**National Textile University, Faisalabad**



**Department of Computer Science**

**IOT Group Assignment-1**

**Designing a Webserver**

**Group Members:**

**Jahanzaib Arshad 22-NTU-CS-1363**

**‏‏‏‏‏M. Talha Javed 22-NTU-CS-1366**

**Safder Rehman 22-NTU-CS-1371**

**Task 1**

**Report on ESP32-S3 Web-Based Monitoring and Control System**

**Objective**

The primary goal of this project is to create an interactive web-based system using the ESP32-S3 microcontroller. The system allows users to monitor environmental parameters like temperature and humidity, control an RGB LED, and display text on an OLED screen via a web interface. This is achieved through a combination of embedded programming, networking, and web technologies.

**System Overview**

The ESP32-S3 is configured to work in both Station Mode (STA) and Access Point Mode (AP). It establishes a web server that provides a user-friendly interface for real-time monitoring and control. The core functionalities include:

* Reading and displaying temperature and humidity values
* Updating text on an OLED display via web input
* Controlling RGB LED colors via a web interface
* Using socket programming for communication between the web server and client

**Module Breakdown**

**1. Temperature and Humidity Data Transmission**

* The DHT11 sensor is connected to GPIO4 of the ESP32-S3.
* The ESP32-S3 reads temperature and humidity values using the dht module.
* The web server provides endpoints (/temperature and /humidity) which return the sensor values as text.
* The JavaScript functions on the web page use fetch() to retrieve these values every second and update them dynamically in the UI.

**2. Text Transfer from Web Page to OLED Display**

* The OLED display is interfaced using the SoftI2C module with SDA on GPIO8 and SCL on GPIO9.
* Users can enter a text string on the web page, which is sent as a GET request (/?TEXT&text=your\_message).
* The ESP32-S3 extracts the text from the request and sends it to the OLED display for rendering.
* The OledDisplay() function ensures proper formatting and updates the screen.

**3. RGB LED Control via Web Interface**

* A NeoPixel RGB LED is connected to GPIO48.
* The web page allows users to set Red, Green, and Blue values (0-255) via input fields.
* When submitted, the values are sent as a GET request (/?RGB&r=value&g=value&b=value).
* The ESP32-S3 extracts the RGB values and updates the NeoPixel LED using UpdateNeoPixel().

**4. Socket Programming for Web Communication**

* A TCP server is created using the socket module, bound to port 80.
* The ESP32-S3 listens for incoming HTTP GET requests.
* Based on the request type:
* It responds with temperature/humidity data.
* It updates the OLED display.
* It modifies the RGB LED colors.
* The connection is then closed to free up resources for new clients.

**Conclusion**

This project successfully demonstrates real-time data acquisition and control using an ESP32-S3 microcontroller with a web-based interface. By integrating DHT11, an OLED display, and a NeoPixel LED, the system provides an interactive experience where users can monitor and control hardware remotely. The use of socket programming ensures efficient communication between the ESP32-S3 and the web client, making the system both robust and scalable.

**Task 2**

**Report on ESP32-S3-Based Calculator with OLED Display and Web Interface**

**Purpose of the Task**

The objective of this project is to create a calculator using the ESP32-S3 microcontroller. The system integrates WiFi connectivity, an OLED display, and a web-based interface to perform arithmetic operations. Users can interact with the calculator via a webpage hosted on the ESP32-S3, while results are displayed on an OLED screen. The project demonstrates the use of embedded networking, web development, and display interfacing within a microcontroller environment.

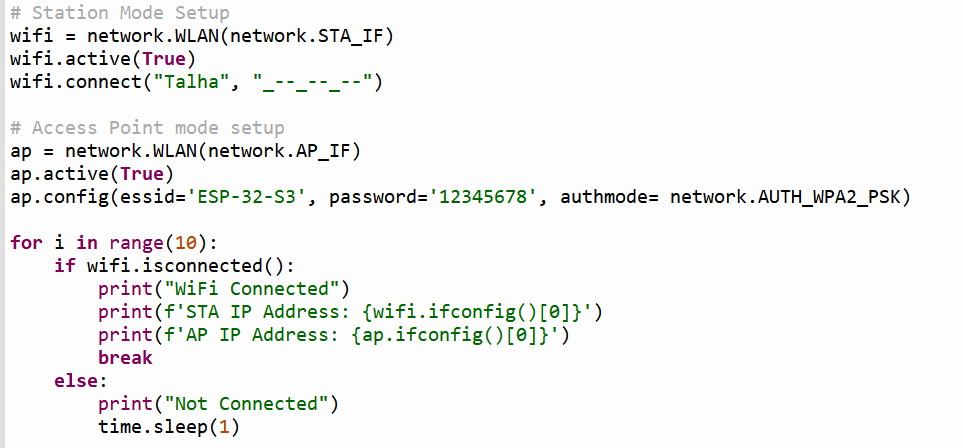
**Modules and Their Functionality**

**1. Network Module (WiFi and Access Point Setup)**

The ESP32-S3 is configured to operate in two modes:

* Station Mode: Connects to an existing WiFi network using the network.WLAN(network.STA\_IF) class.
* Access Point Mode: Creates its own WiFi network (SSID: ESP-32-S3) for users to connect directly if no external network is available.

**Key Code Snippets:**

The system attempts to connect to WiFi for up to 10 seconds, displaying connection status.

**2. Web Server Module**

The ESP32-S3 runs a simple HTTP web server using Python’s socket module. The server listens on port 80 and serves a calculator webpage to connected users. It processes GET requests to perform calculations.

**Key Code Snippets:**

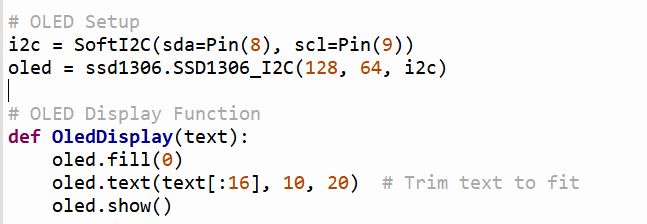
A black text on a white background

AI-generated content may be incorrect.When the server receives a request containing ?CALC, it extracts parameters, performs arithmetic operations, and returns the result.

**3. OLED Display Module**

An SSD1306 OLED display is connected to the ESP32-S3 via I2C (pins SDA = 8, SCL = 9). The display is used to show the current operation and result in real time.

**Key Code Snippets:**

This function updates the OLED display with the arithmetic operation performed.

**4. Web Interface Module**

The web interface is built using HTML, CSS, and JavaScript. The UI includes:

* A numeric keypad
* Arithmetic operators (+, -, \*, /)
* A clear button
* A result display section

**Key Features:**

* Fetch requests are sent to the ESP32 server with user inputs.
* The server processes requests and returns results.
* JavaScript updates the on-screen display.

**Key Code Snippets:**

A computer screen shot of a program

AI-generated content may be incorrect.This function sends user input to the ESP32 server, receives the calculated result, and updates the display.

**Conclusion**

This project demonstrates the integration of networking, web-based user interfaces, and embedded displays in an ESP32-S3 microcontroller. The system successfully enables users to perform calculations via a web browser and displays the results on an OLED screen, making it an effective example of IoT-based interactive computing.