GOVERNMENT OF TAMILNADU

DIRECTORATE OF TECHNICAL EDUCATION, CHENNAI

NAAN MUDHALVAN SCHEME (TNSDC) SPONSORED

STUDENTS DEVELOPMENT PROGRAMME

ON

IoT AND ITS APPLICATIONS

HOST INSTITUTION

xxxxx

COIMBATORE – 04

TRAINING PARTNER

ENTHU TECHNOLOGY SOLUTIONS INDIA PVT LTD

DATE:

Name	Roll No

Table Of Contents

S no	Title	Page no
1	Abstract	
2	Introduction	
3	Hardware and Software Requirements	
4	Block Diagram	
5	Code	
6	Output Results	
7	Cloud Output	
8	Conclusion	

Abstract

The advent of the Internet of Things (IoT) has revolutionized the concept of home automation, enabling the seamless integration of smart devices to enhance the convenience, security, and energy efficiency of modern homes. This paper presents a comprehensive IoT-based home automation system that leverages wireless communication and cloud technology to control and monitor various household appliances. The system allows users to remotely manage devices such as lighting, heating, ventilation, air conditioning (HVAC), and security systems via a smartphone or web interface. Utilizing a combination of sensors, microcontrollers, and actuators, the proposed system offers real-time feedback and automation capabilities, significantly improving user experience. The implementation of this system demonstrates not only the feasibility and scalability of IoT in home automation but also highlights its potential to contribute to sustainable living by optimizing energy consumption and providing personalized environmental controls.

Introduction

The rapid advancement of technology and the proliferation of the Internet of Things (IoT) have profoundly transformed various aspects of modern life, with home automation emerging as a significant area of interest. IoT-based home automation systems enable the seamless interconnection of household devices, allowing for remote control, monitoring, and automation of everyday tasks. These systems not only enhance the convenience and comfort of living spaces but also contribute to improved energy efficiency and security.

Traditional home automation solutions often required extensive wiring, expensive hardware, and complex installations. In contrast, IoT-based systems leverage wireless communication technologies such as Wi-Fi, Zigbee, and Bluetooth, making them more accessible, scalable, and easier to deploy. These systems typically consist of smart devices like sensors, actuators, microcontrollers, and cloud-based platforms that work together to create a connected ecosystem within the home.

This project focuses on the design and implementation of an IoT-based home automation system that allows users to control and monitor various household appliances, such as lights, fans, heating, ventilation, and air conditioning (HVAC) systems, through a user-friendly interface. The system's architecture emphasizes interoperability, real-time feedback, and energy efficiency, providing a robust solution for modern smart homes. By integrating IoT technologies, this home automation system not only enhances the quality of life but also supports sustainable living practices by optimizing energy usage and providing personalized environmental controls.

Hardware and Software Requirements

Hardware Requirements

1.ESP32 Microcontroller

2.Relay Module

3.DC Motor

4.Flame Sensor

5.LED

6.BreadBoard

7.USB Cable

8.Jumper Wires

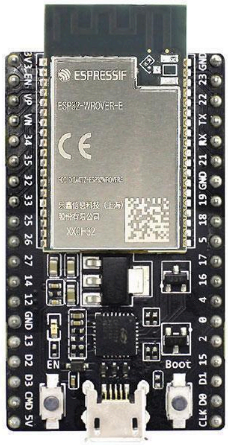
Software Requirements

1.Wokwi Simulator

2.Arduino IDE

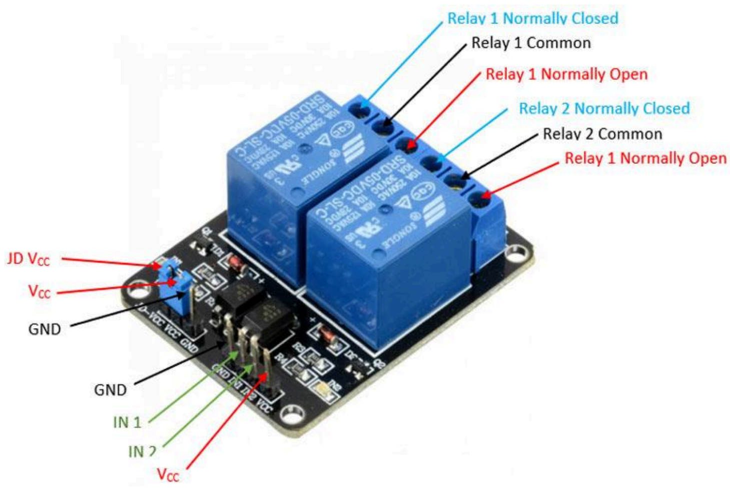
3.Thingzmate Cloud

ESP32 Microcontroller



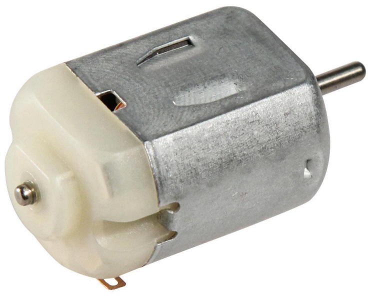
The ESP32 microcontroller is a powerful and versatile choice for IoT-based home automation projects, offering built-in Wi-Fi and Bluetooth capabilities for seamless wireless communication. Its dual-core processor, combined with a rich set of peripherals, allows for efficient control and monitoring of multiple household appliances. Additionally, the ESP32's low power consumption and robust security features make it an ideal solution for maintaining a reliable and secure smart home ecosystem.

Relay Module



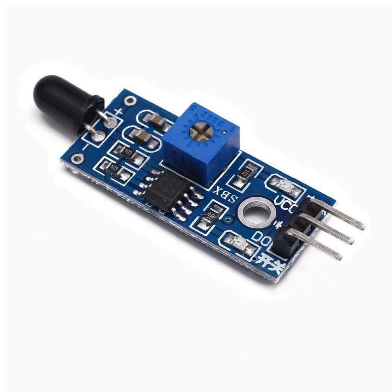
The relay module in this IoT-based home automation project is used to control high-voltage appliances, such as fans and heaters, by interfacing with the low-voltage ESP32 microcontroller. It acts as a switch that opens or closes electrical circuits based on control signals from the microcontroller. This allows for safe and reliable automation of home appliances, enabling their activation or deactivation remotely.

DC Motor



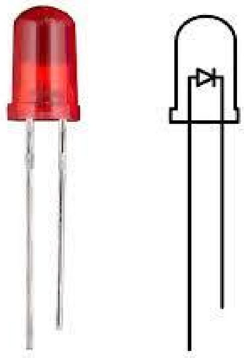
The DC motor in this IoT-based home automation project is used to drive mechanical components such as adjustable window blinds or small fans, adding dynamic control to the system. It is controlled via the ESP32 microcontroller, which regulates its speed and direction through pulse-width modulation (PWM) signals. This integration allows for precise automation and remote operation of mechanical elements within the home.

Flame Sensor



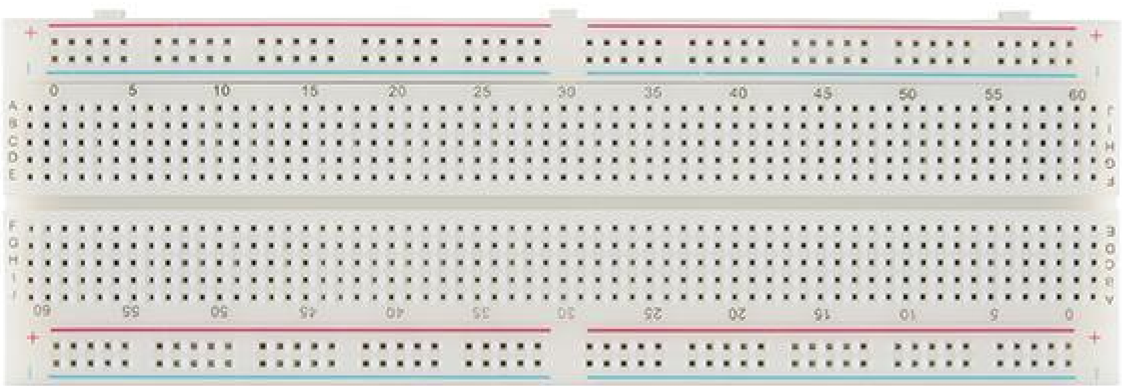
The flame sensor in this IoT-based home automation project is used to detect the presence of fire or flames, enhancing the safety features of the system. It provides real-time feedback to the ESP32 microcontroller, triggering alerts or automatic responses, such as activating a sprinkler system or turning off appliances, in the event of a fire. This integration helps in early fire detection and preventive safety measures within the home.

LED



In this IoT-based home automation project, LEDs serve as visual indicators representing the status of various household appliances, such as lights, fans, and HVAC systems. They provide immediate feedback to users by illuminating when a device is turned on and dimming when it's off. The LEDs are controlled via the ESP32 microcontroller, enabling real-time monitoring and control through the connected IoT platform.

BreadBoard



The breadboard in this IoT-based home automation project is used as a prototyping platform to easily assemble and test the connections between the ESP32 microcontroller, LEDs, and other components. It allows for quick adjustments and modifications without soldering, facilitating the design and troubleshooting process. This setup helps in efficiently demonstrating and validating the circuit before moving on to a more permanent solution.

USB-Cable

The USB cable is a critical tool in this project, used to connect the ESP32 microcontroller to a computer for power supply, programming, and debugging. It enables the transfer of code and data between the development environment and the microcontroller, facilitating the upload of firmware and real-time communication during the development process. The USB connection also allows for serial monitoring, providing valuable insights into the system's performance and behavior.

Jumper Wires

Jumper wires are essential in this project, used to connect the ESP32 microcontroller to the components on the breadboard. These wires provide a flexible and reliable way to link the microcontroller's GPIO pins to the LEDs, resistors, and other circuit elements, enabling proper signal and power flow. Their ease of use allows for quick modifications and testing during the prototyping stage.

Wokwi Simulator

The Wokwi simulator is utilized in this IoT-based home automation project to virtually prototype and test the circuit, including the ESP32 microcontroller, LEDs, and other components, without the need for physical hardware. It provides an interactive environment where code and connections can be simulated and debugged, ensuring that the system functions as expected before real-world implementation. This tool accelerates development by allowing iterative testing and refining of the design in a cost-effective manner.

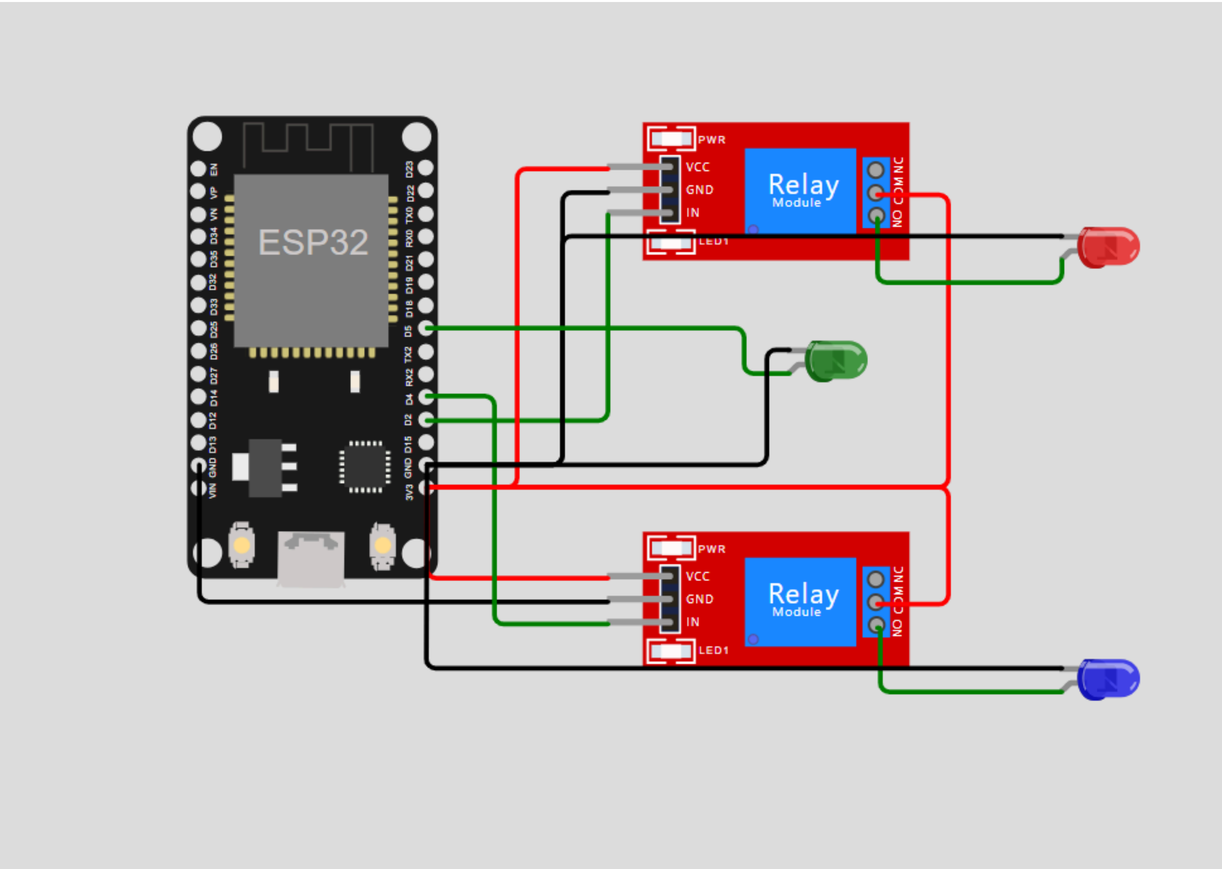
Arduino IDE

The Arduino IDE is employed in this IoT-based home automation project to write, compile, and upload code to the ESP32 microcontroller. Its user-friendly interface and extensive library support streamline the development process, making it easier to program and integrate various components. The IDE also facilitates debugging and real-time monitoring, enhancing the overall efficiency of the project development.

Thingzmate Cloud

ThingzMate Cloud is utilized in this IoT-based home automation project to provide a robust platform for remotely managing and controlling household appliances. It enables secure communication between the ESP32 microcontroller and user interfaces, facilitating real-time updates and control through web and mobile applications. The cloud service also offers data storage and analytics, enhancing the system's functionality and user experience.

Block Diagram



Code

```
#include <WiFi.h>
#include <HTTPClient.h>

#define WIFI_SSID "tyrant"
#define WIFI_PASSWORD "speed123"

const int ledPin = 5;   // GPIO pin for LED
const int relayPin1 = 4; // GPIO pin for Relay 1
const int relayPin2 = 2; // GPIO pin for Relay 2
const char *uplinkUrl =
"https://console.thingzmate.com/api/v1/device-types/esp3211/devices/esp3210/uplink";
const char *downlinkUrl =
"https://console.thingzmate.com/api/v1/device-types/esp3211/devices/esp3210/downlin
k";
String AuthorizationToken = "Bearer 70302768998061dab0bae1accbeabab0";

void setup() {
  Serial.begin(115200);  // Start the serial communication
  pinMode(ledPin, OUTPUT);
  pinMode(relayPin1, OUTPUT);
  pinMode(relayPin2, OUTPUT);
  digitalWrite(ledPin, LOW);
  digitalWrite(relayPin1, LOW);
  digitalWrite(relayPin2, LOW);

  delay(100);  // Short delay before Wi-Fi connection attempt

  // Connect to Wi-Fi
```

```
  Serial.println("Connecting to WiFi...");

  WiFi.begin(WIFI_SSID, WIFI_PASSWORD);

  int retries = 0;
  while (WiFi.status() != WL_CONNECTED && retries < 50) {  // Retry up to 50 times
    delay(500);  // Wait for 500ms
    Serial.print(".");
    retries++;
  }

  if (WiFi.status() == WL_CONNECTED) {
    Serial.println("\nConnected to WiFi");
    Serial.print("IP address: ");
    Serial.println(WiFi.localIP());
  } else {
    Serial.println("\nFailed to connect to WiFi. Restarting...");
    ESP.restart();  // Restart the ESP32 if Wi-Fi connection fails
  }
}

void loop() {
  if (WiFi.status() == WL_CONNECTED) {
    HTTPClient http;

    // Sending Uplink Data
    http.begin(uplinkUrl);
    http.addHeader("Content-Type", "application/json");
    http.addHeader("Authorization", AuthorizationToken);
```

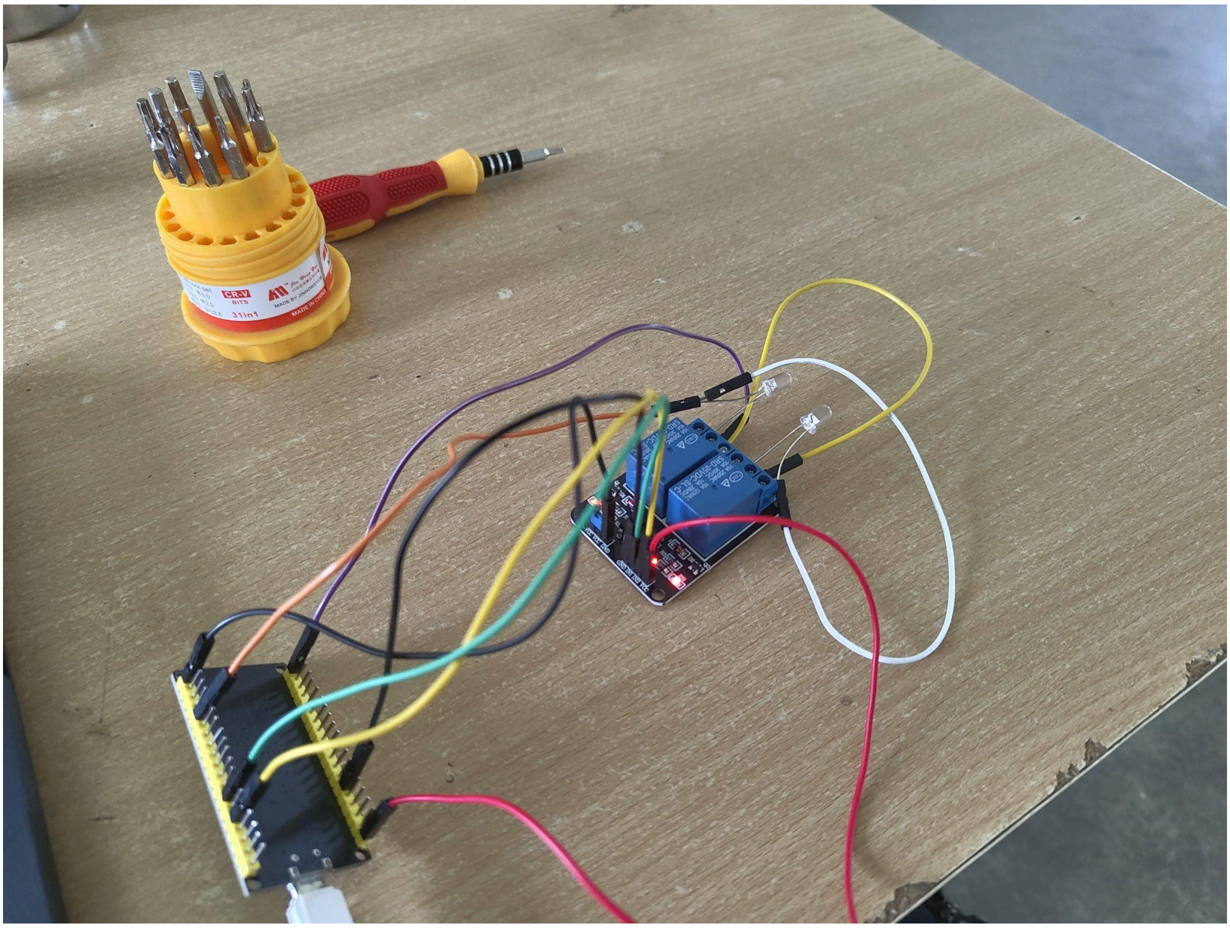
```
// Check if LED, Relay1, and Relay2 are ON or OFF
String ledStatus = digitalRead(ledPin) == HIGH ? "ON" : "OFF";
String relay1Status = digitalRead(relayPin1) == HIGH ? "ON" : "OFF";
String relay2Status = digitalRead(relayPin2) == HIGH ? "ON" : "OFF";

// Create JSON payload with LED, Relay1, and Relay2 statuses
String payload = "{\"led_status\":\"" + ledStatus + "\",\"relay1_status\":\"" + relay1Status + "\",\"relay2_status\":\"" + relay2Status + "\"}";

// Send POST request
int httpResponseCode = http.POST(payload);

if (httpResponseCode > 0) {
  String response = http.getString();
  Serial.println("Uplink HTTP Response code: " + String(httpResponseCode));
  Serial.println(response);
} else {
  Serial.print("Uplink Error code: ");
  Serial.println(httpResponseCode);
}

http.end(); // Free resources

// Receiving Downlink Data
http.begin(downlinkUrl);
http.addHeader("Authorization", AuthorizationToken);

// Send GET request
httpResponseCode = http.GET();
```

```
    if (httpResponseCode > 0) {
      String response = http.getString();
      Serial.println("Downlink HTTP Response code: " + String(httpResponseCode));
      Serial.println(response);

      // Parse the response and control LED/Relay based on separate "ON" and "OFF" commands
      if (response.indexOf("LON") >= 0) {
        digitalWrite(ledPin, HIGH);
        Serial.println("LED turned ON");
      } else if (response.indexOf("LOFF") >= 0) {
        digitalWrite(ledPin, LOW);
        Serial.println("LED turned OFF");
      }

      if (response.indexOf("R1ON") >= 0) {
        digitalWrite(relayPin1, HIGH);
        Serial.println("Relay 1 turned ON");
      } else if (response.indexOf("R1OFF") >= 0) {
        digitalWrite(relayPin1, LOW);
        Serial.println("Relay 1 turned OFF");
      }

      if (response.indexOf("R2ON") >= 0) {
        digitalWrite(relayPin2, HIGH);
        Serial.println("Relay 2 turned ON");
      } else if (response.indexOf("R2OFF") >= 0) {
        digitalWrite(relayPin2, LOW);
        Serial.println("Relay 2 turned OFF");
```

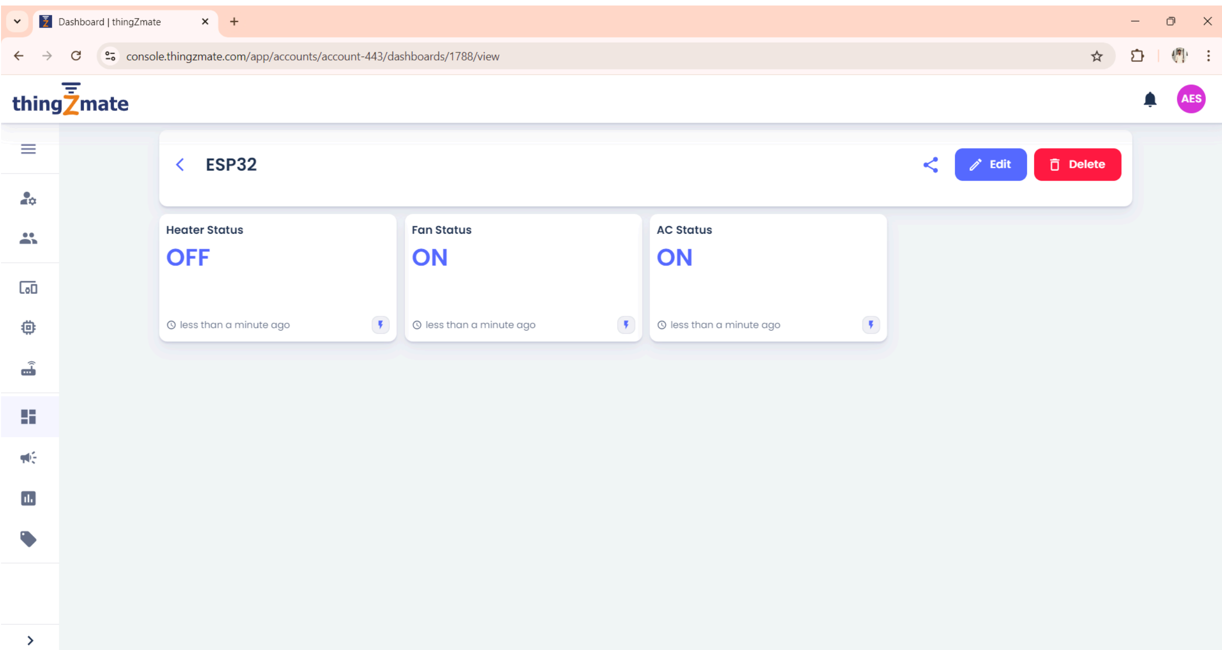
```
    }

  } else {
    Serial.print("Downlink Error code: ");
    Serial.println(httpResponseCode);
  }

  http.end(); // Free resources

 } else {
  Serial.println("WiFi not connected. Attempting to reconnect...");

  // Attempt to reconnect to Wi-Fi
  WiFi.begin(WIFI_SSID, WIFI_PASSWORD);

  int retries = 0;
   while (WiFi.status() != WL_CONNECTED && retries < 50) {  // Retry up to 50
times
    delay(500);  // Wait for 500ms
    Serial.print(".");
    retries++;
  }

  if (WiFi.status() == WL_CONNECTED) {
    Serial.println("\nReconnected to WiFi");
  } else {
    Serial.println("\nFailed to reconnect to WiFi. Restarting...");
    ESP.restart();  // Restart the ESP32 if Wi-Fi connection fails
  }
```

```
  }

  delay(5000); // Wait for 5 seconds before sending the next request
}
```

Output Results



Cloud Output



Conclusion

The implementation of an IoT-based home automation system utilizing components like the ESP32 microcontroller, LEDs, relay modules, and various sensors has demonstrated significant advancements in modern home management. By integrating these technologies, the project offers enhanced control, monitoring, and automation of household appliances, contributing to greater convenience, energy efficiency, and safety. The use of platforms like ThingzMate Cloud for remote management and simulation tools such as Wokwi further streamline development and ensure robust functionality. Overall, this smart home solution not only improves user experience but also paves the way for more innovative and sustainable living environments.