National Textile University, Faisalabad

Department of Computer Science

Name:	Zainab sultan
Class:	BSCS_B 5th Semester
Registration No:	23-NTU-CS-1097
Lab Plan:	01
Course Name:	IOT and Embedded system
Submitted To:	**Nasir Masood**
Submission Date:	14/12/2025

Homework-01 – After mid

Question-1 ESP32 Webserver (webserver.cpp)

Part A: Short Questions

1. **What is the purpose of Webserver server (80); and what does port 80 represent?**

   Web Server (80) is used to create a web server on the ESP32 that listens for incoming client request. Port 80 is the default port for HTTP web traffic and it is used for listening the incoming HTTP requests. This pot is used for web communication.

2. **Explain the role of server. On ("/", handle Root); in this program.**

Server. On ("/", handle Root) tells what action ESP32 performs when a client accesses the root URL (/) of web server. This line tells the web server to call handle Root () function. It defines what content should be sent to the clients when they visit the main page.

3. **Why is server handle Client (); placed inside the loop () function? What will happen if it is removed?**

server handle Client () is placed inside the loop so that it can continuously check and handle client requests. If it is removed from the loop() then it will stop checking and handling the client requests and then web page will not load and process and response because client request is not proceeded.

4. **In handle Root (), explain the statement:**
server. send (200, "text/html", html).

This statement is a response send to the client Browser from ESP32. 200 is the HTTP status code for “OK” or a successful HTTP response. “text/html” tells that the content is a web page or it specifies the content type as HTML. Html contains actual HTML code to display in the browser.

5. **What is the difference between displaying last measured sensor values and taking a fresh DHT reading inside handle Root ()?**

Displaying last measured sensor value shows the data or the values of temperature and humidity, which are recently stored in the main memory. These values can be outdated but this process is faster.

Taking a fresh DHT reading inside handle Root () means the current reading, read the sensor in real-time. This may be up to date however this will take a time.

Part B: Long Question

Describe the complete working of the ESP32 webserver-based temperature and humidity monitoring system.

Your answer should include:
- ESP32 Wi-Fi connection process and IP address assignment
- Web server initialization and request handling
- Button-based sensor reading and OLED update mechanism.
- Dynamic HTML webpage generation
- Purpose of meta refresh in the webpage
- Common issues in ESP32 webserver projects and their solutions

1. ESP32 Wi-Fi Connection and IP Address Assignment:

ESP32 connected with WIFI network using SSID and password. After successful connection, the router assign IP address to ESP32. This IP address allow user to access web server of ESP32 from any device that is connected to the same network using web browser.

2. Web server initialization and request handling:

After connecting to WIFI, the ESP32 start web server on port 80 for web communication. The server continuously listen to the incoming HTTP requests. When user use URL, a root handle function is call that prepare and send the web pages containing sensor data.

3. Button-based sensor reading and OLED update mechanism:

A push button is used fro sensor updates. When a button is pressed, the ESP32 read up to date values of temperature and sensor from DHT sensor. The values are then displayed on the OLED screen, proving immediate visual feedback to the user.

4. **Dynamic HTML webpage generation:**

The webpage is generated dynamically inside the program. The reading are inserted into the HTML content before sending it to the browser.

5. **Purpose of meta refresh in the webpage:**

The meta refresh in the webpage is used to automatically reload the web page after a fixed amount of time. This is for the sensor values to get update automatically without requiring the user to manually refresh the browser.

6. Common issues in ESP32 webserver projects and their solutions

Common issues in ESP32 web server projects include WIFI connection failures, incorrect password, incorrect ip address, slow server responses and sensor reading error. These problems can be resolved by verifying WI-FI credentials, checking the ip address through the serial monitor, avoiding long delays in the code, and ensuring proper wiring and timing.

Question-2 Blynk Cloud Interfacing (blynk.cpp)

Part-A: Short Questions

1. **What is the role of Blynk Template ID in an ESP32 IoT project? Why must it match the cloud template?**

The Blynk Template ID links the ESP 32 devices to a specific projects template on the Blynk Cloud. This is used to define the dashboard layout, widget, and virtual pins used in the projects. If this template does not match the cloud template, then the device will fail to sync with the correct dashboard.

2. **Differentiate between Blynk Template ID and Blynk Auth Token.**

Blynk Template ID identifies the project template on the Blynk Cloud, while the Blynk Auth Token identifies and authenticated a specific device. Template defines the interface, whereas the auth token allows secure communication between the ESP32 and Blynk.

3. **Why does using DHT22 code with a DHT11 sensor produce incorrect readings? Mention one key difference between the two sensors.**

If we use DHT22 code with DHT11 sensor produces incorrect readings because both sensor have different data formats and measurement ranges. DHT 22 has higher accuracy and support wider temperature range as compare to DHT11.

4. **What are Virtual Pins in Blynk? Why are they preferred over physical GPIO pins for cloud communication?**

Virtual Pins are software-based pins used to send and receive data between the ESP32 and the Blynk cloud. They are preferred because they are not tied to physical hardware pins and allow flexibility without hardware limitations.

5. **What is the purpose of using Blynk Timer instead of delay () in ESP32 IoT applications?**

Blynk Timer is used to execute tasks at specific intervals without blocking the program. Unlike delay(), it allows the ESP32 to continue handling cloud communication and other tasks smoothly, ensuring stable and responsive IoT operation.

Part-B: Long Question

Explain the complete workflow of interfacing ESP32 with Blynk Cloud to display temperature and humidity values.

Your answer should include:
- Creation of Blynk Template and DataStream
- Role of Template ID, Template Name, and Auth Token
- Sensor configuration issues (DHT11 vs DHT22)
- Sending data using Blynk.virtualWrite()
- Common problems faced during configuration and their solutions.

1. Blynk Template and DataStream Setup:

A Blynk template is established on the Blynk Cloud to outline the project structure. DataStreams, for humidity and temperature are. Connected to virtual pins, which serve to gather sensor data from the ESP32.

2**. Role of Template ID, Template Name, and Auth Token:**

The Template ID links the ESP32 to the Blynk project whereas the Template Name designates the template on the cloud. The Auth Token serves as a key enabling secure interaction, between the ESP32 and the Blynk Cloud.

3. Sensor configuration issues (DHT11 vs DHT22):

Choosing the sensor in the code is crucial for precise readings. Employing DHT22 code with a DHT11 sensor results in values because of variations, in accuracy and measurement range.

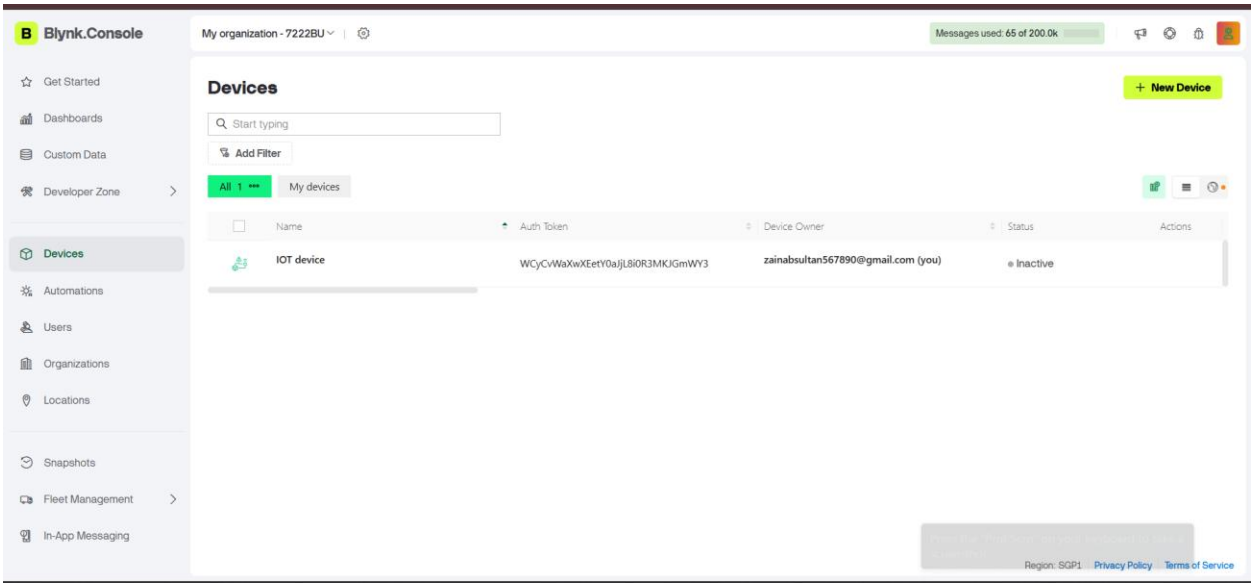
4. Sending data using Blynk.virtualWrite():

The Blynk.virtualWrite() function transmits temperature and humidity readings to the Blynk Cloud. Every measurement corresponds to a designated pin enabling live visualization, on the dashboard.

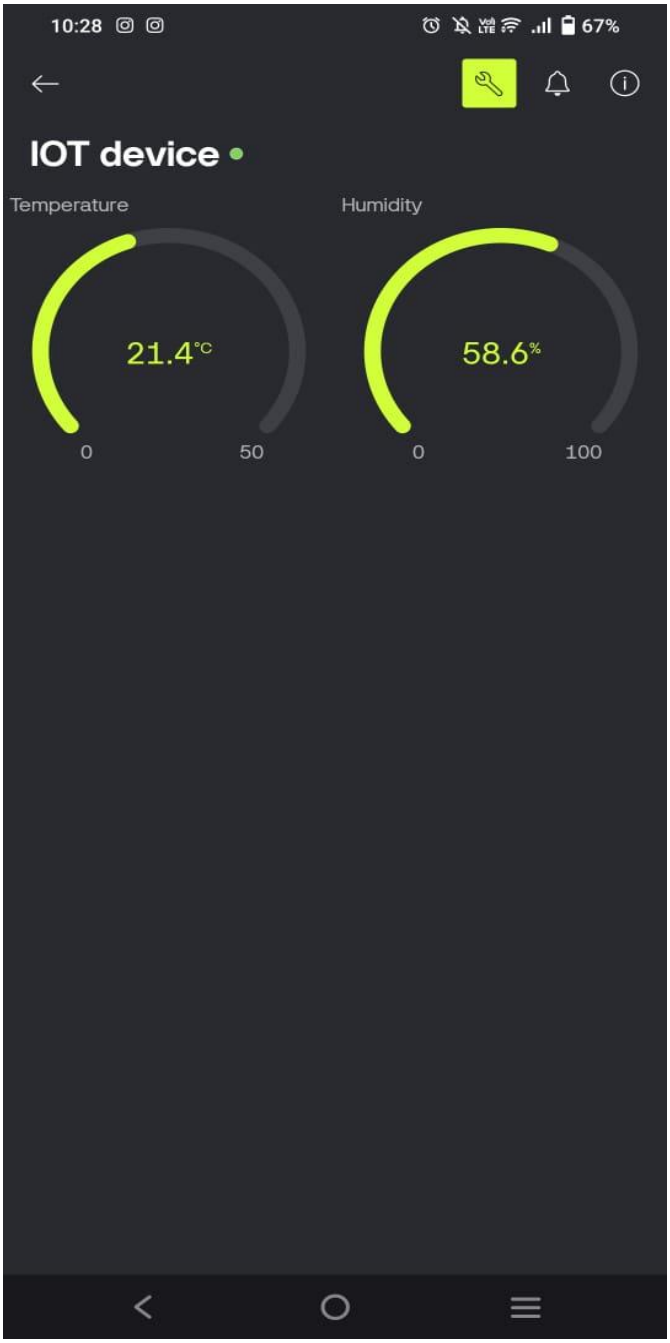
5. Common problems faced during configuration and their solutions:

Common problems include incorrect cloud credentials, wrong virtual pin mapping, Wi-Fi connection issues, and sensor mismatch. These can be fixed by verifying IDs and tokens, checking pin assignments, and ensuring stable network connectivity.

Web dashboard:



Mobile dashboard:


10:28
67%
IOT device
Temperature
21.4°C
0
50
Humidity
58.6%
0
100


VCC GND SCL SDA
Environment Node
Temp: 21.4 C
Hum : 55.9 %
BTN -> manual update