National Textile University,

Faisalabad

Department of Computer Science

Name:	Maryam Hamid
Class:	BSCS ‘5A’
Registration No:	23-NTU-CS-1046
Course Name:	Embedded IOT
Submitted To:	Sir Nasir
Submission Date:	21 December 2025

Assignment 1

Question 1 (webserver.cpp)

Part A — Short Questions

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

   This line of code creates an object named server that handles incoming web requests. Port 80 is the default port used for HTTP web requests. By using port 80, the ESP32 acts like a standard website, so you can just type its IP address into a browser's URL.

2. **Explain the role of server.on("/", handleRoot); in this program.**

   This is essentially the "route" setter. This tells the ESP32 that as soon as a person accesses the main webpage (denoted as "/" in the URL), according to the route, the corresponding function called "handleRoot" should be executed. This function is what generates the HTML you see on your screen.

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

   This function needs to run constantly so the ESP32 can "listen" for new people trying to connect to the website. It listens for incoming web requests and responds to them.It must run repeatedly inside the loop. If you remove it, the ESP32 will still be connected to Wi-Fi, but it will never respond to any web requests, making the web server effectively dead

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

   This is how the ESP32 talks back to your browser. The 200 is the HTTP success code (meaning "OK"), text/html tells the browser it is receiving a webpage, and html is the actual string containing the website's code. it sends an HTML web page to the browser with a success response.

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

When using the values that have been measured, the website loads really fast because it only retrieves data that is already there in the ESP32's memory. Instead taking the “fresh” value in the handleRoot() method guarantees that the data is 100% up to date, though it makes the website take a little longer in response because the DHT sensor doesn't respond instantly.

Part B— Long Questions

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

This system uses an ESP32 and DHT11 sensor to read Humidity and Temperature , display its values on an OLED screen, connected to Wi-Fi and show that same data on a webpage using built in web server

1. **ESP32 Wi-Fi Connection and IP Address Assignment**
   The system starts by connecting the ESP32 to a Wi-Fi network using the provided SSID and password. ESP32 keeps checking the connection if it is disconnected. Once connected, the router assigns it a local IP address, which is printed to the Serial Monitor so the user knows where to find the website. Just copy paste the IP to browser.
2. **Web Server Initialization and Request Handling**
   The web server is then initialized on port 80 (default port).Like when a user pastes the ESP32 IP address in a browser, a request is sent to the server. This request is handled by a function “handleRoot()”, which prepares and sends an HTML response back to the browser. Inside loop(), function “server.HandleClient() listen for incoming request and responds them repeatedly.
3. **Button-Based Sensor Reading and OLED Update Mechanism**
   On hardware circuit , push Button is connect to esp32. The code constantly checks if a physical button has been pressed. When pressed, the ESP32 reads temperature and humidity values from the DHT11 sensor. Then display these values on OLED Screen and saves them to memory.
4. **Dynamic HTML Webpage Generation**
   When a user visits the IP address, the handleRoot() function dynamically builds an HTML page. On this dynamic HTML webpage , the values of sensor changes accorfing to real time readings. The ESP32 sends the updated HTML content that displays the latest values from DHT sensor.
5. **Purpose of Meta Refresh in the Webpage**
   I included a "meta refresh" tag in the HTML which tells the browser to automatically reload every 5 seconds. It allows the webpage to update values automatically without user refreshing the page manually.
6. **Common Issues in ESP32 Webserver Projects and Their Solutions**

Common Esp32 webserver issues include **Wi-Fi connection failure** (Solution: by checking SSID/password or checking laptop internet connection to same internet) **Sensor not responding** (Solution: by checking wiring , GPIO or sensor type like DHT 11). **OLED not displaying** (Solution: by checking I2C address and using libraries properly). **Slow Response** (Solution: Reduce delays and code)

Question 2 (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 template ID uniquely identifies the Blynk template on the cloud. It must match exactly with the cloud template so the ESP32 knows which template to use for displaying data and linking virtual pins.

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

   Template ID identifies the cloud template structure (like layout, datastremas, widgets).

   Auth Token is a unique key for the device that allows the ESP32 to connect securely to BLYNK cloud and send or receive data.

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

   Using DHT22 code with a DHT11 sensor connected to ESP32 , it start giving wrong readings because the sensors(11 or 22) have different temperature or humidity ranges.

   For example, DHT11 measures 0-50°C, while DHT22 meansure -40 to 80°C.

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

   Virtual Pins (V0,V1,etc.) are software pins used to send or receive data between ESP32 and BLYNK Cloud. They are preferred over Physical GPIO pins because they don't interfere with the hardware wiring and it allows you to send any data(string or decimal) that simple physical pins (ON/OFF) can't handle .

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

   Using delay() is bad because it makes everything stop while waiting which cause it to lose connection to BLYNK server and crash it.

   BlynkTimer allows you to keep executing code without blocking the program. It stays connected and keeps sending sensor data

Part B—Long Questions

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

1. **.Create Blynk Template and Datastreams:**
   On the Blynk Cloud, create a template for project (“dht”)
   Add Datastreams for tempersature and humidity. They define how data will be sent and displayed on dashboard as a widget.
2. **Role of Template ID, Name, and Auth Token:**
   - **Template ID** links the ESP32 to correct cloud template . It define Type of project
   - **Template Name** is just user defined name given to project template (“dht”) It helps to organize different projects with BLYNK.
   - **Auth Token** is a unique key that allows ESP32 to connect and send data to Blynk Cloud. **Blynk.begin(BLYNK_AUTH_TOKEN, ssid , pass);** it requires AuthToken to begin Blynk.
3. **Sensor Configuration Issues (DHT11 vs DHT22):**
   Make sure the sensor type in the code (DHT11) matches to hardware sensor
   As we have DHT11 , so we will intitailze DHT11 in the start of code.
   Using DHT22 code with a DHT11 sensor, it start giving wrong readings because sensors have different ranges.
4. **Sending Data using Blynk.virtualWrite():**
   Read temperature and humidity from the DHT sensor and display it on the OLED
   Send data to the cloud to its correspondence virtual pins using Blynk.virtualWrite(V0, temp) and Blynk.virtualWrite(V1, hum).
   V0 and V1 are virtual pin linked to widgets on the blynk app (V0 for temperature and V1 for Humidity)
5. **Common Problems and Solutions:**

- **Blynk not connecting:** Check Auth Token, Wi-Fi SSID/password, and cloud server.
- **Incorrect readings:** Check sensor type matches code.
- **OLED not showing data:** Check I2C pins and address.