SMART PARKING

PHASE_3: DEVELOPMENT PART_1

INTRODUCTION:

In today's rapidly urbanizing world, the demand for parking spaces is ever-increasing, leading to congestion, pollution, and frustration among urban dwellers. Smart parking, empowered by the Internet of Things (IoT), offers a revolutionary solution to these challenges. By integrating IoT devices, sensors, and real-time data analysis, smart parking systems transform the way we approach parking management.

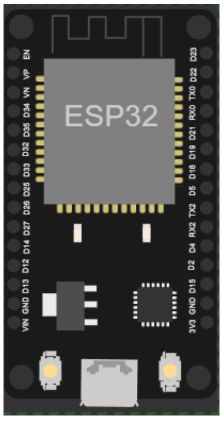
COMPONENTS REQUIRED:

1)ESP32 Microcontroller

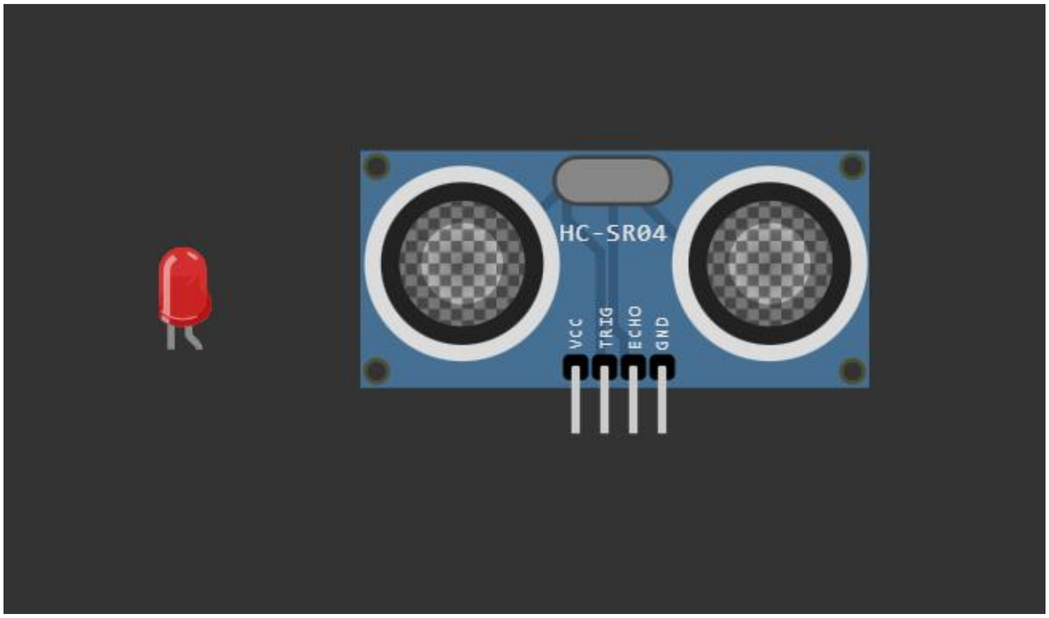
2)Ultrasonic Distance Sensors

3)LED Displays or Indicators

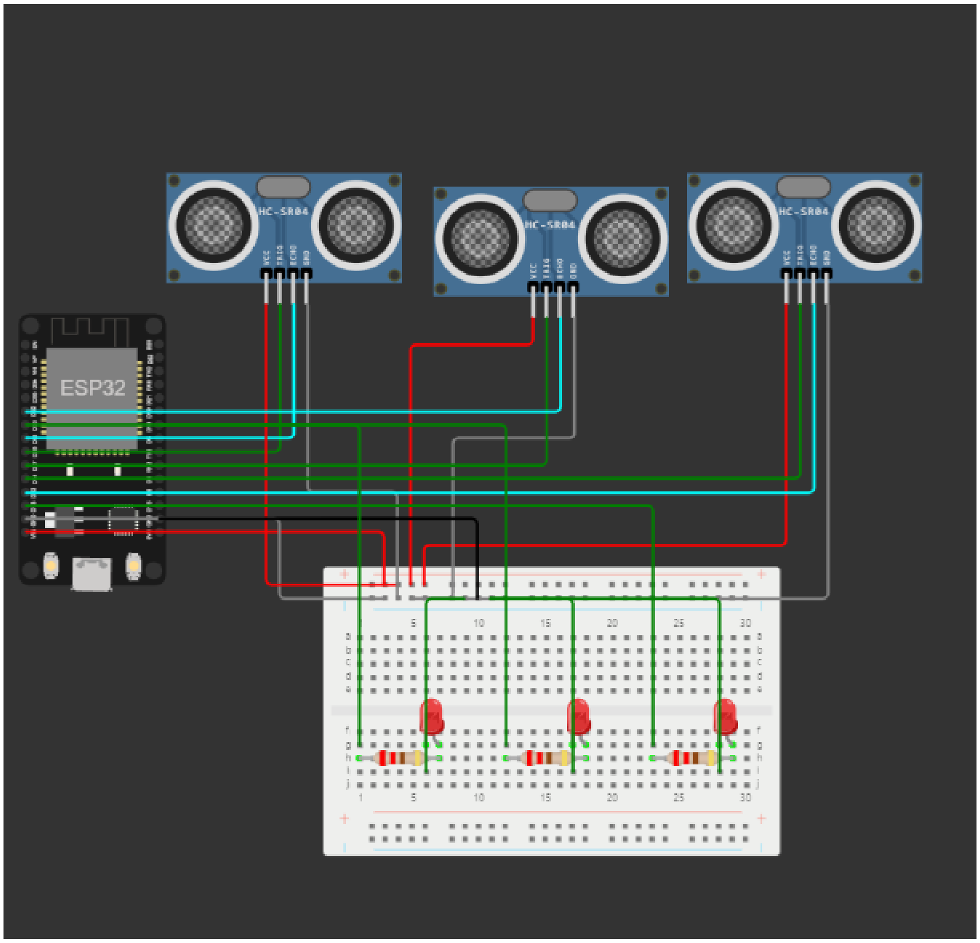
STEP-1: ESP32



STEP-2: SENSORS



STEP-3: ASSEMBLY



SOFTWARE DEVELOPMENT:

Sensor Data Acquisition:

Development of code to interface with the IoT sensors (e.g., ultrasonic or magnetic sensors) to collect data about parking space occupancy.

Microcontroller Programming:

By writing firmware for the microcontroller (e.g., ESP32) to process data from the sensors.

By using an integrated development environment (IDE) like the Arduino IDE or Platform IO to write and upload code to the microcontroller.

Data Processing:

By Implementing logic for processing sensor data to determine parking space occupancy.

Have to use algorithms to filter and smooth sensor data to reduce false readings.

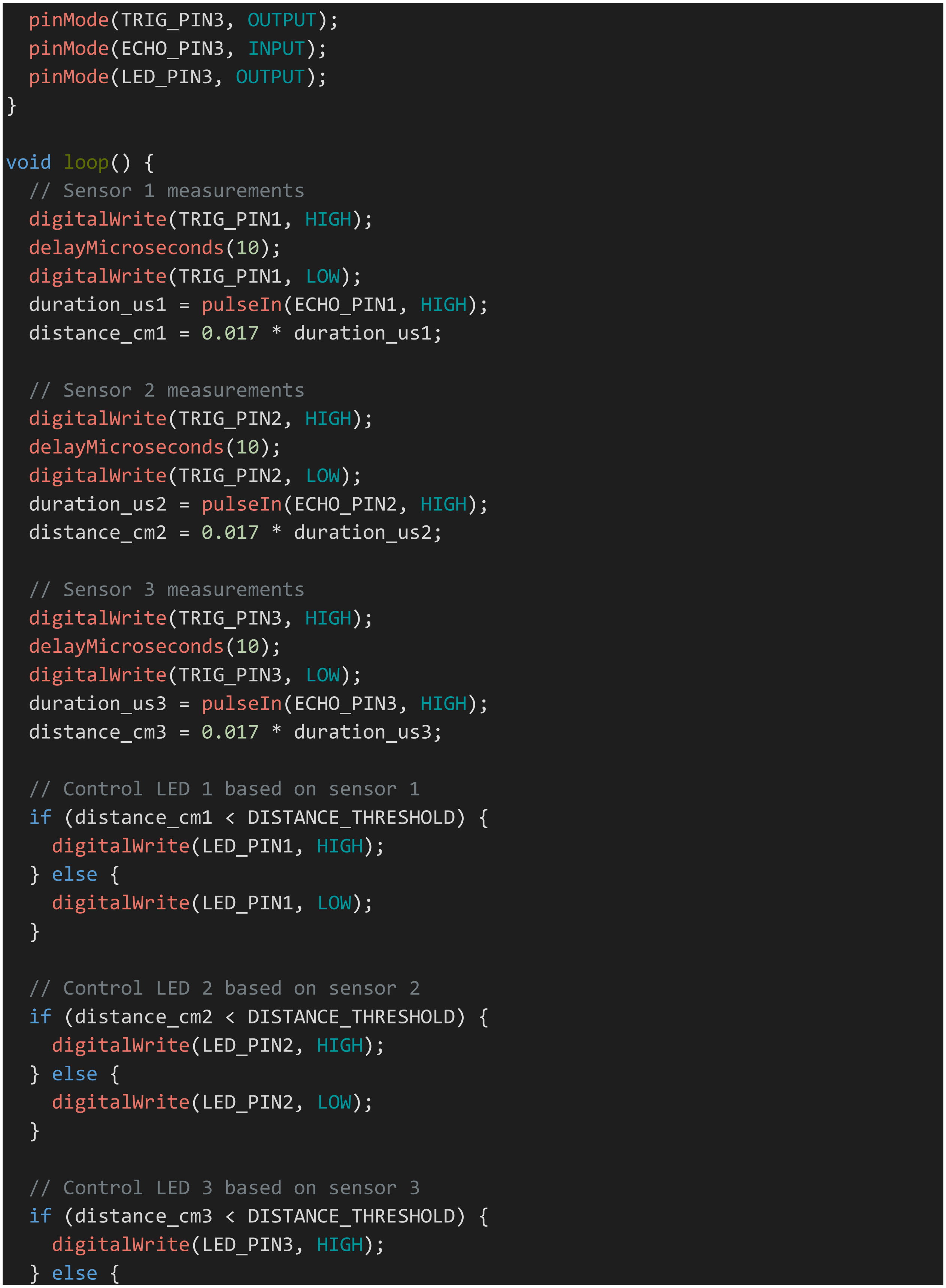
Connectivity:

Implement code for establishing a connection to the Wi-Fi network.

Set up communication protocols (e.g., MQTT, HTTP, or WebSocket) to send data to a central server or cloud platform

CODE:

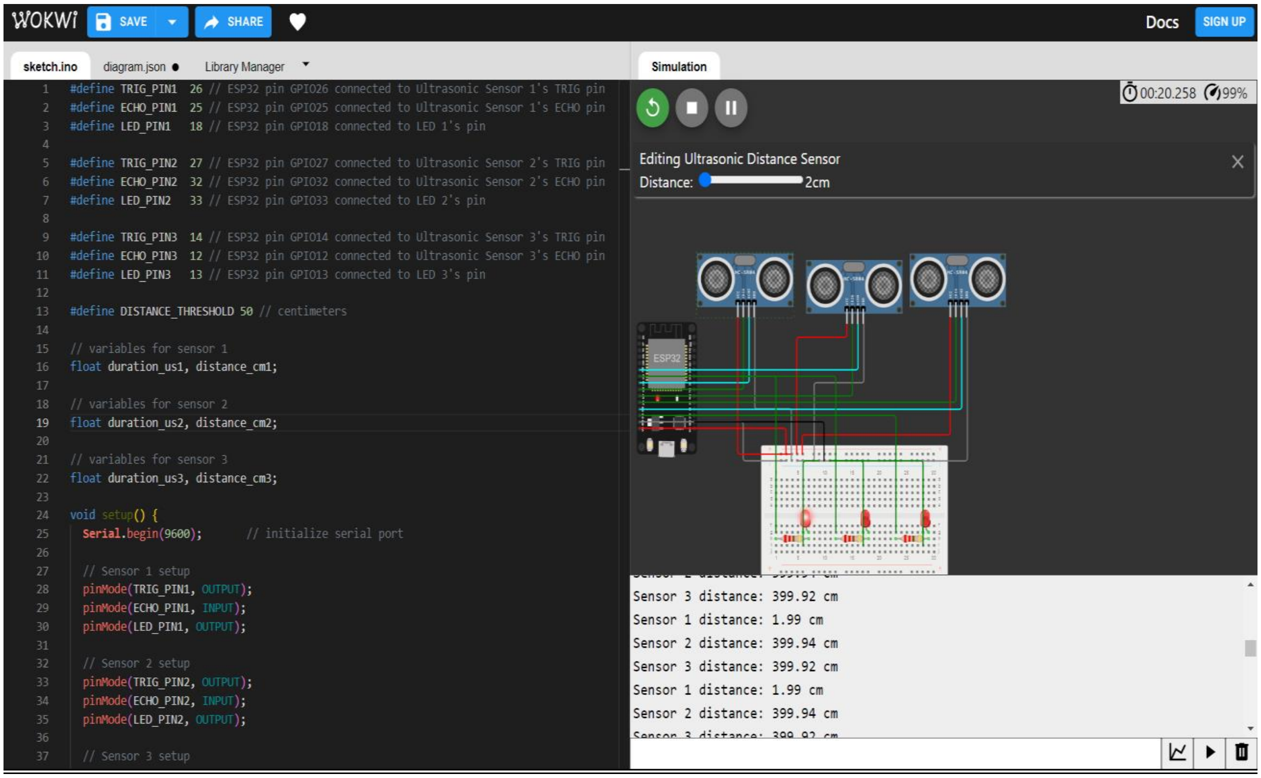
```
#define TRIG_PIN1  26 // ESP32 pin GPIO26 connected to Ultrasonic Sensor 1's TRIG pin
#define ECHO_PIN1  25 // ESP32 pin GPIO25 connected to Ultrasonic Sensor 1's ECHO pin
#define LED_PIN1   18 // ESP32 pin GPIO18 connected to LED 1's pin

#define TRIG_PIN2  27 // ESP32 pin GPIO27 connected to Ultrasonic Sensor 2's TRIG pin
#define ECHO_PIN2  32 // ESP32 pin GPIO32 connected to Ultrasonic Sensor 2's ECHO pin
#define LED_PIN2   33 // ESP32 pin GPIO33 connected to LED 2's pin

#define TRIG_PIN3  14 // ESP32 pin GPIO14 connected to Ultrasonic Sensor 3's TRIG pin
#define ECHO_PIN3  12 // ESP32 pin GPIO12 connected to Ultrasonic Sensor 3's ECHO pin
#define LED_PIN3   13 // ESP32 pin GPIO13 connected to LED 3's pin

#define DISTANCE_THRESHOLD 50 // centimeters

// variables for sensor 1
float duration_us1, distance_cm1;

// variables for sensor 2
float duration_us2, distance_cm2;

// variables for sensor 3
float duration_us3, distance_cm3;

void setup() {
  Serial.begin(9600);       // initialize serial port

  // Sensor 1 setup
  pinMode(TRIG_PIN1, OUTPUT);
  pinMode(ECHO_PIN1, INPUT);
  pinMode(LED_PIN1, OUTPUT);

  // Sensor 2 setup
  pinMode(TRIG_PIN2, OUTPUT);
  pinMode(ECHO_PIN2, INPUT);
  pinMode(LED_PIN2, OUTPUT);

  // Sensor 3 setup
```



```
  pinMode(TRIG_PIN3, OUTPUT);
  pinMode(ECHO_PIN3, INPUT);
  pinMode(LED_PIN3, OUTPUT);
}

void loop() {
  // Sensor 1 measurements
  digitalWrite(TRIG_PIN1, HIGH);
  delayMicroseconds(10);
  digitalWrite(TRIG_PIN1, LOW);
  duration_us1 = pulseIn(ECHO_PIN1, HIGH);
  distance_cm1 = 0.017 * duration_us1;

  // Sensor 2 measurements
  digitalWrite(TRIG_PIN2, HIGH);
  delayMicroseconds(10);
  digitalWrite(TRIG_PIN2, LOW);
  duration_us2 = pulseIn(ECHO_PIN2, HIGH);
  distance_cm2 = 0.017 * duration_us2;

  // Sensor 3 measurements
  digitalWrite(TRIG_PIN3, HIGH);
  delayMicroseconds(10);
  digitalWrite(TRIG_PIN3, LOW);
  duration_us3 = pulseIn(ECHO_PIN3, HIGH);
  distance_cm3 = 0.017 * duration_us3;

  // Control LED 1 based on sensor 1
  if (distance_cm1 < DISTANCE_THRESHOLD) {
    digitalWrite(LED_PIN1, HIGH);
  } else {
    digitalWrite(LED_PIN1, LOW);
  }

  // Control LED 2 based on sensor 2
  if (distance_cm2 < DISTANCE_THRESHOLD) {
    digitalWrite(LED_PIN2, HIGH);
  } else {
    digitalWrite(LED_PIN2, LOW);
  }

  // Control LED 3 based on sensor 3
  if (distance_cm3 < DISTANCE_THRESHOLD) {
    digitalWrite(LED_PIN3, HIGH);
  } else {
    digitalWrite(LED_PIN3, LOW);
  }

  // Print values to Serial Monitor
  Serial.print("Sensor 1 distance: ");
  Serial.print(distance_cm1);
  Serial.println(" cm");

  Serial.print("Sensor 2 distance: ");
  Serial.print(distance_cm2);
  Serial.println(" cm");

  Serial.print("Sensor 3 distance: ");
  Serial.print(distance_cm3);
  Serial.println(" cm");

  // Delay for a short time before repeating the measurement
  delay(100); // Adjust this delay as needed
}
```

OUTPUT:

https://wokwi.com/projects/378004204785628161



```
#define TRIG_PIN1  26 // ESP32 pin GPIO26 connected to Ultrasonic Sensor 1's TRIG pin
#define ECHO_PIN1  25 // ESP32 pin GPIO25 connected to Ultrasonic Sensor 1's ECHO pin
#define LED_PIN1   18 // ESP32 pin GPIO18 connected to LED 1's pin

#define TRIG_PIN2  27 // ESP32 pin GPIO27 connected to Ultrasonic Sensor 2's TRIG pin
#define ECHO_PIN2  32 // ESP32 pin GPIO32 connected to Ultrasonic Sensor 2's ECHO pin
#define LED_PIN2   33 // ESP32 pin GPIO33 connected to LED 2's pin

#define TRIG_PIN3  14 // ESP32 pin GPIO14 connected to Ultrasonic Sensor 3's TRIG pin
#define ECHO_PIN3  12 // ESP32 pin GPIO12 connected to Ultrasonic Sensor 3's ECHO pin
#define LED_PIN3   13 // ESP32 pin GPIO13 connected to LED 3's pin

#define DISTANCE_THRESHOLD 50 // centimeters

// variables for sensor 1
float duration_us1, distance_cm1;

// variables for sensor 2
float duration_us2, distance_cm2;

// variables for sensor 3
float duration_us3, distance_cm3;

void setup() {
  Serial.begin(9600);       // initialize serial port

  // Sensor 1 setup
  pinMode(TRIG_PIN1, OUTPUT);
  pinMode(ECHO_PIN1, INPUT);
  pinMode(LED_PIN1, OUTPUT);

  // Sensor 2 setup
  pinMode(TRIG_PIN2, OUTPUT);
  pinMode(ECHO_PIN2, INPUT);
  pinMode(LED_PIN2, OUTPUT);

  // Sensor 3 setup
```

```
Sensor 2 distance: 399.94 cm
Sensor 3 distance: 399.92 cm
Sensor 1 distance: 1.99 cm
Sensor 2 distance: 399.94 cm
Sensor 3 distance: 399.92 cm
Sensor 1 distance: 1.99 cm
Sensor 2 distance: 399.94 cm
Sensor 3 distance: 399.92 cm
```