Smart parking System

Phase 5

Project Objectives:

The primary objective of this project is to create a real-time parking availability system using IoT sensors, specifically ultrasonic sensors interfaced with an ESP32 microcontroller. The project aims to:

IoT Sensor Setup:

The IoT sensor setup includes ultrasonic sensors positioned in individual parking slots. These sensors detect the presence of vehicles by measuring the distance between the sensor and the car. The ESP32 microcontroller processes the sensor data to determine the parking slot's availability.

Mobile App Development (Blynk Integration):

The Blynk mobile app serves as a user interface for monitoring parking availability remotely. The app displays real-time data about each parking slot's status (occupied or vacant) using virtual pins linked to the ESP32. Users can view parking availability, monitor status changes in real time through the app.

ESP32 Integration:

The ESP32 acts as the central controller that processes data from the ultrasonic sensors. It interfaces with the Blynk platform to transmit real-time parking availability data. The ESP32 also controls the OLED display to locally exhibit the parking status.

Code Implementation:

The code written for the ESP32 integrates the sensor readings, Blynk API, and display control. It collects data from sensors, communicates with the Blynk platform using Wi-Fi connectivity, updates virtual pins on Blynk, and simultaneously displays parking availability on the OLED screen.

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;
```

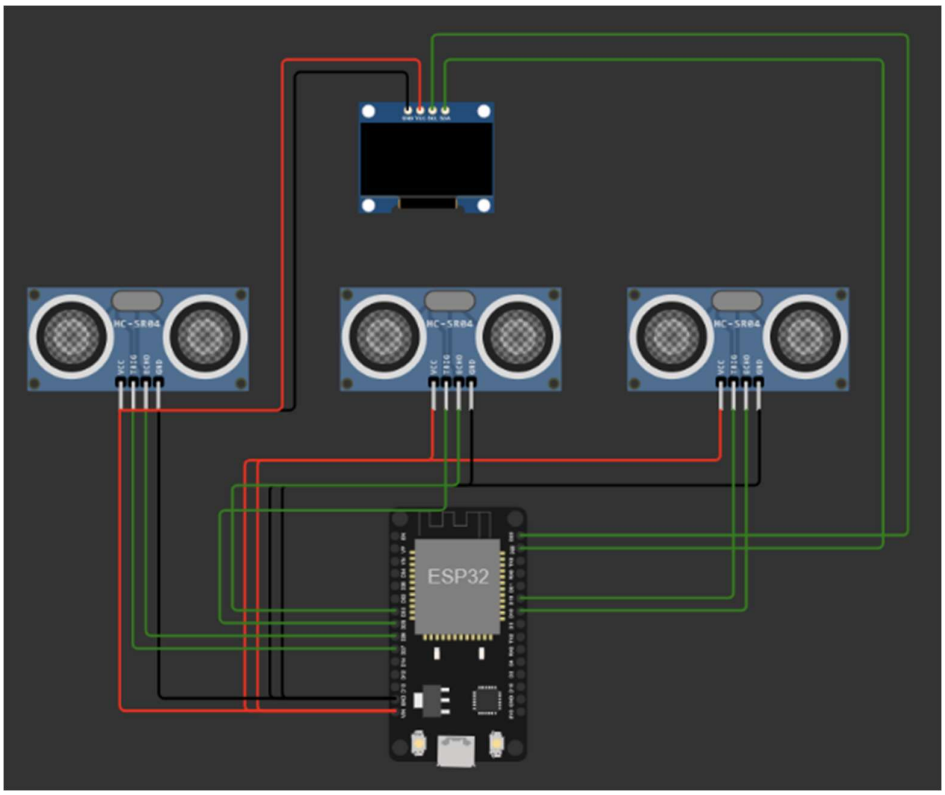
```
char ssid[] = "Wokwi-GUEST";
char pass[] = "";
#define SDA_PIN 22
#define SCL_PIN 23
#define SCREEN_WIDTH 128
#define SCREEN_HEIGHT 64
Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT);
BlynkTimer timer;
void setup() {
  Serial.begin(115200);
  pinMode(26, INPUT);
  pinMode(33, INPUT);
  pinMode(18, 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);  // Send sensor data to Blynk every second
}
void sendSensorData() {
  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);
  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);
  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 {
```

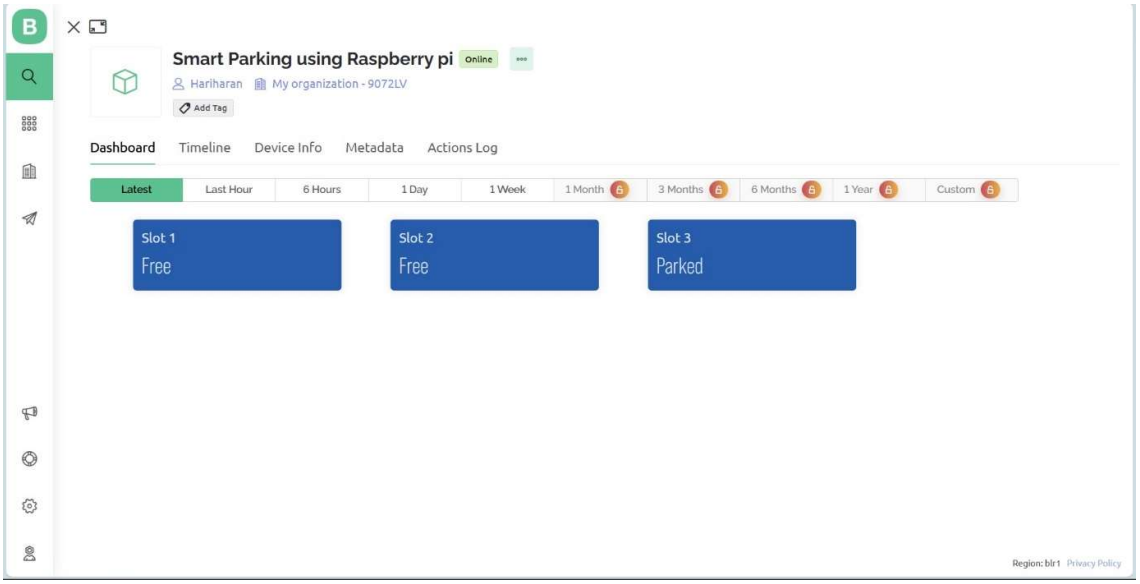
```
    Blynk.virtualWrite(V1, "Free");
  }
  if (distance2 < 200) {
    Blynk.virtualWrite(V2, "Parked");
  } else {
    Blynk.virtualWrite(V2, "Free");
  }
  if (distance3 < 200) {
    Blynk.virtualWrite(V3, "Parked");
  } else {
    Blynk.virtualWrite(V3, "Free");
  }

  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();
}
```

Diagram:



Blynk:



Benefits of Real-Time Parking Availability System:

1. Enhanced Parking Management:

Efficient Space Utilization: By remotely accessing real-time parking availability information through the Blynk app, drivers can swiftly identify vacant parking spaces. This efficient management of parking spots helps in optimizing space usage within parking facilities.

2. Reduced Search Time:

Quick Spot Identification: Access to immediate parking availability information significantly decreases the time spent searching for parking. Drivers can identify available spots more rapidly, reducing the frustration and stress often associated with parking in congested areas.

3. Improved Traffic Flow:

Minimized Congestion: Swift identification of available parking spaces reduces unnecessary circling or queuing to find a spot. This reduction in time spent searching for parking contributes to smoother traffic flow and diminishes traffic congestion around parking areas.

4. Environmental Benefits:

Reduced Emissions: Minimizing the time spent looking for parking spots results in less idling and driving around in search of a space. This reduction in unnecessary vehicle movement leads to decreased fuel consumption and emissions, contributing to a cleaner and less polluted environment.

5. User Convenience:

Real-Time Updates: Blynk's interface provides drivers with real-time updates on parking slot availability, allowing them to plan their parking before arriving at a location. This convenience adds a level of predictability and comfort to their parking experience.

6. Accessibility and Monitoring:

Remote Monitoring: The Blynk mobile app allows for remote monitoring of parking availability. Drivers can check parking availability from a distance, aiding them in making informed decisions about their parking strategy even before reaching a destination.

7. Alleviating Parking Issues:

Addressing Parking Challenges: The system mitigates common parking issues like extensive search times, uncertainty about parking availability, and traffic congestion caused by drivers circling in search of parking spaces.

Submission:

GitHub link: https://github.com/hari-xd/IOT.git

Wokwi project link: https://wokwi.com/projects/380113876889875457Steps to replicate the projecct

Steps to replicate the project:

Step 1: Gathering Hardware

1. **ESP32 Development Board:** Acquire an ESP32 microcontroller (like the one specified in the diagram).
2. **Ultrasonic Sensors (HC-SR04):** Obtain three HC-SR04 ultrasonic sensors.
3. **OLED Display:** Get an SSD1306 OLED display.
4. **Wires, Breadboard, Power Source**: Prepare jumper wires, a breadboard, and a suitable power source (USB cable, battery) for the ESP32.

Step 2: Connections:

Serial Communication (ESP32 and Serial Monitor):

- ESP32 TX0 to Serial Monitor RX (with unspecified details for the connection)

Power and Ground Connections:

- Ultrasonic Sensor 1 (VCC, GND) to ESP32 VIN and GND
- Ultrasonic Sensor 2 (VCC, GND) to ESP32 VIN and GND
- Ultrasonic Sensor 3 (VCC, GND) to ESP32 VIN and GND
- Ultrasonic Sensor 3 GND to OLED1 GND
- Ultrasonic Sensor 3 VCC to OLED1 VCC

Trigger and Echo Pins Connections:

- Ultrasonic Sensor 1 (TRIG, ECHO) to ESP32 digital pins (D25, D33)
- Ultrasonic Sensor 2 (TRIG, ECHO) to ESP32 digital pins (D19, D18)
- Ultrasonic Sensor 3 (TRIG, ECHO) to ESP32 digital pins (D27, D26)

OLED Display Connections:

- OLED1 SCL to ESP32 digital pin D23
- OLED1 SDA to ESP32 digital pin D22

Includes the necessary libraries (e.g., **WiFi.h, BlynkSimpleEsp32.h, Adafruit_SSD1306.h**).

Establishes the connections and functionality for the ultrasonic sensors, OLED display, and Blynk integration (for remote monitoring).

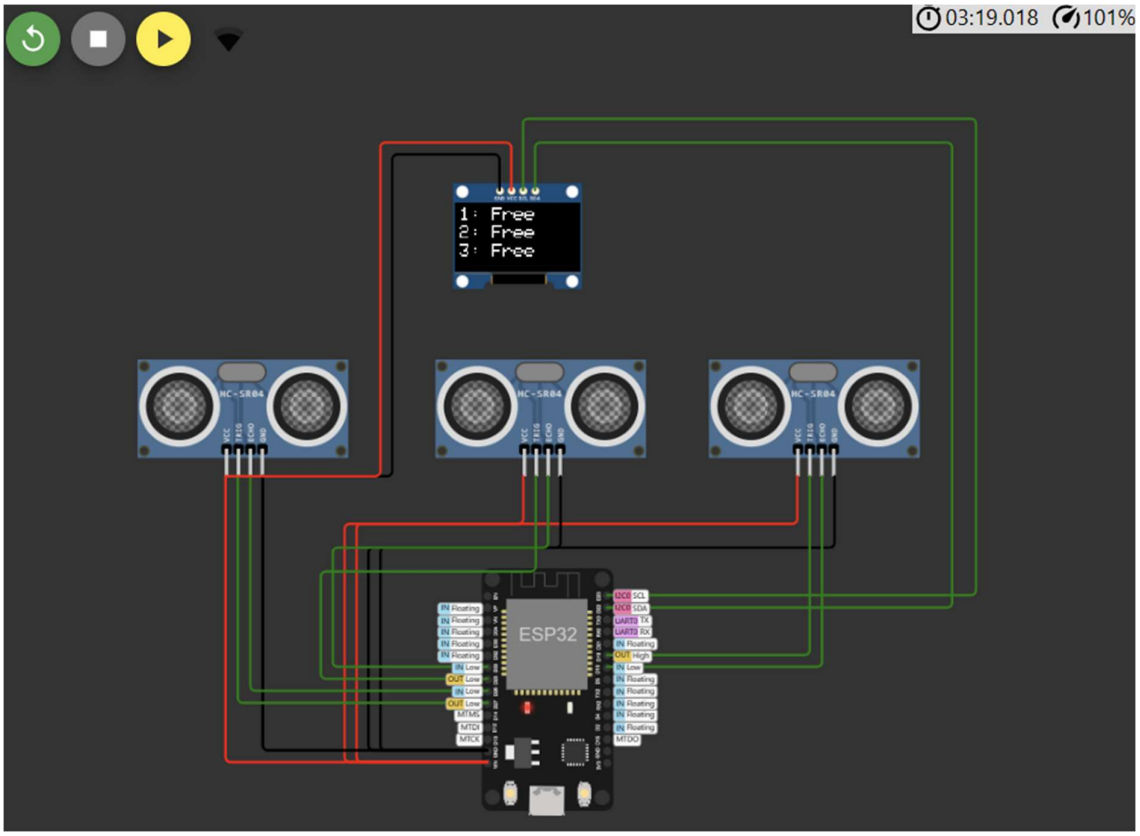
Utilizes Blynk to send the sensor data to the cloud and updates the OLED display based on the sensor readings.

Step 4: Testing and Integration

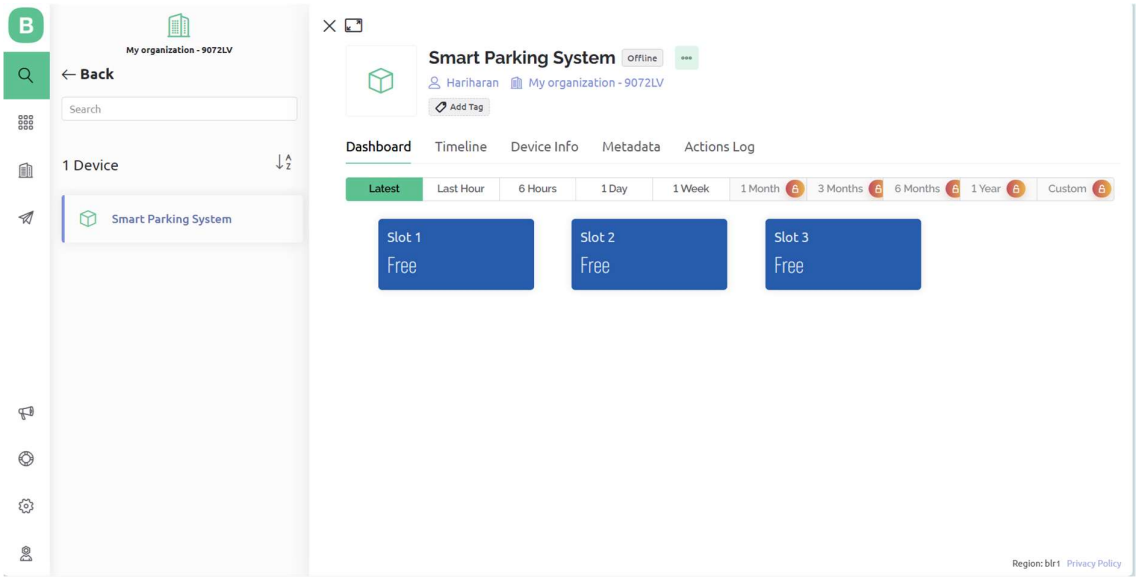
1. **Compile and Upload Code:** Use the Arduino IDE or compatible software to compile the code and upload it to the ESP32.
2. **Check Connections:** Ensure all connections match the wiring specified in the diagram.json.
3. **Monitor Outputs**: Observe the serial monitor for sensor readings and verify if the OLED display shows the parking status.

Connecting Blynk: Go to blynk and create a template and add a device, then blynk generates a blynk auth token, template-id, template-name. Using that link Blynk and the ESP32

Data Transmission:



Blynk UI:


My organization - 9072LV
Back
Search
1 Device
Smart Parking System
Smart Parking System
Offline
Hariharan
My organization - 9072LV
Add Tag
Dashboard
Timeline
Device Info
Metadata
Actions Log
Latest
Last Hour
6 Hours
1 Day
1 Week
1 Month
3 Months
6 Months
1 Year
Custom
Slot 1
Free
Slot 2
Free
Slot 3
Free
Region: blr1
Privacy Policy