National Textile University, Faisalabad

Department of Computer Science

Name:	Hajra Ahmad
Class:	BS CS “A”
Registration No:	**23-NTU-CS-1034**
Course Name:	**EMBEDDED IOT SYSYTEM**
Assignment	**2**
Submitted To:	**Dr. Nasir Mahmood**
Submission Date:	December,2025

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?**

WebServer server(80) creates a webserver object on Esp32.The server listens for HTTP requests on port 80.Port 80 is the default port for HTTP web traffic. When a user enters the esp32 ip address in the browser the request comes through port 80.

2. **Explain the role of server.on("/", handleRoot); in this program.**
   - "/" represents the the root URL basically the homepage.
   - When the browser access the esp32 ip Address the webserver automatically calls the handleRoot() function.
   - HandleRoot generate an html webpage and than displays the temperature and humidity and than sends the webpage to the browser.

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

   server.handleClient():
   This functions checks if a client on the browser requests a webpage. It processes HTTP requests continuously.
   Why in loop:
   It must be inside loop so that the esp32 can keep responding.
   If it will be removed the esp32 will connect to the wifi the server will start and the browser will never receive a webpage.

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

This line sends the generated Html page from Esp32 to the browser.

- 200 tells the HTTP status which means successful
- "text/html" tells about the data type being sent.

- Html is the string which contain the content of the web page.

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

  The webpage displays last stored values sensor values.
  Values are updated only when the button is pressed.

Last Measured Sensor Values	Fresh DHT reading Inside handlroot()
Fast Page loading	Slow Page Loading
No DHT timming issue	May cause NaN errors
Stable sensor output	DHT sensor may fail due to frequent dht readings.

Part B: Long Question

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

- ESP32 Wi-Fi connection process and IP address assignment

1. Esp32 connects to the wifi using this line:
   Wifi.begin(ssid,password);
2. Router assigns an ip Address automatically.
3. Then ip address is displayed on the serial monitor and the OLED screen.
4. Then this ip address is used to access the esp32 webpage.

- Web server initialization and request handling

Following steps are involved in the web server initialization:

1. Create server object on the port 80.
2. Then define the url handler using server.on().
3. Then start the server using the server.begin().
4. Continuously handle browser request using server.handleclient()
5. When the browser request the line including / and handleroot() executes.

- Button-based sensor reading and OLED update mechanism

Button is configured with the following line of code:

Pinmode(BUTTON_PIN, INPUT_PULLUP);

1. In this line of code the button logic is ACTIVE LOW which will detect falling edge which is HIGH to LOW.
2. When the button is pressed the DHT reads the temperature and humidity.
3. The values are stored in **lastTemp** and **lastHum** and Oled screen is updated using **showOnOLED().**
4. Webpage will automaticallyupdates on refersh.

- Dynamic HTML webpage generation

Inside handleRoot()

1. HTML is created dynamically using a string.
2. Sensor values are inserted in the HTML.
3. Webpage will display the temperature and Humidity.

- Purpose of meta refresh in the webpage

It will automatically refresh the webpage every 5 seconds. Updates the sensor values without using the reload button. It is very usefull for live monitoring.

- Common issues in ESP32 webserver projects and their solutions

Web page not loading
Cause: Missing handleclient().
Solution: Add in loop().

NaN Readings
Cause: Fast DHT access.
Solution: Read sensor less frequently.

OLED blank
Cause: Wrong I2c pins**.**
Solution: use GPIO 21 and 22

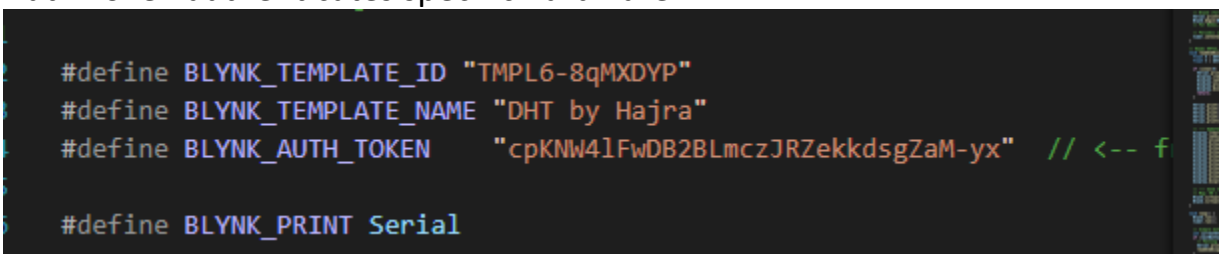
Button not Working
Cause: Missing PULLup
Solution: use INPUT_PULLUP

WiFi not connecting
Cause: Wrong ssid or password
Solution: verify the credentials properly.

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?
   It identifies the cloud dashboard configuration and Esp32 must match the template created on blynk cloud. If it doesnot match the device will not connect.
2. Differentiate between Blynk Template ID and Blynk Auth Token.

- The template id links the device to cloud UI layout.
- Auth Token authenticates specific hardware.
- 

3. Why does using DHT22 code with a DHT11 sensor produce incorrect readings?
   Mention one key difference between the two sensors.
   - DHT11 takes integer value only.
   - DHT22 takes decimal and higher accuracy numbers.
   - Using wrong type can cause incorrect readings or NaN values
4. What are Virtual Pins in Blynk? Why are they preferred over physical GPIO pins for cloud communication?

- Virtual pins in blynk are software pins used for cloud communication they are not linked to physical GPIOs.
- They are preffered over physical pins because they are cloud friendly,

Hardware independent and gives easier data visualization.

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

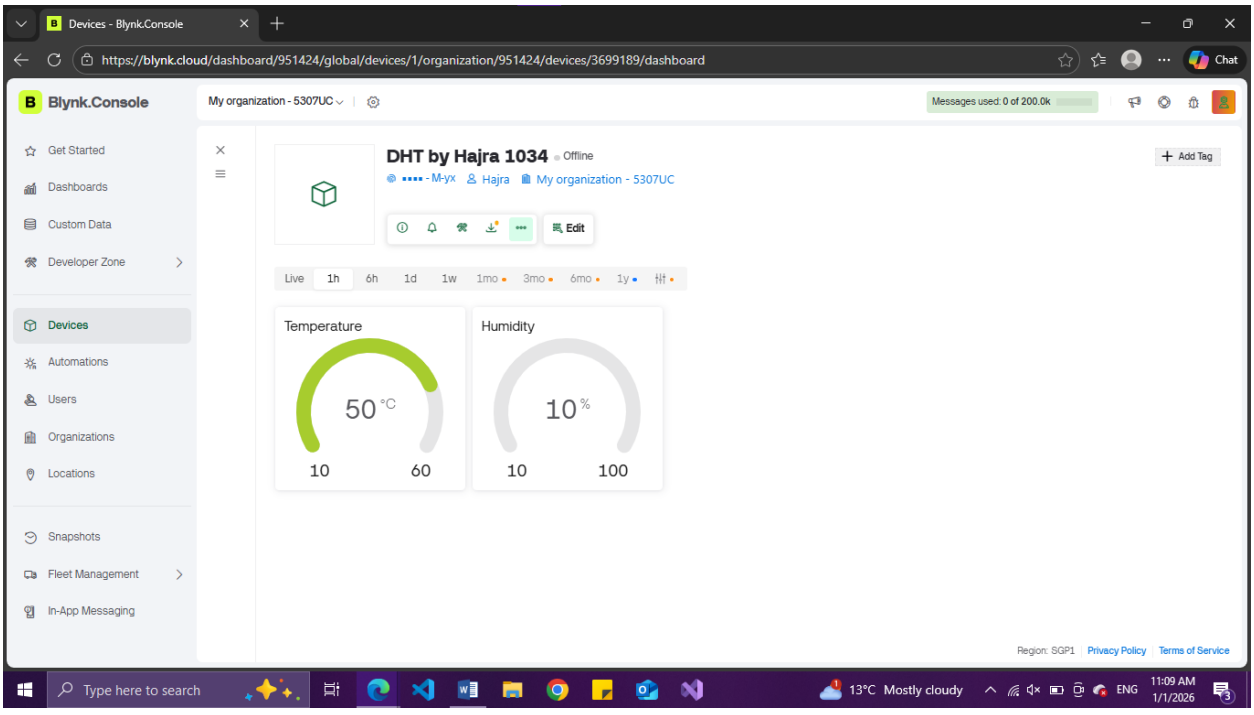
- Delay () blocks the execution
- BlynkTimer allows non blocking code , stable wifi and cloud connection and also multiple timed tasks.

Part-B: Long Question

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

- Creation of Blynk Template and Datastreams

- First of all we created a account on blynk cloud app.
- Then we login using that accounts on our mobile phone blynk app.
- Then on the cloud we created a template.
- Added the Datastreams V0 Temperature and V1 Humidity.
- Then we design our mobile dashboard using the widgets.



- Role of Template ID, Template Name, and Auth Token

```
  // --- Send to Blynk (Virtual Pins) ---
  // Map: V0 = Temp, V1 = Humidity
  Blynk.virtualWrite(V0, t);
  Blynk.virtualWrite(V1, h);
}

void loop() {
  Blynk.run();
  timer.run();
```

Template ID : Links the firmware to the dashboard.
Template Name : For the identification.
Auth Token : Authenticates Esp32 device with the blynk cloud
All three must match the cloud configuration.

- Sensor configuration issues (DHT11 vs DHT22)

If we will apply wrong DHT Type the sensor will give us incorrect values which will affect our project.
Correct setting
#define DHTTYPE DHT22

```
// ------------ Pins (match your Wokwi diagram) ------------
#define DHTPIN   23
#define DHTTYPE  DHT22
```

- Sending data using Blynk.virtualWrite()

```
  // --- Send to Blynk (Virtual Pins) ---
  // Map: V0 = Temp, V1 = Humidity
  Blynk.virtualWrite(V0, t);
  Blynk.virtualWrite(V1, h);
}

void loop() {
  Blynk.run();
  timer.run();
```

- Sends the sensor value to the cloud.
- Then widgets are updated in real time.

- Common problems faced during configuration and their solutions

- **Device offline**: check auth token.
- **No data on the app**: Verify virtual pins.
- **NaN Readings**: correct DHT type.
- **App disconnects**: remove delay() and use timer.
- **Oled not updating**: check i2c pins.