National Textile University, Faisalabad

Department of Computer Science

Name:	Amna
Class:	BSCS-5A
Registration No:	23-NTU-CS-1013
Assignment Name:	Homework-01 – After mid
Course Name:	Embedded IOT Systems
Submitted To:	Sir Nasir Mahmood
Submission Date:	20.12.2025

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?**

WebServer server (80) creates a web server object that responds to client requests. **Port 80** is the default port. It is used for HTTP web communication.

For example, when you enter a **http: // 192.168.1.100** the browser automatically connects to port 80. It gives the users access of ESP32 through a web browser.

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

This Line connects root **webpage (/)** to the **handleRoot** function. Whenever someone enters the ESP32 IP address in a browser. This function will be called and will send the web page or display the temperature and humidity.

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

The main reason behind placing **server. handleClient ()** inside the loop function is so the ESP32 can continuously check and handle the incoming client requests.

If **it's not placed** inside the loop, the web server will not respond, and the webpage will neither load nor update and users will get connection timeout errors when trying to access the webpage.

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

This statement send the webpage to browser with three things.

- **200** = request has been successfully delivered.
- **text/html** = this tells the browser it's a webpage it means content is HTML format.
- **html** = it's an actual webpage data that will be displayed (temperature and humidity data)

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

Last measured sensor values	**Fresh DHT reading inside handleRoot ()**
It uses the previously stored values of temperature and humidity.	It reads the sensor every time the page loads.
As we take no new readings. So, its faster	As we take readings every time. So, it's taking time and slower (~2 sec delay).
Sensor is accessed only when button is pressed. So, stress on sensor is low.	Sensor is accessed frequently. So, stress on sensor is high.
It quickly displays the existing data.	It always shows up-to-date readings.

Part B: Long Question

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

The system uses ESP32 microcontroller to measure temperature and humidity readings using a DHT22 sensor.

There are two ways to display the readings.

1. It can be displayed on the OLED screen.
2. It can be displayed on webpage that can be accessed through web browser using ESP32 IP address.

ESP32 Wi-Fi connection process and IP address assignment

- In **setup () function**, using **WiFi.begin(ssid, password)** ESP32 connects to the Wifi.
- It checks the status repeatedly with **Wifi.status ()**, until it's successfully connected.
- After the connection, the router gives the IP address which gets through **Wifi.localIP().**
- **OLED Screen** and **Serial Monitor** both will display this IP address.
- This IP address will be used to access the **ESP32 webserver** through a browser

Web server initialization and request handling

- When the connection is established, webserver is created on port 80 by using **WebServer server(80).**
- In **server.on("/", handleRoot)**, the root URL / is linked to handleRoot.
- The server started working with **server.begin().**
- In the loop function**,** **server.handleClient()** handles and responds to incoming browser requests. It ensures that webpage properly loads and updates

Button-based sensor reading and OLED update mechanism

- In ESP32, push button connected to **INPUT PULL UP**.
- In **loop function**, when the program detects a button press by checking **HIGH -> LOW, Falling edge**.
- When button is pressed, ESP32 reads temperature and humidity values from **readDHTValues().**
- **showOnOLED()** updates the OLED display.
- This makes sures that display and updation only happens when button is pressed.

Dynamic HTML webpage generation

- The webpage is dynamically generated inside the function **handleRoot()**.
- HTML code creation used the string variable. It stored **temperature** values in **(lastTemp)** and **humidity** values in **(lasthum).** These values are inserted into webpage.
- If valid readings are not available. It will display a message that asks the user to press the button to take the measurement.
- So, it makes sure it displays real time sensor data dynamically on the webpage every time its accessed.

Purpose of meta refresh in the webpage

- The HTML page has a meta refresh tag
  html += "<meta http-equiv='refresh' content='5'>";
- It automatically refreshes the page after 5 seconds. So it shows the updated temperature and humidity values. The user doesn't have to refresh the browser manually.

Common issues in ESP32 webserver projects and their solutions

- **Failure in Web Connection**: it can occur if SSID or password is incorrect. Make sure the credentials and router is working.

- **Web page not loading**: It can happen if server.handleClient() is not inside the loop. Make sure it is inside the loop so it can handle requests.

- **Web page response is slow**: It can occur if blocking delays are long. Make sure delays are short to keep the server responsive.
- **Invalid sensor readings**: It can occur due to frequently DHT reads. Check readings only from sensor when button is pressed.

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?**

   #define BLYNK_TEMPLATE_ID "TMPL6UCKQ_P6D"
   BLYNK_TEMPLATE_ID tells the ESP32 which cloud template will be use.
   It must match with cloud template so the ESP32 can send data like temperature and humidity to the correct virtual pins and widgets in the **Blynk app.**

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

Blynk Template ID	Blynk Auth Token
It gives the information to ESP32 about what kind of device it is in the Blynk Cloud.	It works like a password for each ESP32 device.
It can be same for many devices.	Auth Token different for each device.
It defines the virtual pins, widgets etc.	It is used to securely connect ESP32 to Blynk Cloud.

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

   The sensor can produce incorrect readings because DHT11 sensor uses different data formats and timing protocols than DHT22.

Errors or incorrect readings can happen when DHT22 tries to read decimal values but DHT11 does not provide it.

Key Difference

DHT11 Sensor	DHT22 Sensor
Its resolution is 1°C temperature and 1 % humidity.	Its resolution is 0.1°C temperature and 0.1% humidity
Data Format is integer.	Data format is decimal.
Slower communication protocol.	Faster or highly precise communication protocol.

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

Virtual Pins in Blynk are the software pins **($V_0 – V_{255}$).** They are used for communication like sending data between **ESP32 and the Blynk Cloud.**

Virtual pins are preferred because they provide flexibility like:

- They are not limited by hardware pin count (GPIOS).
- They can represent any data type (temperature, colors, text).
- Even one pin can handle multiple hardware actions.
- They can handle complex functions easily.

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

BlynkTimer does not stop the program like delay ().
It keeps running things in the usual routine in ESP32 like button input, sensor readings, blynk cloud communication.
Everything works smoothly without any interruption.

Part-B: Long Question

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

ESP32 can send real time values of temperature and humidity values to the BLYNK cloud.
These values are sent using sensors such as DHT22.
The Blynk Mobile App displays these values.

Creation of Blynk Template and Datastreams

In the first step we create a device template in the Blynk Cloud.
Datastreams are created for temperature and humidity.
V0 → Temperature
V1 → Humidity
Widgets are used to show sensor values.
The virtual pins will help cloud understand what kind of data ESP32 sent

Role of Template ID, Template Name, and Auth Token

There are three Blynk parameters that are important:

```
#define BLYNK_TEMPLATE_ID   "TMPL6UCKQ_P6D"
#define BLYNK_TEMPLATE_NAME "dht"
#define BLYNK_AUTH_TOKEN    "...token..."
```

- **Roles:**

  Template ID:
  - It identifies the template in Blynk which ESP32 belongs to.
  - Data will not receive or sync if it does not match with the Blynk cloud template.

  Template Name:
  - It is a label that helps in organizing devices inside the Blynk.

- It is used for identification, does not affect the connection.

Auth Token:

- It is a security password that is used for connecting ESP32 to the cloud.
- Auth token must be unique for every device for security purposes.

Sensor configuration issues (DHT11 vs DHT22)

In the code, we used.

#define DHTTYPE DHT22

It must match the actual sensor type.

If in code, we used DHT22 type and we connected to sensor DHT11. Then there will be errors or readings can be wrong

- **Because:**
  - DHT11 sends data in **integers** and DHT22 sends data in **decimal values.**
  - Both sensors use different data and timing structures.
  - DHT22 provides wider range of temperature and humidity values.

Sending data using Blynk.virtualWrite()

Data is sent to cloud after reading the sensor values using these code lines:

- **Blynk.virtualWrite(V0, t);**
- **Blynk.virtualWrite(V1, h);**

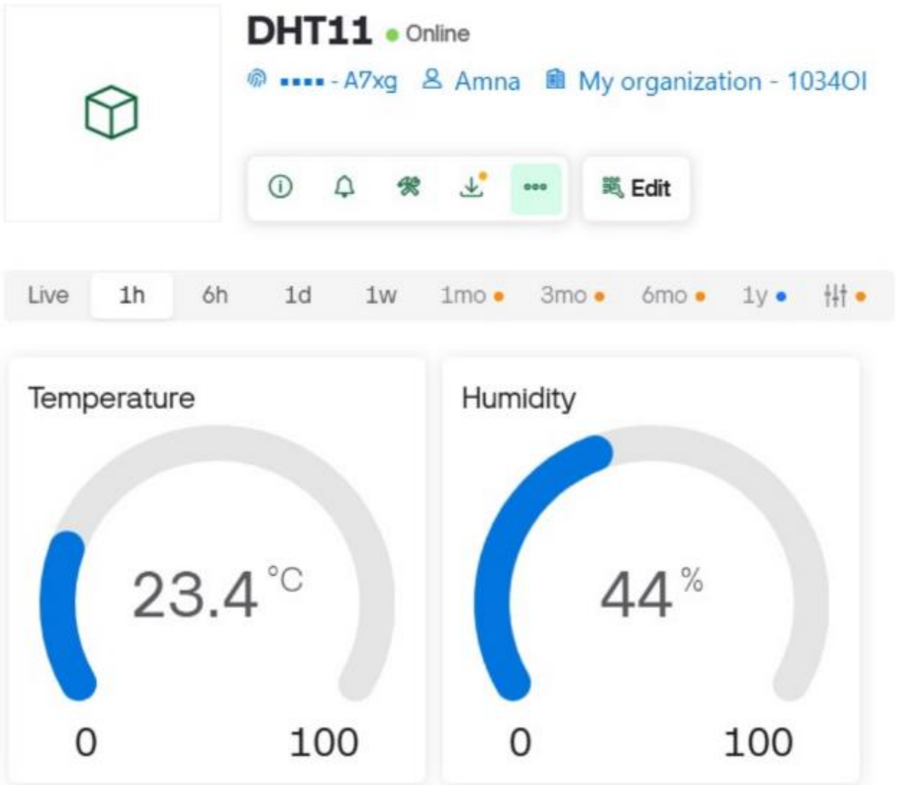
These function transfer sensor values to data streams. When the data reaches to cloud, widgets in Blynk app update the values and show the live graphs according to the values.

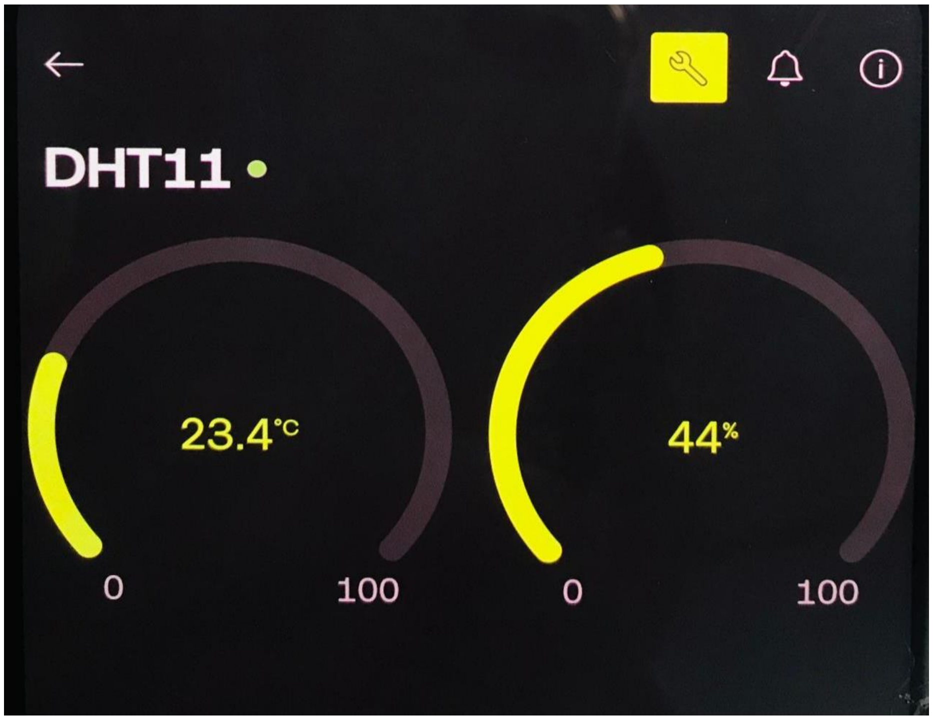
Common problems faced during configuration and their solutions

Problem	**Cause**	**Solution**
Device keeps showing offline.	The reason can be wrong auth token or Wi-Fi details	Check auth token or Wi-Fi credentials
Data is not showing in the app	The reason can be incorrect virtual pins.	Check virtual pins again. For V0 and V1 data streams

OLED keeps showing booting. But not showing sensor values	The reason can be incorrect DHT type	Check DHT type, if using DHT11 sensor then Define DHT11 not 22.
Keep showing old values.	Using Delay () function in loop. It slowly updates the value.	Use BlynkTimer() function despite Delay ().

Screenshots of Blynk Cloud (Web) and Blynk Mobile App Dashboard




DHT11
23.4°C
0
100
44%
0
100