National Textile University, Faisalabad

Department of Computer Science

Name:	**Muhammad Haseeb**
Class:	**BSCS 5th (Section – B)**
Registration No:	**23-NTU-CS-1069**
Lab Report:	**Homework-01_AfterMid**
Course Name:	**Embedded IOT Systems**
Submitted To:	**Sir. Nasir Mahmood**
Submission Date:	**16th Dec – 2025**

Question:01

Part A: Short Questions

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

Ans: This sets up a web server on the ESP32 that listens for requests and Port 80 is the normal HTTP port.

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

Ans: This line defines what happens when someone visits a specific URL. "/" is the **root path** (the main page, like http://ESP32_IP/). Like "Whenever someone asks for the main page, run this code and give them the result."

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

Ans: It is placed inside the loop() so the ESP32 keeps checking for incoming requests. Without it, the page won't load.

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

Ans: It just sends back a response with status 200(OK), tells the browser it's HTML, and sends the page content. Without this line, the browser would not get any content to display.

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

Ans: Showing the last values just uses what's already saved, while taking a fresh reading actually reads the DHT22 again and updates everything.

Part B: Long Question

1. ESP32 Wi-Fi Connection and IP Assignment

Ans:

- Wi-Fi connection is established in the `setup()` function using:

```
WiFi.begin(ssid, password);
```

ESP32 continuously attempts to connect until `WiFi.status() == WL_CONNECTED`.

- Once connected, the ESP32 is assigned an **IP address** by the router (via DHCP).

The assigned IP is stored internally and can be obtained using:

```
WiFi.localIP();
```

In short: Wi-Fi connection gives the ESP32 a network address so it can act as a mini web server accessible to any device on the same network.

2. Web Server Initialization and Request Handling

Ans:

- The web server is created with: `WebServer server(80).`

80 is the default HTTP port.

- A route handler is defined: `server.on("/", handleRoot.`

This tells the server to run `handleRoot()` whenever someone visits the root URL `/`.

- Inside the `loop()`, `server.handleClient();` continuously checks for incoming HTTP requests and responds.

3. Button-Based Sensor Reading and OLED Update

Ans:

- **Button setup:** `INPUT_PULLUP` mode ensures the button reads `HIGH` when not pressed and `LOW` when pressed.
- **Edge detection:** `If (lastButtonState == HIGH && currentButtonState == LOW)`
- **Debouncing:** A small delay (`delay(50)`) ensures the button press is read cleanly without multiple triggers.
- **Sensor reading:** `readDHTValues()` reads temperature and humidity from the DHT22.
- **OLED update:** `showOnOLED()` clears the display and shows the latest readings.
- **Stored values:** Last readings are saved in `lastTemp` and `lastHum`, which are used for the web page.

4. Dynamic HTML Webpage Generation

Ans:

- The `handleRoot()` function builds an HTML page as a **string**.
- Page includes:
  - Temperature and humidity readings.
  - Instructions for pressing the button to update readings.
- HTML is sent using:

  ```
  server.send(200, "text/html", html);
  ```

  This allows any device on the same network to view the readings live.

5. Purpose of Meta Refresh in the Webpage

Ans:

- Meta refresh is used in the `<head>` of HTML:

```
<meta http-equiv='refresh' content='5'>
```

It automatically reloads the page every 5 seconds.

- **Purpose:** Users see updated readings without manually refreshing.Without it, the page would show only the last reading captured at the time of opening.

6. Common Issues in ESP32 Webserver Projects and Solutions

Issue	Cause	Solution
Web page not loading	`server.handleClient()` missing or Wi-Fi not connected	Ensure `server.handleClient();` is in `loop()`, check Wi-Fi credentials and IP
Slow page refresh / stale data	Page only uses last stored value	Either read sensor in `handleRoot()` or press the button to update readings
IP changes after reboot	DHCP assignment	Either use static IP or check current IP each boot using `WiFi.localIP()`

Question:02

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?

Ans:

The Blynk Template ID links the ESP32 device to a specific template on the Blynk cloud. It must match the cloud template so the device knows which dashboard layout, virtual pins, and settings to use.

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

Ans:

The Template ID identifies the project/template on the Blynk cloud, while the Auth Token uniquely authenticates and authorizes a specific device to connect to that template.

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

Ans:

DHT11 and DHT22 use different internal data formats and accuracy ranges. A key difference is that DHT22 provides decimal (higher-resolution) reading**s**, while **DHT11 gives** only integer valu**es**, so using DHT22 code with DHT11 causes incorrect interpretation.

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

Ans:

Virtual Pins are software-defined pins used to send and receive data between the ESP32 and the Blynk cloud. They are preferred because they are not tied to physical GPIOs and allow flexible, hardware-independent cloud communication.

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

Ans:

BlynkTimer allows tasks to run periodically without blocking the program. Unlike delay(), it keeps the ESP32 responsive, ensures Blynk.run() continues working, and prevents Wi-Fi or cloud disconnections.

Part-B: Long Question

1. Creation of Blynk Template and Datastreams

First, a Template is created on the Blynk Cloud platform.
The template defines how data will be displayed in the Blynk mobile or web dashboard.

- Widgets (such as gauges or labels) are added to the dashboard.
- Datastreams are created and linked to Virtual Pins:
  - Example: **V0, V1**
- Each datastream defines:
  - Data type (float / integer)
  - Min–max range
  - Update mode (push or periodic)

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

- **Template ID**
  Identifies which cloud template the ESP32 belongs to. It ensures the device connects to the correct dashboard and datastreams.
- **Template Name**
  A human-readable name for the project, mainly used for identification and organization.
- **Auth Token**
  A unique security key generated for each device. It authenticates the ESP32 with the Blynk Cloud and allows data exchange.

All three must be correctly defined in the code; otherwise, the device will fail to connect.

3. Sensor Configuration Issues (DHT11 vs DHT22)

DHT11 and DHT22 sensors are **not interchangeable in code**.

- **DHT11**
  - Lower accuracy
  - Integer-only temperature and humidity values
  - Smaller operating range
- **DHT22**
  - Higher accuracy
  - Provides decimal values
  - Wider temperature and humidity range

If DHT22 code is used with a DHT11 sensor (or vice versa), the ESP32 misinterprets sensor data, resulting in **incorrect or NaN readings**.

4. Sending Data

After reading the sensor values, data is sent to the Blynk Cloud using:

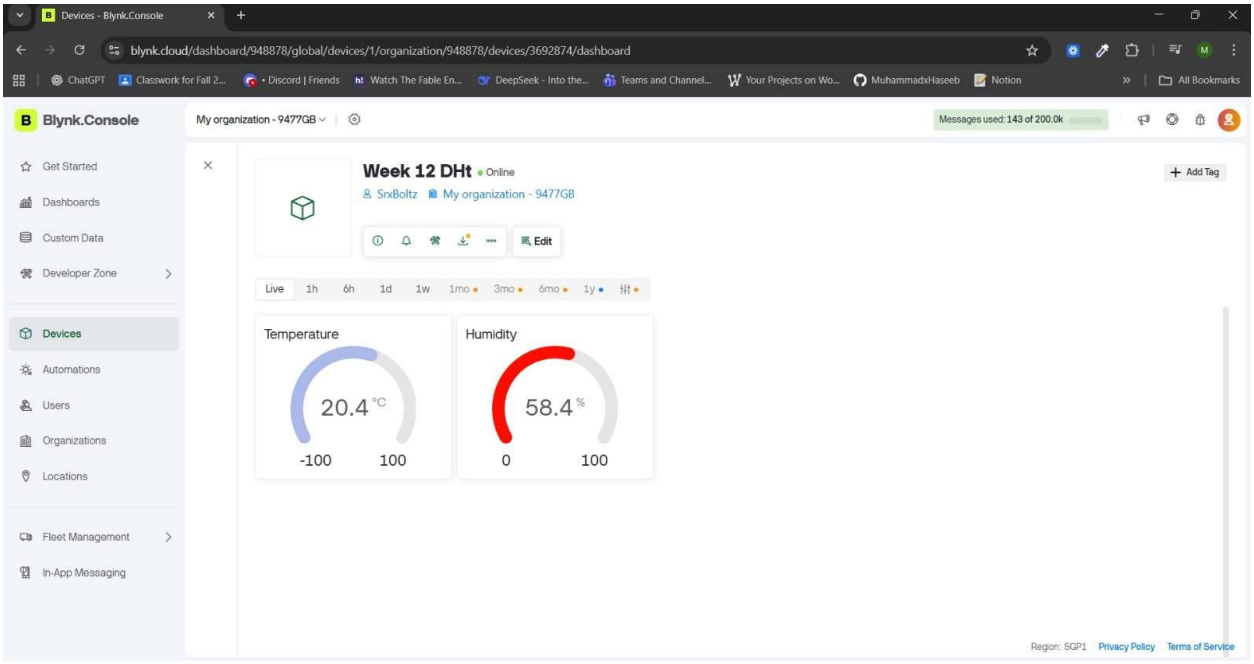
```
Blynk.virtualWrite(V0, temperature);
Blynk.virtualWrite(V1, humidity);
```

- Virtual pins act as a bridge between the ESP32 and Blynk dashboard widgets.
- This method sends sensor data directly to the cloud without relying on physical GPIO pins.
- The dashboard updates in real time whenever new data is pushed.

5. Common Problems and Their Solutions

Problem	Cause	Solution
Device shows offline in Blynk	Wrong Auth Token or Template ID	Verify token and template details
No data on widgets	Virtual pins mismatch	Ensure code and datastream pins match
ESP32 disconnects frequently	Use of `delay()`	Replace `delay()` with `BlynkTimer`

Cloud Based:


Devices - Blynk.Console
blynk.cloud/dashboard/948878/global/devices/1/organization/948878/devices/3692874/dashboard
Blynk.Console
My organization - 9477GB
Messages used: 143 of 200.0k
Get Started
Dashboards
Custom Data
Developer Zone
Devices
Automations
Users
Organizations
Locations
Fleet Management
In-App Messaging
Week 12 DHt
Online
SrxBoltz
My organization - 9477GB
Edit
Add Tag
Live
1h
6h
1d
1w
1mo
3mo
6mo
1y
Temperature
20.4 °C
-100
100
Humidity
58.4 %
0
100
Region: SGP1
Privacy Policy
Terms of Service

Mobile Based: