SMART PARKING-DEVELOPMENT PART 1

Designing a smart parking system based on the ESP32 using the WiKoWi application can be a complex project, but I can provide you with a high-level overview of the steps involved. The WiKoWi application could be used to monitor and control the smart parking system through a mobile device. Here's a simplified roadmap for creating such a system:

Components Needed:

1. **ESP32 Microcontroller:** The ESP32 will serve as the brain of the smart parking system, connecting to sensors, Wi-Fi, and the WiKoWi application.

2. **Ultrasonic Sensors:** Use ultrasonic sensors to detect the presence of vehicles in parking spaces.

3. **Wi-Fi Module:** ESP32 has built-in Wi-Fi, which can be used to connect to the internet and communicate with the WiKoWi application.

4. **Relays or Servo Motors:** To control barriers or gates to open and close parking spaces.

5. **Power Supply:** Ensure you have a reliable power source for the ESP32 and sensors.

Steps to Create the Smart Parking System:

1. **Hardware Setup:**

   - Connect the ultrasonic sensors to the ESP32. Place these sensors in each parking space to detect the presence of a vehicle.

   - Connect relays or servo motors to control the barriers or gates of each parking space.

   - Power up the ESP32 and ensure all hardware components are functioning properly.

2. **Programming the ESP32:**

   - Write code for the ESP32 using the Arduino IDE or the ESP-IDF (Espressif IoT Development Framework).

   - Program the ESP32 to read data from the ultrasonic sensors and control the barriers based on the availability of parking spaces.

   - Implement a Wi-Fi interface to allow communication with the WiKoWi application. Use the ESP32's Wi-Fi capabilities to create a local hotspot or connect to an existing Wi-Fi network.

   - Set up MQTT or WebSocket communication for real-time updates between the ESP32 and the WiKoWi application.

Programming:

```
#define ECHO_PIN1 15 //Pins for Sensor 1
#define TRIG_PIN1 2 //Pins for Sensor 1

#define ECHO_PIN2 5    //Pins for Sensor 2
#define TRIG_PIN2 18   //Pins for Sensor 2

#define ECHO_PIN3 26  //Pins for Sensor 3
#define TRIG_PIN3 27   //Pins for Sensor 3



int LEDPIN1 = 13;
int LEDPIN2 = 12;
int LEDPIN3 = 14;

void setup() {
  Serial.begin(115200);
  pinMode(LEDPIN1, OUTPUT);
  pinMode(TRIG_PIN1, OUTPUT);
  pinMode(ECHO_PIN1, INPUT);

   pinMode(LEDPIN2, OUTPUT);
  pinMode(TRIG_PIN2, OUTPUT);
  pinMode(ECHO_PIN2, INPUT);

   pinMode(LEDPIN3, OUTPUT);
  pinMode(TRIG_PIN3, OUTPUT);
  pinMode(ECHO_PIN3, INPUT);
}

float readDistance1CM() {
```

```
  digitalWrite(TRIG_PIN1, LOW);
  delayMicroseconds(2);
  digitalWrite(TRIG_PIN1, HIGH);
  delayMicroseconds(10);
  digitalWrite(TRIG_PIN1, LOW);
  int duration = pulseIn(ECHO_PIN1, HIGH);
  return duration * 0.034 /2 ;
}

float readDistance2CM() {
  digitalWrite(TRIG_PIN2, LOW);
  delayMicroseconds(2);
  digitalWrite(TRIG_PIN2, HIGH);
  delayMicroseconds(10);
  digitalWrite(TRIG_PIN2, LOW);
  int duration = pulseIn(ECHO_PIN2, HIGH);
  return duration * 0.034 / 2;
}

float readDistance3CM() {
  digitalWrite(TRIG_PIN3, LOW);
  delayMicroseconds(2);
  digitalWrite(TRIG_PIN3, HIGH);
  delayMicroseconds(10);
  digitalWrite(TRIG_PIN3, LOW);
  int duration = pulseIn(ECHO_PIN3, HIGH);
  return duration * 0.034 / 2;
}

void loop() {
  float distance1 = readDistance1CM();
```

```
  float distance2 = readDistance2CM();
  float distance3 = readDistance3CM();

  bool isNearby1 = distance1 > 200;
  digitalWrite(LEDPIN1, isNearby1);



  bool isNearby2 = distance2 > 200;
  digitalWrite(LEDPIN2, isNearby2);



  bool isNearby3 = distance3 > 200;
  digitalWrite(LEDPIN3, isNearby3);

  Serial.print("Measured distance: ");
  Serial.println(readDistance1CM());
  Serial.println(readDistance2CM());
  Serial.println(readDistance3CM());
  delay(100);
}
```

3. **WiKoWi Application Integration:**

   - Develop a mobile application using the WiKoWi SDK that communicates with the ESP32.

   - The application should provide a user-friendly interface for users to find available parking spaces, reserve them, and pay for parking if necessary.

   - Use the WiKoWi SDK to send and receive data from the ESP32 in real-time.

   - Ensure the application provides alerts and notifications to users regarding parking space availability and reservation status.

4. **Data Storage and Analysis:**

   - Set up a cloud-based database to store data on parking space availability, reservations, and user information.

   - Implement data analysis and reporting features to gather insights on parking space utilization.

5. **Security and Access Control:**

- Implement security measures to protect the system from unauthorized access and data breaches.

- Use authentication and authorization mechanisms for users and administrators.

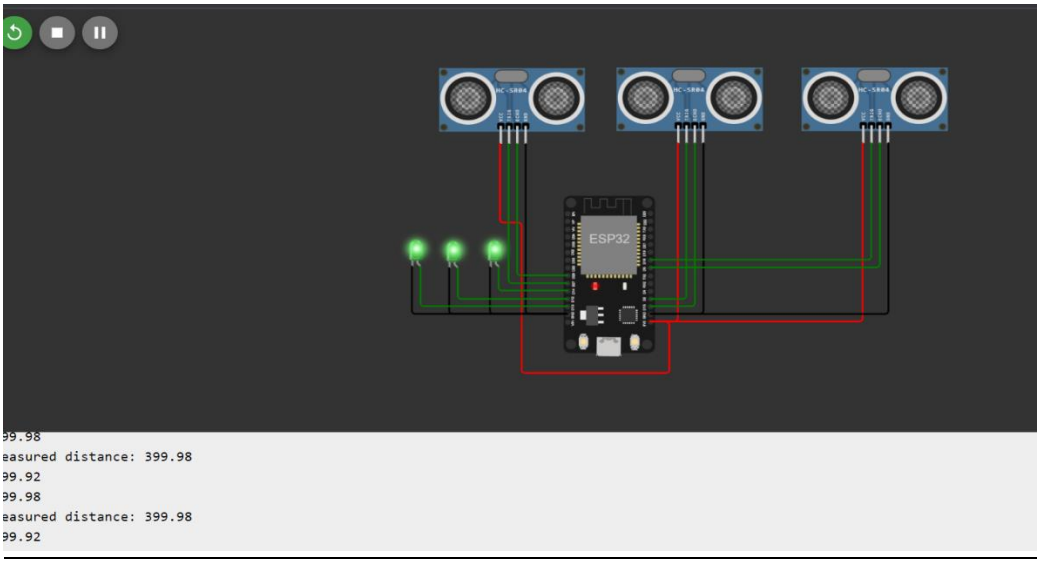
- Employ encryption for data transmission between the ESP32 and the mobile application.

6. **Testing and Deployment:**

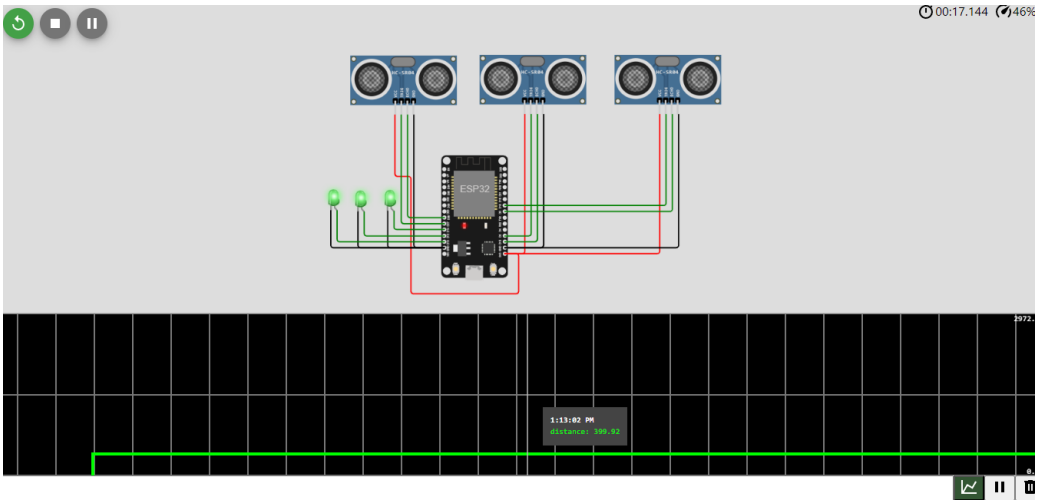
- Thoroughly test the smart parking system to ensure it functions as expected.

- Deploy the system in your parking area and make it accessible to users through the WiKoWi application.

<u>Simulate the project on Wokwi and observed output:</u>



a.output



b.simulation

7. **Maintenance and Updates:**

   - Regularly maintain and update the system to address bugs, security vulnerabilities, and user feedback.

Remember that this is a simplified overview, and the actual implementation may vary based on your specific requirements and the capabilities of the WiKoWi application. Additionally, you should consider scalability, robustness, and user experience as essential aspects of the design.