**SMART PARKING**

**PHASE 2 : INNOVATION**

Creating a Smart Parking project for ESP32 on the Wokwi platform with mobile application integration, data logging, and alerts involves several components and steps.

**COMPONENTS NEEDED:**

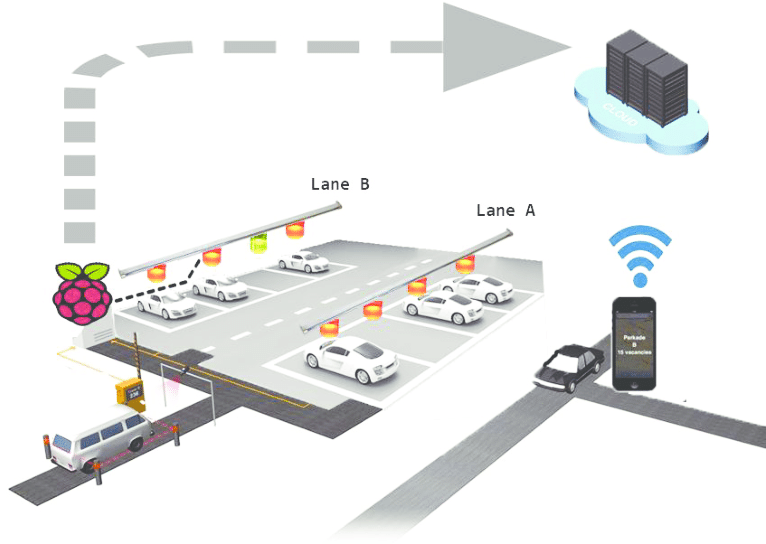
1. Ultrasonic distance sensors (HC-SR04) for each parking space

2. ESP32 development board

3. Breadboard and jumper wires

4. Mobile application development platform

5. Wokwi virtual simulator (https://wokwi.com/)

****

**PROJECT STEPS:**

1. Hardware Setup:

a. Connect the HC-SR04 ultrasonic sensors to ESP32 board, one sensor per parking space.

b. Wire the HC-SR04 sensors.

- VCC to 5V on ESP32

- GND to GND on ESP32

- Trig to a digital GPIO pin on ESP32 (e.g., GPIO2)

- Echo to another digital GPIO pin on ESP32 (e.g., GPIO4)

2. Programming:

a. An Arduino sketch for the ESP32 to read distance data from ultrasonic sensors:

The code to include data logging and alerts:

```cpp

#include <Ultrasonic.h>

Ultrasonic sensor1(GPIO\_TRIGGER1, GPIO\_ECHO1);

Ultrasonic sensor2(GPIO\_TRIGGER2, GPIO\_ECHO2);

const int thresholdDistance = 30;

// Define the threshold distance for occupancy

void setup()

{

Serial.begin(115200);

}

void loop() {

long distance1 = sensor1.read();

long distance2 = sensor2.read();

// Process distance data and manage parking spaces

boolean isSpace1Occupied = distance1 < thresholdDistance;

boolean isSpace2Occupied = distance2 < thresholdDistance;

// Data Logging (send data to a server or store locally)

// Implement alert logic based on occupancy status

delay(1000); // Delay for better readability

}

```

b. In the loop function, process the distance data from each sensor to determine whether a parking space is occupied or vacant. We can set a threshold distance to decide when a space is occupied.

c. We want to use a data structure to keep track of the parking space status.

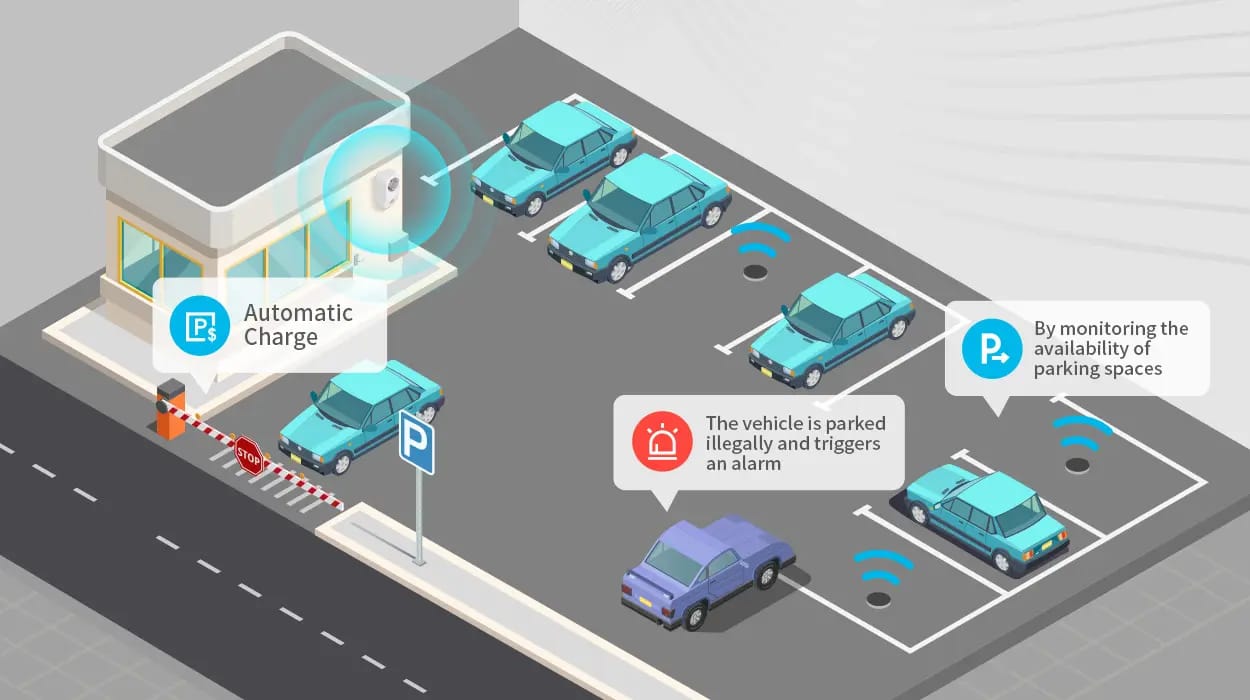
3. Mobile App Development:

a. Create a mobile application using a platform like MIT App Inventor, which allows easy integration with ESP32 via Bluetooth or Wi-Fi.

b. Design the application's user interface to display real-time parking space occupancy status and alerts.

c. Implement communication with the ESP32 board. We can use Wi-Fi or Bluetooth, depending on the requirements and hardware capabilities.

d. Set up real-time data updates and alerts within the application. For example, we can notify users when a parking space becomes vacant or when all spaces are occupied.



4. Visualization on Wokwi:

a. Go to the Wokwi platform (https://wokwi.com/) and create a new project.

b. Select the ESP32 as our target board.

c. Import the Arduino sketch that we created earlier into the Wokwi editor.

d. Use the virtual interface provided by Wokwi to display the parking space status.We can use LEDs or graphical elements to represent the parking spaces.

5. Testing:

a. Simulate the project on Wokwi to ensure that the parking space status updates are reflected in the virtual interface.

b. Test the mobile application with the ESP32 to verify real-time data updates and alerts.

6. Deployment:

a. Once we are confident that the system works as expected in the virtual environment, we can deploy it to a physical ESP32 board and connect it to real sensors in a parking area.

a. Depending on our project's requirements, we can further enhance it by adding features such as user registration, payment integration, and more advanced data analytics.

By following these steps,we can create a Smart Parking system that not only detects and manages parking spaces but also provides real-time updates and alerts to users via a mobile application.