|  |
| --- |
|  |

**Internet of Things Lab**

**CSE 402**

# Project Report on

# ESP8266-Based Web Server for GPIO Control

**Submitted To:**

**Tahmina Islam**

Lecturer, Department of CSE

University of Information Technology & Sciences

**Ayanava Paul**

Lecturer, Department of CSE

University of Information Technology & Sciences

**Submitted By:**

1. Israt Jahan Eva (7B1\_57)

2. Irin Sultana Poly (7B2\_91)

3. Sadia Akter (7B1\_59\_)

4. Asadur Rahman Asif (7B2\_79)

**Date of Submission:** 17/12/24

## Introduction

The ESP8266 is a popular Wi-Fi module used for IoT applications due to its low cost, ease of use, and ability to connect to the internet. In this project, we implement a simple web server using the ESP8266 to control two GPIO pins remotely. By accessing a web interface, users can turn GPIO pins ON or OFF, controlling connected devices such as LEDs or relays.

## Table of Contents

1. **Introduction**
2. **Project Objective**
3. **Hardware Components**
4. **Software and Libraries**
5. **Circuit Design**
6. **Code Explanation**
7. **Web Server Interface**
8. **Results and Testing**
9. **Conclusion**

## Project Objective

The primary goal of this project is to:

* Set up an ESP8266-based web server.
* Control GPIO pins 0 and 2 through a web interface.
* Allow users to switch the GPIO pins ON and OFF remotely using HTTP requests.

## Hardware Components

1. **ESP8266 Module (NodeMCU)** – The main controller with Wi-Fi capabilities.
2. **LEDs** – Used to visualize GPIO control.
3. **220-ohm Resistors** – For limiting current through the LEDs.
4. **Breadboard** – For easy prototyping.
5. **Jumper Wires** – For connections between components.

## Software and Libraries

* **Arduino IDE** – Used to write and upload the code to the ESP8266.
* **ESP8266WiFi Library** – For managing Wi-Fi connectivity.

## Circuit Design

### Pin Connections

* **GPIO 0 (D3)**: Connected to LED 1 through a 220-ohm resistor.
* **GPIO 2 (D4)**: Connected to LED 2 through a 220-ohm resistor.

## Code Explanation

### Wi-Fi Setup

const char\* ssid = "REPLACE\_WITH\_YOUR\_SSID";

const char\* password = "REPLACE\_WITH\_YOUR\_PASSWORD";

WiFiServer server(80);

* **ssid** and **password** hold your Wi-Fi credentials.
* **server** initializes a web server on port 80.

### GPIO Initialization

const int output5 = 0; // GPIO 0

const int output4 = 2; // GPIO 2

void setup() {

pinMode(output5, OUTPUT);

pinMode(output4, OUTPUT);

digitalWrite(output5, LOW);

digitalWrite(output4, LOW);

* Sets **GPIO 0** and **GPIO 2** as outputs and initializes them to LOW.

### 6.3. Web Server Loop

void loop() {

WiFiClient client = server.available();

if (client) {

String currentLine = "";

while (client.connected()) {

if (client.available()) {

char c = client.read();

header += c;

if (c == '\n' && currentLine.length() == 0) {

// HTTP Response

client.println("HTTP/1.1 200 OK");

client.println("Content-type:text/html");

client.println("Connection: close");

client.println();

* Listens for client connections and processes incoming HTTP requests.
* Sends an HTTP response with the appropriate HTML content.

### 6.4. GPIO Control via HTTP Requests

if (header.indexOf("GET /5/on") >= 0) {

digitalWrite(output5, HIGH);

} else if (header.indexOf("GET /5/off") >= 0) {

digitalWrite(output5, LOW);

}

* Checks the HTTP request and sets GPIO 0 or 2 to HIGH or LOW based on the request.

## Web Server Interface

### HTML Interface

<!DOCTYPE html>

<html>

<head>

<meta name="viewport" content="width=device-width, initial-scale=1">

<style>

html { font-family: Helvetica; text-align: center; }

.button { background-color: #195B6A; color: white; padding: 16px 40px; cursor: pointer; }

.button2 { background-color: #77878A; }

</style>

</head>

<body>

<h1>ESP8266 Web Server</h1>

<p>GPIO 5 - State OFF</p>

<p><a href="/5/on"><button class="button">ON</button></a></p>

<p><a href="/5/off"><button class="button button2">OFF</button></a></p>

</body>

</html>

* Provides buttons to toggle GPIO pins ON and OFF.
* Displays the current state of each GPIO pin.

## Results and Testing

### Testing Procedure

1. **Upload the code** to the ESP8266 using the Arduino IDE.
2. **Connect to the ESP8266's IP address** displayed on the Serial Monitor.
3. Use the web interface to turn GPIO 0 and GPIO 2 ON or OFF.
4. **Observe LEDs** connected to GPIO pins responding to button clicks.

### Observations

* Successful control of GPIO pins via the web interface.
* Real-time feedback on GPIO states displayed on the web page.
* Stable Wi-Fi connection and responsive web server.

## Conclusion

This project demonstrates how to use the ESP8266 module to create a simple IoT-based web server for controlling GPIO pins remotely. The web interface allows users to toggle GPIO pins easily, enabling applications like remote lighting control, automation, and basic IoT systems.

## Future Improvements

1. **Add Password Protection**: Secure the web server to prevent unauthorized access.
2. **Use AJAX for Real-Time Updates**: Improve the interface to provide real-time status updates without refreshing the page.
3. **Add More GPIO Pins**: Extend the project to control more devices.
4. **Implement MQTT Protocol**: Enable communication with other IoT platforms or cloud services.
5. **Mobile App Interface**: Develop a mobile application for easier control.

**The End**