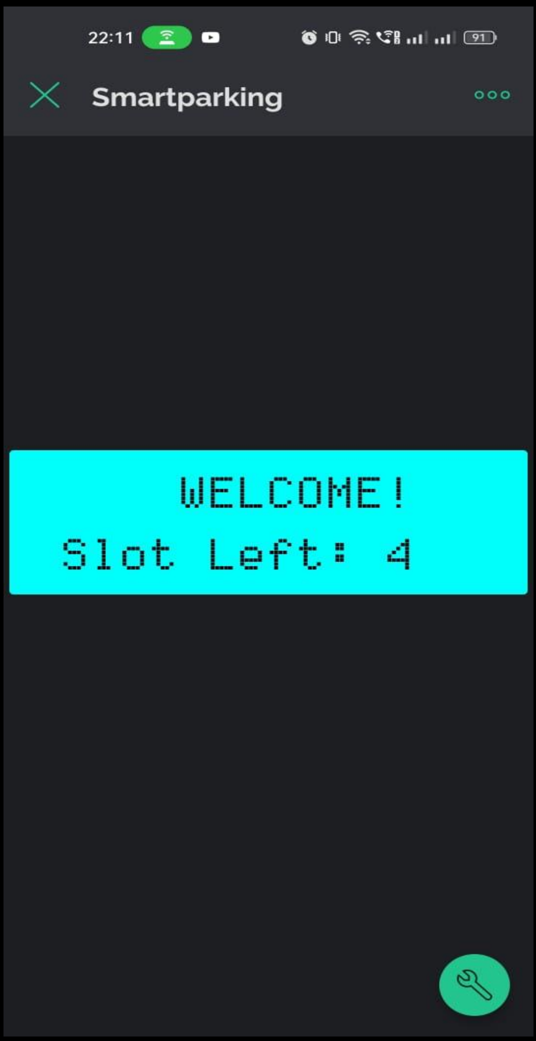
- We've developed a mobile application that is designed to be user-friendly and accessible via smartphones, providing a simple interface for users.
- The ESP32 microcontroller is equipped with sensors to collect real-time data about parking space occupancy. It processes the sensor data to determine whether parking spaces are available or occupied.
- Then connects to the mobile app through the Blynk platform, enabling real-time data communication.
- Users can request and view up-to-the-minute parking availability data in the mobile app ,enhancing the overall parking experience.



11:47

Smartparking

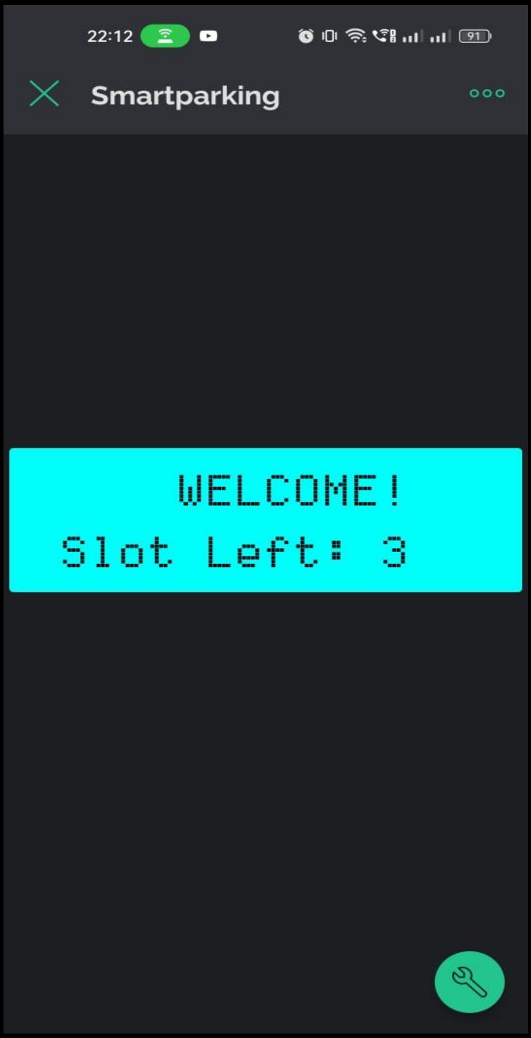
ESP32
Parking System



22:11

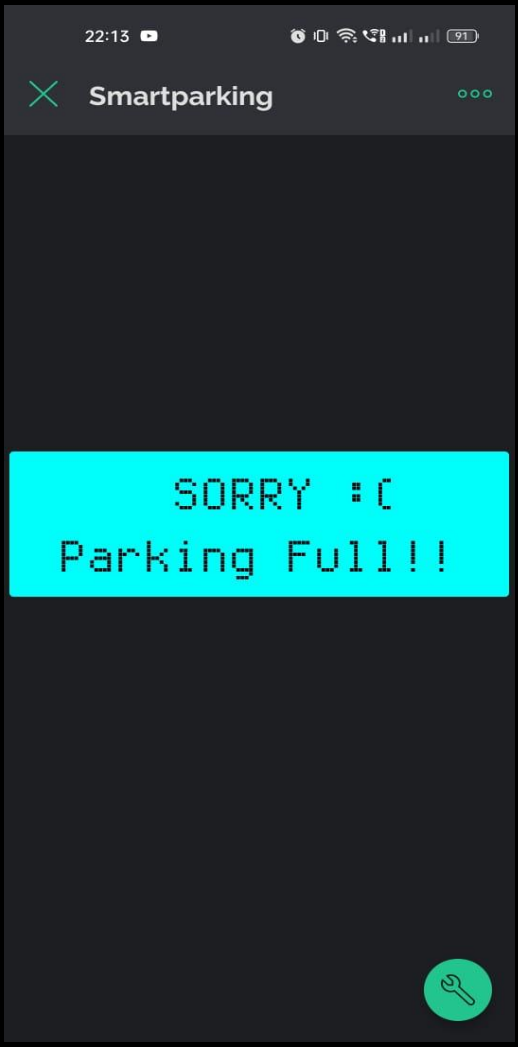
Smartparking

WELCOME!
Slot Left: 4



Smartparking

WELCOME!
Slot Left: 3



Smartparking

SORRY :(
Parking Full!!

BENEFITS OF SMART PARKING:

- Drivers can quickly find available parking spaces, reducing the time and fuel spent searching for parking, which is not only cost-effective but also environmentally friendly.
- Real-time data eliminates the uncertainty and frustration associated with finding parking, making the process less stressful and more convenient for drivers.
- Efficient parking reduces congestion and minimizes traffic jams, contributing to smoother traffic flow in busy areas.
- Monitoring and well-lit parking areas improve safety for drivers and passengers, reducing the vulnerability of parking lots to criminal activities.

Step by step instructions on how to replicate the project, deploy IoT sensors, develop the transit information platform, and integrating them.

Hardware Setup:

IoT Sensors placement:

1) Identify the parking spaces where you want to monitor availability.
2) Install Infrared (IR) sensors at each parking space to detect the presence of vehicles. These sensors can be placed on the ground to detect vehicle tires.

ESP32 Microcontroller:

1) Obtain an ESP32 microcontroller board.
2) Connect the IR sensors to the digital pins of the ESP32, ensuring they are appropriately powered and grounded.
3) Connect the Smoke Detector to a dedicated digital pin.
4) Connect a Buzzer to another digital pin for alarm purposes.

Software Setup:

Mobile App Development:

1) Install the Blynk app on your smartphone.
2) Create a new project in the Blynk app for your Smart Parking system.
3) Add widgets in the Blynk app for user interaction, such as a Button widget for requesting data and a Value Display widget for showing parking availability.
4) Generate a QR code for the Blynk project and scan it with your smartphone.
5) Enter the Blynk project's authentication token in the app's settings.
6) Configure widgets to communicate with the ESP32 through Blynk Cloud.

ESP32 Programming:

1) Develop the code for the ESP32 using Embedded C program language.
2) Include libraries for sensor control, Servo motor control, and Blynk for IoT communication.
3) Write code to collect data from the sensors and determine parking space availability.

4) Implement communication with the Blynk server using the Blynk project's authentication token.

Sensor Calibration:

Calibrate the IR sensors to accurately detect vehicle presence and absence.

Testing and Deployment:

Testing:

1) Test the system thoroughly by simulating different parking scenarios.
2) Verify that the mobile app correctly displays real-time parking availability data and responds to user requests.

Deployment:

1) Install the sensors and the ESP32 microcontroller at the chosen parking spaces.
2) Ensure that the ESP32 is connected to a Wi-Fi network for internet connectivity.