**Lab Exercise 16- ESP32 Http Wi-Fi to ON/ OFF LED**

In this lab exercise, you'll learn how to control an LED via a web browser using the **ESP32** and **Wokwi** (an online simulator). We will use **HTTP** to send requests to turn the LED on and off. Wokwi allows you to simulate ESP32 projects directly in the browser without needing physical hardware.

**Objective**

To control an LED connected to the ESP32 through a web interface using **Wi-Fi** and **HTTP** in the **Wokwi** simulator.

**Materials Required**

* ESP32 Development Board (in Wokwi)
* LED (in Wokwi)
* Wokwi Account (<https://wokwi.com/>)
* Arduino IDE or Wokwi's built-in editor

**Steps to Simulate the ESP32 in Wokwi**

**1. Set Up the Circuit in Wokwi**

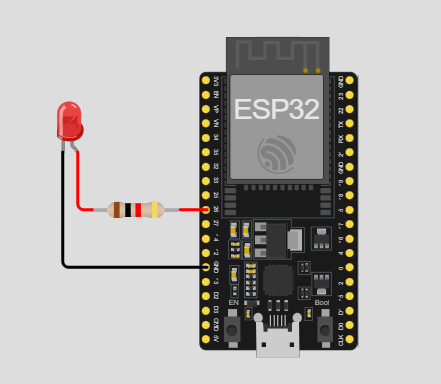
1. Visit [**https://wokwi.com/**](https://wokwi.com/) and create a new project.
2. Add an **ESP32** and an **LED** in the simulation environment. You can do this by clicking on "Add a part" and selecting **ESP32** and **LED**.
   * **Connect the LED**:
     + Connect the **anode** of the LED to **GPIO 2** of the ESP32.
     + Connect the **cathode** of the LED to the **GND** pin of the ESP32.
   * No need for a resistor in Wokwi simulations, but typically you would use a **220Ω** resistor in real-life applications.

**Wiring Diagram**

In Wokwi, it will look something like this:

ESP32 GPIO 2 -> Anode of LED

ESP32 GND -> Cathode of LED



**2. Write the HTTP Web Server Code**

Once you have the circuit ready, you can write and upload the code in the **Wokwi simulator**.

#include <WiFi.h>

#include <WebServer.h> // Use WebServer library for simple HTTP server

// Replace these with your Wi-Fi credentials

const char\* ssid = "Wokwi-GUEST"; // Wi-Fi name (Wokwi default network)

const char\* password = ""; // Password is blank for Wokwi network

// Set GPIO pin for LED

int ledPin = 2;

// Create a web server object that listens for HTTP requests on port 80

WebServer server(80);

void handleRoot() {

// HTML content for the web page

String html = "<html><body><h1>ESP32 LED Control</h1>";

html += "<p><a href=\"/LEDON\"><button>Turn LED ON</button></a></p>";

html += "<p><a href=\"/LEDOFF\"><button>Turn LED OFF</button></a></p>";

html += "</body></html>";

// Send the HTTP response

server.send(200, "text/html", html);

}

void handleLEDON() {

digitalWrite(ledPin, HIGH); // Turn LED ON

server.send(200, "text/html", "<html><body><h1>LED is ON</h1><p><a href=\"/\"><button>Back</button></a></p></body></html>");

}

void handleLEDOFF() {

digitalWrite(ledPin, LOW); // Turn LED OFF

server.send(200, "text/html", "<html><body><h1>LED is OFF</h1><p><a href=\"/\"><button>Back</button></a></p></body></html>");

}

void setup() {

Serial.begin(115200);

// Initialize LED pin as output

pinMode(ledPin, OUTPUT);

digitalWrite(ledPin, LOW); // Turn off the LED initially

// Connect to Wi-Fi

Serial.print("Connecting to Wi-Fi");

WiFi.begin(ssid, password);

while (WiFi.status() != WL\_CONNECTED) {

delay(500);

Serial.print(".");

}

Serial.println("");

Serial.println("Wi-Fi connected.");

Serial.print("ESP32 IP address: ");

Serial.println(WiFi.localIP());

// Define HTTP routes

server.on("/", handleRoot); // Define what happens when you access the root "/"

server.on("/LEDON", handleLEDON); // Route to turn the LED on

server.on("/LEDOFF", handleLEDOFF); // Route to turn the LED off

// Start the web server

server.begin();

Serial.println("HTTP server started.");

}

void loop() {

server.handleClient(); // Handle incoming client requests

}

**3. Simulate and Test in Wokwi**

1. **Run the simulation** by clicking the green "Play" button in Wokwi.
2. In the **Serial Monitor** (in Wokwi), you’ll see the **IP address** of the ESP32 once it's connected to Wi-Fi (usually something like 192.168.x.x).
3. Open a browser (in your computer, not in Wokwi) and enter the IP address shown in the Serial Monitor.
   * For example, if the IP is 192.168.4.1, you would type:  
     http://192.168.4.1 in your browser.
4. You should see a simple webpage with two buttons: **Turn LED ON** and **Turn LED OFF**. Clicking these buttons will control the LED in the Wokwi simulation.

**4. Troubleshooting**

* Make sure the ESP32 is connected to the default Wokwi network Wokwi-GUEST.
* If the LED does not turn on or off, verify that the **GPIO 2** is correctly wired to the LED in the simulation.

**Explanation of the Code**

1. **Wi-Fi Setup**:
   * The ESP32 connects to the Wokwi-GUEST network, which is pre-configured in the Wokwi simulator. There's no password, so the password is left as an empty string.
2. **WebServer**:
   * The **WebServer** library is used to create a simple web server that listens on port 80. It hosts an HTML webpage with buttons to control the LED.
3. **Handling HTTP Requests**:
   * **handleRoot()**: This function sends the HTML content for the main page.
   * **handleLEDON()** and **handleLEDOFF()**: These functions are triggered when the user clicks the "Turn LED ON" or "Turn LED OFF" buttons. They change the LED's state by setting the **GPIO 2** pin high or low and return a confirmation webpage.
4. **server.begin()**: Starts the web server.
5. **server.handleClient()**: Continuously listens for client requests and responds accordingly.

**Conclusion**

In this lab exercise, you’ve simulated an **ESP32 HTTP server** in **Wokwi** that allows you to control an LED using a web browser. This is an excellent foundation for creating IoT projects, where you can control multiple devices or read sensor data via a web interface.