# Smart Parking System
# Phase 4: Development Part 2

## Introduction:

In the first phase of the project, we successfully established a connection between the ESP32 microcontroller, ultrasonic sensors, and an OLED display. The setup was designed to monitor multiple parking slots and display their occupancy status directly on the local OLED screen based on distance measurements obtained from the sensors.

In this second phase, we further enhance the project's functionality by integrating the Blynk platform, a versatile IoT service, into the system. Blynk allows us to visualize and monitor the parking slot statuses remotely using a mobile application. By connecting our ESP32-based system to Blynk, we'll be able to observe real-time data and receive updates on parking slot availability through the Blynk app on a smartphone or tablet.

The primary objectives of this phase are as follows:

1. Blynk Integration: Establish a connection between the ESP32, ultrasonic sensors, and Blynk to enable remote monitoring of parking slot statuses.
2. Real-time Visualization: Transmit the sensor data to the Blynk platform to display the availability status of parking slots in real time.
3. User-Friendly Interface: Use the Blynk mobile application to view, track, and receive notifications about parking slot occupancy.

- By combining the capabilities of the ESP32, ultrasonic sensors, OLED display, and Blynk, we aim to create an efficient and user-friendly parking monitoring system. This integration empowers users to remotely access and monitor parking slot statuses conveniently and in real time, offering a practical solution for various applications that require remote monitoring and management of resources**.**

## Blynk:

IoT Platform: Blynk is an IoT platform that simplifies the process of connecting hardware to the internet. It offers a drag-and-drop interface to create custom IoT applications and control connected devices from a smartphone or tablet.

User-Friendly Interface: Blynk provides an intuitive mobile app, allowing users to create a graphical interface with widgets to control hardware and visualize data. Widgets include buttons, sliders, graphs, and displays that can be easily configured and linked to hardware components.

Cloud Connectivity: Blynk offers cloud services, enabling remote access and control of connected devices from anywhere with an internet connection. Users can monitor, receive notifications, and control their devices through the Blynk app.

## Code:

```
#define BLYNK_TEMPLATE_ID "TMPL3HVPXZ_98"
#define BLYNK_TEMPLATE_NAME "Raspberry pi and blynk"
#define BLYNK_AUTH_TOKEN "_QWuaO56UA_DYjcyRDaiDm-6QLcpJNcA"
#define BLYNK_PRINT Serial
#include <WiFi.h>
#include <BlynkSimpleEsp32.h>
#include <Adafruit_SSD1306.h>
#include <Wire.h>
char auth[] = BLYNK_AUTH_TOKEN;
char ssid[] = "Wokwi-GUEST";
char pass[] = "";
#define SDA_PIN 22
#define SCL_PIN 23
#define SCREEN_WIDTH 128
#define SCREEN_HEIGHT 64
Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT);
BlynkTimer timer;
void setup() {
  Serial.begin(115200);
  pinMode(26, INPUT);
  pinMode(33, INPUT);
  pinMode(18, INPUT);
  pinMode(27, OUTPUT);
  pinMode(25, OUTPUT);
  pinMode(19, OUTPUT);
  Wire.begin(SDA_PIN, SCL_PIN);
  if (!display.begin(SSD1306_SWITCHCAPVCC, 0x3C)) {
    Serial.println("SSD1306 allocation failed");
    for (;;)
      ;
```

```
  }
  display.clearDisplay();
  display.setTextSize(2);
  Blynk.begin(auth, ssid, pass);
  timer.setInterval(1000L, sendSensorData);
}
void sendSensorData() {
  // Slot 1
  digitalWrite(27, LOW);
  delay(100);
  digitalWrite(27, HIGH);
  delay(100);
  digitalWrite(27, LOW);
  long duration1 = pulseIn(26, HIGH);
  long distance1 = duration1 * 0.0343 / 2;
  Serial.print("Distance 1 = ");
  Serial.println(distance1);
  delay(100);
  // Slot 2
  digitalWrite(25, LOW);
  delay(100);
  digitalWrite(25, HIGH);
  delay(100);
  digitalWrite(25, LOW);
  long duration2 = pulseIn(33, HIGH);
  long distance2 = duration2 * 0.0343 / 2;
  Serial.print("Distance 2 = ");
  Serial.println(distance2);
  delay(100);
  // Slot 3
```

```
  digitalWrite(19, LOW);
  delay(100);
  digitalWrite(19, HIGH);
  delay(100);
  digitalWrite(19, LOW);
  long duration3 = pulseIn(18, HIGH);
  long distance3 = duration3 * 0.0343 / 2;
  Serial.print("Distance 3 = ");
  Serial.println(distance3);

  if (distance1 < 200) {
    Blynk.virtualWrite(V1, "Parked");
} else {
    Blynk.virtualWrite(V1, "Free");
}
if (distance2 < 200) {
    Blynk.virtualWrite(V2, "Parked");
} else {
    Blynk.virtualWrite(V2, "Free");
}
if (distance3 < 200) {
    Blynk.virtualWrite(V3, "Parked");
} else {
    Blynk.virtualWrite(V3, "Free");
}
  displayStatus(distance1, distance2, distance3);
}
void displayStatus(int distance1, int distance2, int distance3) {
  display.clearDisplay();
  display.setTextSize(2);
```

```
  display.setTextColor(WHITE);
  display.setCursor(0, 0);
  display.print("1: ");
  if (distance1 < 200) {
    display.println("Parked");
  } else {
    display.println("Free");
  }
  display.setCursor(0, 20);
  display.print("2: ");
  if (distance2 < 200) {
    display.println("Parked");
  } else {
    display.println("Free");
  }
  display.setCursor(0, 40);
  display.print("3: ");
  if (distance3 < 200) {
    display.println("Parked");
  } else {
    display.println("Free");
  }
  display.display();
}
void loop() {
  Blynk.run();
  timer.run();
}
```

## Code Overview:

### Libraries Included:

```
#include <WiFi.h>

#include <BlynkSimpleEsp32.h>

#include <Adafruit_SSD1306.h>

#include <Wire.h>
```

The code includes essential libraries for Wi-Fi connectivity, Blynk functionality for ESP32, OLED display support via Adafruit_SSD1306, and the Wire library for I2C communication.

### Blynk Configuration:

```
#define BLYNK_TEMPLATE_ID "TMPL3HVPXZ_98"

#define BLYNK_TEMPLATE_NAME "Raspberry pi and blynk"

#define BLYNK_AUTH_TOKEN "_QWuaO56UA_DYjcyRDaiDm-6QLcpJNcA"

#define BLYNK_PRINT Serial
```

Defines Blynk template ID, name, and authentication token for the Blynk app, while also setting the print output to Serial for debugging.

### Global Variables and Objects:

```
char auth[] = BLYNK_AUTH_TOKEN;  // Blynk Auth Token

char ssid[] = "Wokwi-GUEST";     // WiFi SSID

char pass[] = "";                // WiFi password

#define SDA_PIN 22

#define SCL_PIN 23

#define SCREEN_WIDTH 128

#define SCREEN_HEIGHT 64

Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT);
```

Declares variables for Blynk authentication, Wi-Fi SSID, and password. Also, defines pin numbers for the OLED screen's I2C communication and initializes the Adafruit_SSD1306 display object.

**Setup Function:**

```
void setup() {
  Serial.begin(115200);
  pinMode(26, INPUT);
  pinMode(33, INPUT);
  pinMode(18, INPUT);
  pinMode(27, OUTPUT);
  pinMode(25, OUTPUT);
  pinMode(19, OUTPUT);
  Wire.begin(SDA_PIN, SCL_PIN);
  if (!display.begin(SSD1306_SWITCHCAPVCC, 0x3C)) {
    Serial.println("SSD1306 allocation failed");
    for (;;)
      ;
  }
  display.clearDisplay();
  display.setTextSize(2);
  Blynk.begin(auth, ssid, pass);
  timer.setInterval(1000L, sendSensorData);  // Send sensor data to Blynk every second
}
```

Configures the initial setup for serial communication, pin modes for sensors and actuators, initializes the OLED display, connects to Blynk using Wi-Fi credentials, and sets a timer to send sensor data at a specified interval.

**sendSensorData Function:**

```
void sendSensorData() {
  // Slot 1
  digitalWrite(27, LOW);
```

```
delay(100);
digitalWrite(27, HIGH);
delay(100);
digitalWrite(27, LOW);
long duration1 = pulseIn(26, HIGH);
long distance1 = duration1 * 0.0343 / 2;
Serial.print("Distance 1 = ");
Serial.println(distance1);
delay(100);
// Slot 2
digitalWrite(25, LOW);
delay(100);
digitalWrite(25, HIGH);
delay(100);
digitalWrite(25, LOW);
long duration2 = pulseIn(33, HIGH);
long distance2 = duration2 * 0.0343 / 2;
Serial.print("Distance 2 = ");
Serial.println(distance2);
delay(100);
// Slot 3
digitalWrite(19, LOW);
delay(100);
digitalWrite(19, HIGH);
delay(100);
digitalWrite(19, LOW);
long duration3 = pulseIn(18, HIGH);
long distance3 = duration3 * 0.0343 / 2;
Serial.print("Distance 3 = ");
```

```
  Serial.println(distance3);
  if (distance1 < 200) {
    Blynk.virtualWrite(V1, "Parked");
} else {
    Blynk.virtualWrite(V1, "Free");
}
if (distance2 < 200) {
    Blynk.virtualWrite(V2, "Parked");
} else {
    Blynk.virtualWrite(V2, "Free");
}
if (distance3 < 200) {
    Blynk.virtualWrite(V3, "Parked");
} else {
    Blynk.virtualWrite(V3, "Free");
}
  displayStatus(distance1, distance2, distance3);
}
```

Performs measurements from ultrasonic sensors for three parking slots. It calculates distances, updates corresponding Blynk virtual pins with the parking status, and triggers the OLED display update with the parking slot statuses.

**displayStatus Function:**

```
void displayStatus(int distance1, int distance2, int distance3) {
  display.clearDisplay();
  display.setTextSize(2);
  display.setTextColor(WHITE);
  display.setCursor(0, 0);
  display.print("1: ");
  if (distance1 < 200) {
```

```
  display.println("Parked");
 } else {
  display.println("Free");
 }
 display.setCursor(0, 20);
 display.print("2: ");
 if (distance2 < 200) {
  display.println("Parked");
 } else {
  display.println("Free");
 }
 display.setCursor(0, 40);
 display.print("3: ");
 if (distance3 < 200) {
  display.println("Parked");
 } else {
  display.println("Free");
 display.display();
}
```

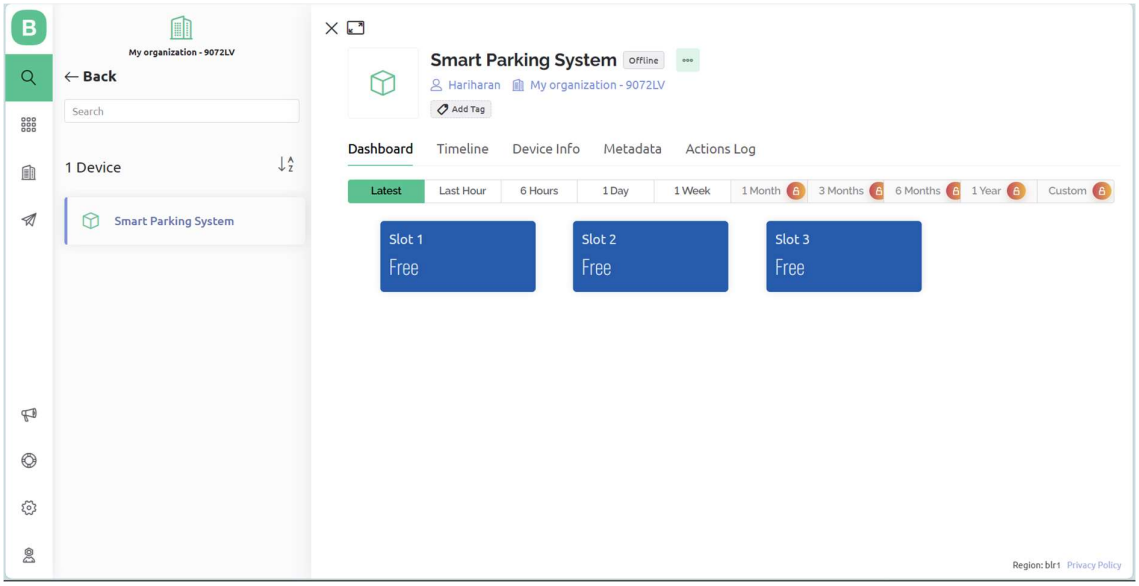
Clears the OLED display and showcases the parking status for each slot based on the provided distances whether it is Parked or Free

**Loop Function:**

```
void loop() {
 Blynk.run();
 timer.run();
}
```

Handles continuous execution of Blynk and the timer for sensor data transmission.

## Blynk Platform:




## Circuit Diagram: