Name:	Muhammad Ibtisam Butt
Reg No:	23-NTU-CS-1269
Section:	BSAI^{5TH}
Subject	Embedded IOT
Submitted To:	Sir. Nasir Mehmood

HomeWork-01 (After Mid)

Question 1

Part A:

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

It creases web-server object on ESP32 that listen for incoming HTTP requests.

- **Purpose:** It allow ESP32 to act as a web server, which will serve webpages to clients on browser(laptop,phone)
- **Port 80:**
  - It is Default **HTTP** port
  - Browsers automatically uses this port, when an IP addess is typed in the browser without assigning a port

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

This line registers a **URL route** for the web server

- / means the **root URL** (home page)
- **handleroot** is the function that runs when someone opens the ESP's IP address in a browser

When a client request **http://ESP32_IP/** , the function **handleroot()** is executed and sends the HTML page.

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

server.handleClient(); is for

- Listeing incoming client requests
- Process HTTP requests
- Send responses

Why inside loop()?

- The ESP32 check for new requests constantly.
- Loop() runs repeatedly, so server stays responsive

If removed:

- The webserver will stop responding
- The browser will show:
  - Page not loading
  - Connection timeout

ESP32 will still run other code,but web interface will fail.

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

200: HTTP **status code meaning "Successful"**

This line sends data to the web browser .

- **200** means request is successful
- **text/html** means the content is a webpage
- **html** contains the webpage code

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

Aspect	Last Measured Values	Fresh Reading in handleRoot()
Sensor Access	Uses stored values	Reads sensor every page refresh
Speed	Faster	Slower
DHT safety	Safe	Risky
Accuracy	Slightly old	Real-time
Reliability	High	Can cause DHT errors

DHT22 sensors should **not be read frequently**. Reading inside handleRoot() may cause:

- NaN values
- Sensor communication failure

That's why your code wisely updates values **only on button press**.

Part B:

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

- First, the **ESP32** connects to a Wi-Fi network using **SSID** and **password.** After successful connection, the router assigns an **IP address** to the ESP32.
- Then, the web server is started using **port 80.** When a user enters the **ESP32 IP address** in a browser, the request is received and handled by the **server.**
- A **button** is used to read temperature and humidity from the **DHT sensor**. When the button is pressed, the **ESP32** reads the sensor values and shows them on the **OLED display**.
- The webpage is created using **HTML code** stored in a string. Sensor values are added dynamically inside the **HTML** code so updated data is shown.
- A **meta refresh** tag is used in the webpage to automatically reload the page after some seconds, so new values appear without manual refresh.

Common Issues and Solutions

- **Wi-Fi not connecting** → check SSID and password
- **Webpage not loading** → ensure server.handleClient() is in loop
- **Wrong sensor values** → check DHT type and wiring
- **Slow response** → avoid delay() in web server code

Question 2

Part A:

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** is used to identify the IoT project created on the **Blynk Cloud**.
It connects the ESP32 code with the correct project dashboard on Blynk.

Role of Blynk Template ID:

- Links ESP32 firmware with the correct Blynk Cloud project
- Ensures correct widgets and datastreams are used
- Helps Blynk Cloud recognize the project

Why it must match:
If the Template ID in the code does not match the cloud template:

- ESP32 will not connect to Blynk Cloud
- Virtual Pins will not work
- Sensor data will not appear on the dashboard

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

Feature	Template ID	Auth Token
Purpose	Identifies the project template	Authenticates a specific device
Scope	Same for all devices using template	Unique per device
Stored in	Blynk Cloud template	Device settings
Used for	Project structure	Secure device connection

- **Template ID = Project identity**
- **Auth Token = Device password**

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

- DHT11 and DHT22 sensors work differently and have different data formats.
- Using **DHT22 code with a DHT11 sensor** causes incorrect or unstable readings.

- **Key difference:**
  - **DHT11**
    - Temperature range: **0–50** °C
    - Integer values only
  - **DHT22**
    - Temperature range: **−40 to 80** °C
    - Decimal precision

Because of this difference, the library reads the data incorrectly if the wrong sensor type is selected.

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 Blynk Cloud

Example:

```
Blynk.virtualWrite(V0, t);
Blynk.virtualWrite(V1, h);
```

Why preferred:

- They are not connected to actual ESP32 hardware pins
- They allow flexible data transfer
- They work easily with Blynk Cloud and mobile app
- Multiple widgets can use the same Virtual Pin

Virtual Pins act as **communication channels between device and cloud**.

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

BlynkTimer is used to perform tasks at regular intervals without stopping the program.

Why **delay()** should be avoided:

- It blocks Wi-Fi communication
- It disconnects ESP32 from Blynk Cloud

Advantage of BlynkTimer:

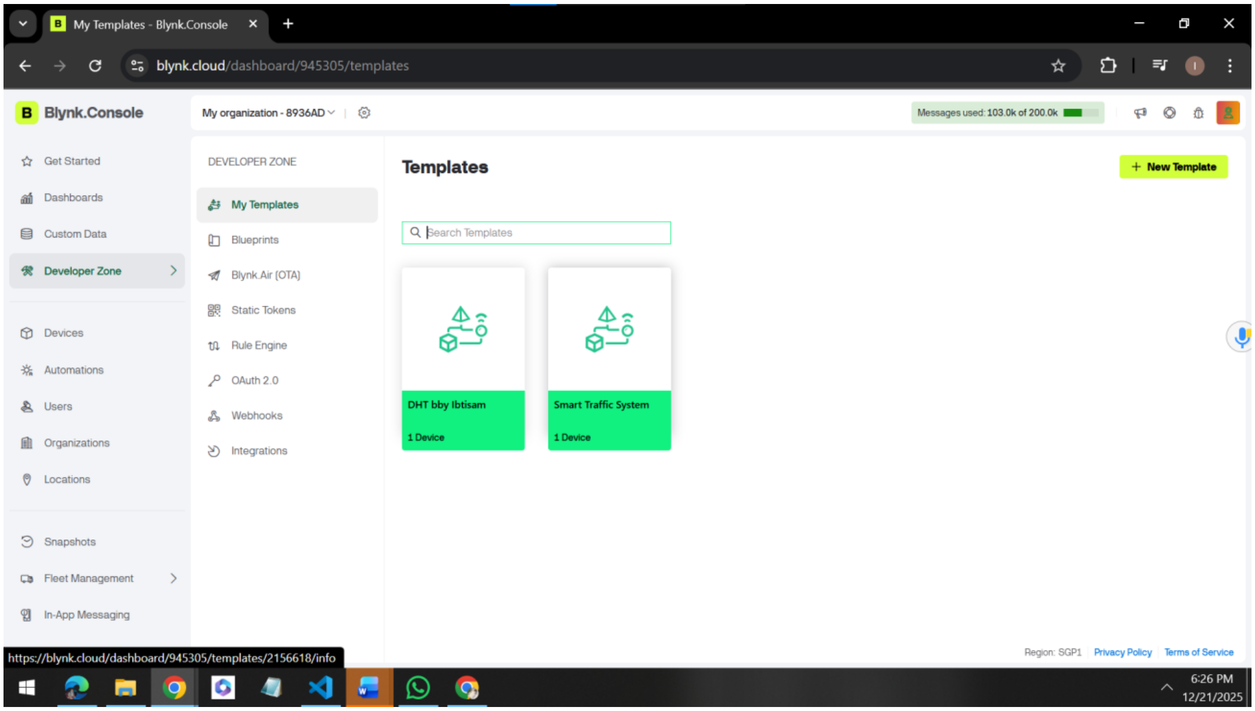
- Keeps ESP32 responsive
- Maintains stable cloud connection
- Allows multitasking

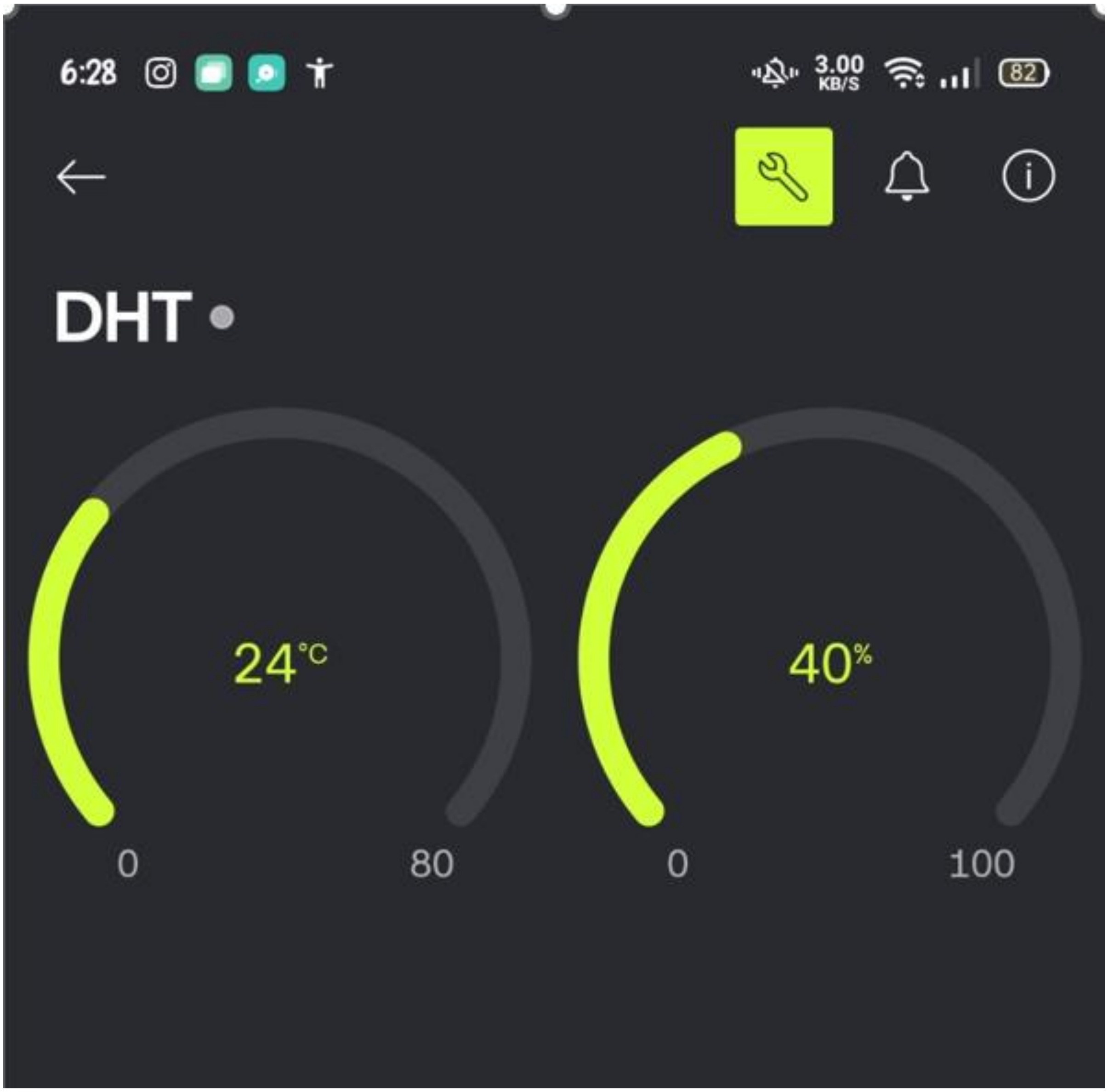
Part B:

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

- First, a **Blynk Template** is created on the Blynk Cloud.
- Datastreams are added for **temperature** and **humidity**, usually using Virtual Pins such as **V0 and V1.**
- Widgets like labels or gauges are connected to these datastreams.
- The **Template ID, Template Name,** and **Auth Token** are added in the ESP32 program.
- These details allow the ESP32 to connect securely with the Blynk Cloud.
- The **DHT sensor (DHT11 or DHT22**) is connected to the ESP32 and configured correctly in the code.
- Incorrect sensor configuration can cause wrong readings.
- The ESP32 **reads temperature** and **humidity** values from the sensor.
- These values are sent to the Blynk Cloud using **Blynk.virtualWrite().**

- A **BlynkTimer** is used to send data periodically without blocking the program.
- The cloud receives the data and updates the widgets on both the **Blynk mobile app** and **web dashboard** in real time.
- This system allows users to monitor temperature and humidity remotely using their mobile phone or browser.




6:28
3.00 KB/S
82
DHT
24°C
0
80
40%
0
100