National Textile University, Faisalabad

Department of Computer Science

Name:	Minahil Fatima
Class:	BSCS-5B
Registration No:	23-NTU-CS-1051
Assignment:	Homework-01 – After mid
Course Name:	Embedded IoT Systems
Submitted To:	Sir Nasir Mahmood
Submission Date:	17-December, 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); is used to create a web server object on the ESP32.
     - The object server allows the ESP32 to host a webpage.
     - It responds to incoming HTTP requests from a web browser and sends data to the user.
   - Port 80 is the default port for HTTP communication.
     - When a user enters the IP address of ESP32 in a browser, the request is sent to port 80.
     - The ESP32 sends a webpage that displays the sensor readings.

2. **Explain the role of server.on("/", handle Root); in this program.**
   When a web browser visits a specific URL, it tells the web server what to do.
   - / represents the home page of the web server.
   - handleRoot is a function. It runs whenever someone accesses the root page.
   - handleRoot() function generates the HTML page that shows the temperature and humidity readings from the DHT sensor.

3. **Why is server.handleClient(); placed inside the loop() function? What will happen if it is removed?**
   It is placed inside the loop() function because it continuously checks and responds to requests from clients. As the loop() function runs repeatedly, placing this function inside ensures that whenever a browser requests the webpage the ESP32 always responds. If it is removed from the loop() function, the web server will not respond to any HTTP requests, and the users will not be able to see the web page or sensor data.

4. **In handleRoot(), explain the statement: server.send(200, "text/html", html);**
   It is used to send a response from the ESP32 web server to the client. It sends the web page with sensor data to the user's browser.

- 200 is the HTTP status code. It means the request was successful.
- "text/html" tells the browser that this is an HTML web page.
- html is the actual content being sent. In this program this content is the temperature and humidity readings.

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

Displaying last measured sensor values	Taking a fresh DHT reading
• The program shows the values stored in lastTemp and lastHum from the previous reading.	• The program reads the sensor again by calling readDHTValues() inside handleRoot().
• It does not take a new reading from the DHT sensor.	• Whenever user opens the web page, the data is updated with the latest readings.
• The web page loads faster because it shows existing readings.	• It can be slower because it waits for new readings from the sensor.

Part B: Long Question

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

ESP32 webserver-based temperature and humidity monitoring system is used to measure the temperature and humidity using a DHT sensor. The readings taken by the sensor are displayed on OLED and webpage. Wi-Fi is used to access the webpage. Following steps explain the working of this system:

ESP32 Wi-Fi connection process and IP address assignment:

- ESP32 connects to a Wi-Fi network by using the network name and password through WiFi.begin(ssid, password) in the setup() function.
- ESP32 continuously checks for the connection. Once it is connected, Wi-Fi router assigned an IP address to it.
- IP address is used to access the webpage through the browser. IP address is displayed both on the OLED and serial monitor.

Web server initialization and request handling:

- ESP32 creates a web server object through WebServer server(80); A web server runs on port 80 and handles requests from browsers.

- The root URL / is linked to the function handleRoot() by using server.on("/", handleRoot); It is used to generate HTML page that shows the readings taken by the sensor.
- The web server is started by using server.begin() to accept incoming requests.
- The function server.handleClient() inside loop continuously checks and responds to requests from clients.
- The function handleRoot() is used to generate HTML page that shows the readings taken by the sensor.

Button-based sensor reading and OLED update mechanism:

- A button is connected to ESP32 using INPUT_PULLUP.
- The state of the button is continuously checked in the loop.
- When the user presses the button, temperature and humidity readings are taken through DHT sensor by using dht.readTemperature() and dht.readHumidity().
- The new readings are stored in global variables: lastTemp and lastHum.
- showOnOLED() is used to display latest readings on OLED.

Dynamic HTML webpage generation:

- HTML is created as a string by using handleRoot(). It inserts the latest temperature and humidity readings.
- It shows a message which tells the user to press the button if no valid reading exists.
- server.send(200, "text/html", html) is used to send the webpage with sensor data to the user's browser.

Purpose of meta refresh in the webpage:

- The HTML includes <meta http-equiv= 'refresh' content='5'>.
- It automatically updates the webpage after every 5 seconds. It displays the updated readings, and the user does not have to manually reload the webpage.

Common issues in ESP32 webserver projects and their solutions:

Common issues	Solutions
Wi-Fi not connecting	Ensure that ssid and password are correct.
Webpage not loading	Use server.handleClient() in loop.
Wrong sensor reading	Ensure that wiring is correct.
OLED display not working	Ensure proper initializing.
Webpage shows old readings	Press button or call readDHTValues() in handleRoot() for updated readings.

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?**
    - The Template ID tells the ESP32 which project template in the Blynk cloud it should connect to.
        - Template is used to define the following things: virtual pins, data layout, and widgets.
        - In this project, it ensures that ESP32 sends data to the correct template with matching widgets (e.g., V0 for temperature, and V1 for humidity).
    - If the Template ID in the code and Blynk cloud are not same, the ESP32 won't know where to send data. Temperature and humidity readings will not appear on the app, or the device may fail to connect.

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

Blynk Template ID	Blynk Auth Token
• It identifies the template in the cloud to which the ESP32 device belongs.	• It identifies the specific device in the template and allows it to connect to Blynk securely.
• It is same for all devices using the same template.	• It is different from each device created from the template.
• It is required to match the cloud template for proper mapping of widgets.	• It is required to authorize the device to send/receive data from the cloud.

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

    DHT22 and DHT11 are different sensors. They have different timing, resolution, and data format. If you use DHT22 code with DHT11 sensor, the timing and data reading do not match. This will lead to wrong temperature and humidity readings.
    - DHT11 measures temperature 0-50°C and humidity 20-90% with lower precision.
    - DHT22 measure temperature -40-80°C and humidity 0-100% with higher precision.
    - Key difference: DHT22 has a wider measurement range and higher precision.

4. **What are virtual pins in Blynk? Why are they preferred over GPIO pins for cloud communication?**
   Virtual pins are software-based pins in Blynk. Compared to GPIO pins, virtual pins make cloud communication easier and more flexible. These pins allow the ESP32 to send and receive data from Blynk cloud without using the GPIO pins. Virtual pins work over the internet, so the ESP32 can send data to the Blynk app from anywhere, not just locally. Multiple widgets can read or control the same virtual pin, which is not possible with the single GPIO pin. Example: In this program, V0 is for temperature and V1 for humidity.

5. **What is the purpose of using the Blynk Timer instead of delay() in ESP32 IoT applications?**
   The Blynk Timer is used to perform tasks at regular intervals without blocking the main program, unlike delay(). In this project, Blynk Timer is used to send data to Blynk after every 5 seconds. If delay() is used, the ESP32 would pause all other operations like communication with Blynk, reading the state of the button, and updating the OLED. Blynk Timer allows the program to run smoothly and handle multiple tasks simultaneously.

Part B: Long Question

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

The ESP32 reads temperature and humidity from the DHT sensor, displays the readings on OLED, and sends data to the Blynk using virtual pins. Following steps explain the working of this system:

Creation of Blynk template and data streams:

- In the Blynk cloud, create a template. Give a name to the template.
- Define data streams for temperature and humidity. Set name, virtual pin, data type, units, max value, and default value.
- Widgets are linked to data streams, so the app can display values in gauges or graphs.

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

- Template ID: It identifies the template in the cloud to which the ESP32 device belongs. It ensures that ESP32 communicates with the correct dashboard layout and widgets.
- Template Name: It is a human-readable name for the template. It is useful to maintain multiple templates in your Blynk account.

- Auth Token: It identifies the specific device in the template and allows it to connect to Blynk securely. It authenticates the ESP32, as a result the ESP32 can send and receive data from Blynk cloud safely.

Sensor configuration issues (DHT11 vs DHT22):

- DHT22 and DHT11 are different sensors. They have different timing, resolution, and data format. If you use DHT22 code with DHT11 sensor, the timing and data reading do not match. This will lead to wrong temperature and humidity readings. DHT22 has a wider measurement range and higher precision.
- To avoid reading failures, it is essential to ensure proper wiring and initialization in the code.

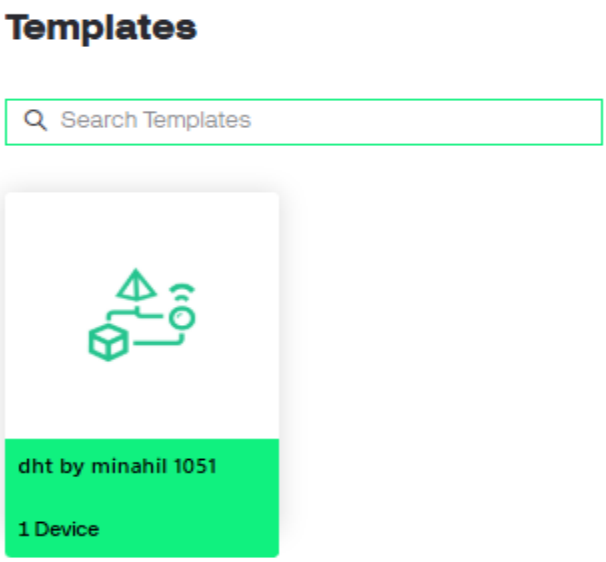
Sending data using Blynk.virtualWrite():

- After reading the sensor data, the data is sent to the corresponding virtual pins in the Blynk Cloud.
- Virtual pins allow flexible mapping to the app widgets. The widgets connected to V0 and V1 update automatically.

Common problems faced during configuration and their solutions:

Common problems	Solutions
Wi-Fi not connecting	Ensure that ssid and password are correct.
Cloud connection issue	Ensure that Auth Token is correct.
Wrong sensor reading	Select correct sensor and check connections.
OLED display not working	Ensure proper initializing.
Code blocking issues	Use Blynk Timer

Screenshots



Virtual Pin Datastream

View mode

You are in a "View" mode. Tap on "Edit" button in the top right corner to make changes.

General | Expose to Automations

NAME: Temperature

ALIAS: Temperature

PIN: V0

DATA TYPE: Double

UNITS: Celsius, °C

MIN | MAX | DECIMALS | DEFAULT VALUE

Close

Virtual Pin Datastream

View mode

You are in a "View" mode. Tap on "Edit" button in the top right corner to make changes.

General | Expose to Automations

NAME: Humidity

ALIAS: Humidity

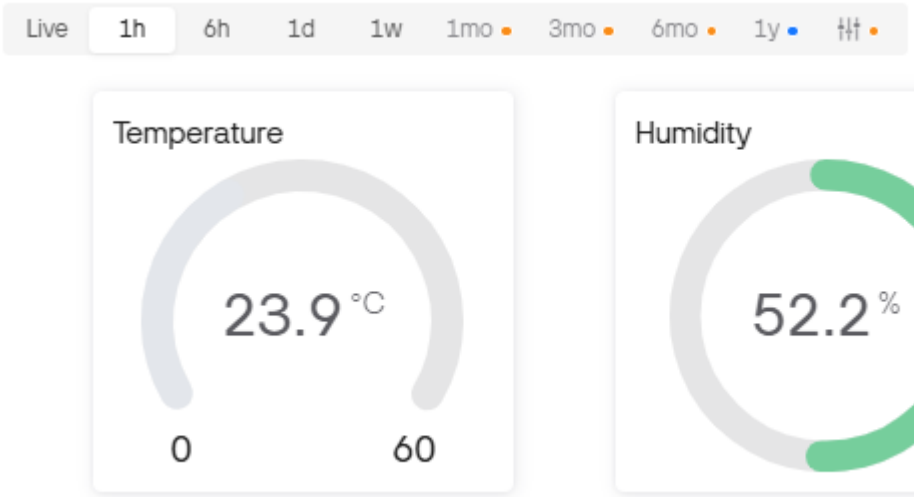
PIN: V1

DATA TYPE: Double

UNITS: Percentage, %

MIN | MAX | DECIMALS | DEFAULT VALUE

Close


Live
1h
6h
1d
1w
1mo
3mo
6mo
1y
Temperature
23.9 °C
0
60
Humidity
52.2 %


dht-esp32
23.5°C
0
60
56.2%
0
100

GitHub Repository Link:

https://github.com/minahilfatima1051/embedded_systems_BSCS-5B_1051