📡 **A Micro Python-based IoT Project**

Prepared by: Maryam Munawar(1344), Fatima Ahmad(1341)
Course: IoT Fundamentals
Instructor Name: Sir Nasir Mahmood
Submission-Date: 3-12-202

Title: Designing a Webserver for AP and Station-Based ESP32

1. Introduction

This document details the development of a webserver using an ESP32 that operates in both **Access Point (AP) and Station (STA) modes**. The webserver allows users to input **RGB values** through a web interface to control an RGB LED. Additionally, the ESP32 reads humidity and temperature values from a **DHT11 sensor** and displays them on the same web page. The project also includes an **OLED display** for showing messages and an additional web page for other tasks.

2. Project Requirements

2.1 Hardware Components

- ESP32 development board
- DHT11 temperature & humidity sensor
- OLED display (SSD1306)
- RGB LED (NeoPixel)
- Power supply (USB or battery)

2.2 Software Components

- MicroPython (Thonny IDE)
- HTML, CSS, JavaScript for the web interface

- MicroPython libraries: network, socket, dht, machine, ssd1306

3. System Implementation

3.1 WiFi Configuration

- The ESP32 connects to a WiFi network in **Station Mode** using credentials.
- The webserver is accessible through the assigned IP address.
- In **AP Mode**, the ESP32 creates a local network.

3.2 Webserver Setup

- The ESP32 runs a **webserver using sockets**.
- A web page is hosted, allowing users to **input RGB values**.
- The ESP32 reads these values and updates the **NeoPixel RGB LED**.

3.3 Sensor Integration (DHT11)

- The ESP32 reads **temperature and humidity** from the DHT11 sensor.
- These values are displayed on the **web page**.

3.4 OLED Display

- The OLED screen is interfaced via **I2C communication**.
- It displays sensor readings and **user messages** from the web interface.

3.5 Additional Web Page

- A secondary web page performs additional ESP32 tasks (defined by the developer).

4. Code Structure

4.1 main.py (Webserver & Sensor Handling)

- Initializes WiFi
- Reads **DHT11 sensor** data

- Controls **RGB LED**
- Serves the **web page**

4.2 wifi.py (WiFi Handling)

- Manages ESP32 WiFi connections
- Handles AP & Station mode switching

4.3 display.py (OLED Display Control)

- Controls text display on the OLED screen

4.4 blynk-test.py (Optional IoT Integration)

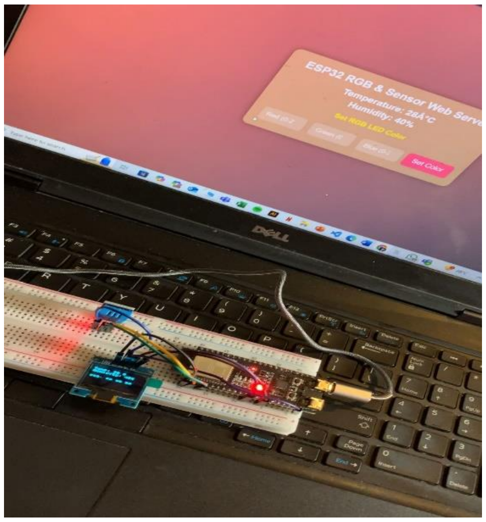
- Provides an interface for **Blynk IoT services** (if used)

5. Web Page Design

- The interface includes **RGB input fields**, a **temperature display**, and an OLED message field.
- Styled using **CSS** for an attractive layout.

6. Project Image

The following image illustrates the setup of the ESP32-based webserver, including the hardware components and web interface.



7. Conclusion

This project successfully integrates **WiFi, sensor data, an OLED display, and RGB LED control** into a webserver running on an ESP32. The design allows for both AP and Station mode operations, making it flexible for IoT applications.