National Textile University, Faisalabad

Name:	Muhammad Zohaib Warraich
Class:	BS-CS(B)
Registration No:	23-NTU-CS-1080
Assignment:	2
Course Name:	IoT and Embedded Systems
Submitted To:	Dr. Nasir Mehmood

IoT and Embedded Systems

Assignment # 2

Question 1

Part A

Short Questions

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

   Ans. The purpose of WebServer server (80) is to create a web server on Esp32 and then allows the ESP32 to respond to HTTP requests coming from a browser.

   Also, the port 80 is a port used for HTTP protocol-based websites. So, when we open ESP32 IP in our browser, it automatically uses port 80 of HTTP protocol.

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

   Ans. When someone opens the root URL which is /, this line guides the web server how to handle it. When any user opens the ESP32 IP in a browser, handleRoot() function is called automatically in the backend. Then the function gets prepared and sends the HTML page to user. In this way, it links the web request to give response to the user.

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

   Ans. The purpose of server.handleClient(); function if to continuously check for incoming web requests in the backend. Also, the loop () runs repeatedly, in this way the server always remains active. And if we remove it, the ESP32 will not be able to respond further to the requests. In short, the webpage will stop loading or may not open.

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

   Ans. When a user makes any request to webserver, this line is used to send a response back to web browser. And 200 is a status code which means the request was successful and “text/html” tells the type of content to the browser that is HTML.

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

Ans. <u>Last measured values:</u> It show the previously stored sensor readings, and they are faster and it avoids the reading from DHT sensor again.

<u>Fresh reading</u>: It means the DHT sensor will be read every time whenever the page loads and, in these ways, we get the updated data, but it can slow the page and stress the sensor.

<u>Part B</u>

<u>Long Questions</u>

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

1. ESP32 Wi-Fi Connection and IP Address Assignment

ESP32 connects to Wi-Fi using the SSID and Password of our network. Once, the ESP32 is connected to the WIFI, our router assigns an available IP address to ESP32 from a range of private IP addresses. In result, this IP is used to access the ESP32 web server from browser.

2. Web Server Initialization and Request Handling

We create a web server on port 80 which is uses HTTP protocol using WebServer library. Then the server listens for browser requests and responds according to that. Moreover, the incoming requests are handled continuously in the loop.

3. Button-Based Sensor Reading and OLED Update

We use a push button that is used to trigger DHT sensor readings manually. When we press the button, fresh values from the sensors that are attached to ESP32 are read. Then these updated values are displayed on the OLED.

4. Dynamic HTML Webpage Generation

Our ESP32 generates an HTML page by using string variables and it displays the latest values that are read from the sensors connected to ESP32. In this way, the webpage is sent to browser as an HTTP response.

5. Purpose of Meta Refresh in Webpage

Meta refresh is used to refresh the web page after every fixed interval of time. This helps in showing the updated values of sensors on the web page without manual refreshing. In this way, it provides a live monitoring effect on web page.

6. Common Issues and Their Solutions

Sometimes WIFI issues occur due to invalid credentials of the network or weak signal. Web page not loading can occur if the request handling is not included in the loop of code. Sensor errors can be fixed by using the correct sensor type and wiring related to that sensor.

Question 2

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. Blynk Template ID is used to link the ESP32 devices to a specific project template that we create in our Blynk Cloud. If it does not match the cloud template, our device will not be able to connect properly. So, we need to ensure the correct data display and device control.

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

   Ans.

   Blynk Template ID: It is used to identify the project template that we create in our cloud. Also, the Template ID is same for all the devices using that template.

   Blynk Auth Token: It is used to uniquely identify and authenticate a specific device. Also, the token is unique for every device and is used for secure connection.

3. **Why does using DHT22 code with a DHT11 sensor produce incorrect readings?**

Ans. The issue is that DHT11 and DHT22 uses different data formats and timings. And DHT22 code expects the higher precision data than DHT11 provides. Therefore, this mismatch results in wrong or unstable readings.

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

   Ans. Virtual pints are software-based pings that don't exist physically and used for sending the data from ESP32 sensors to Blynk Cloud. We prefer these because it has more flexibility for data mapping between device and app widgets which makes communication easier.

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

   Ans. BlynkTimer function is used to run tasks at fixed intervals without stopping the program. On the other side, delay () blocks the CPU and stops all the other executions of microcontroller. If we delay, it can cause hindrance in communication, but BlynkTimer keeps the Wi-Fi and sensors run smoothly.

Part B

Long Questions

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

1. **Creation of Blynk Template and DataStream**

Blynk Template is used to create a template on Blynk cloud to define the structure of our project. In the template, we add data streams for our sensor's readings. These data streams can be virtual pins or physical pins. Widgets are attached to these data streams in Blynk cloud.

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

Template ID connects the ESP32 to the cloud project.
Template Name is used to identify the project in our Blynk cloud.
Auth Token is used to uniquely authenticate the ESP32 device that we want to connect to the Blynk Cloud.

3. **Sensor Configuration Issues (DHT11 vs DHT22)**

DHT11 and DHT22 uses different data formats and accuracy levels for readings values.
If we try to use DHT22 code with DHT11, it will cause hindrances in reading the values.
Therefore, correct type must be defined in the code to get correct value.

4. **Sending Data Using Blynk.virtualWrite()**

Blynk.virtualWrite() sends values of sensor to Blynk cloud template that we connected without ESP32. Temperature and humidity are sent via virtual pins to cloud. These values are displayed on widgets of Blynk app in real time.

5. Common Problems and Their Solutions

If we use wrong Template ID and token, it can cause issue while connecting to Blynk cloud. Using delay () can also cause lead to disconnection of Blynk app, so BlynkTimer is preferred solution. Incorrect wiring and sensor type selection can also lead to sensors errors.

SCREENSHOTS

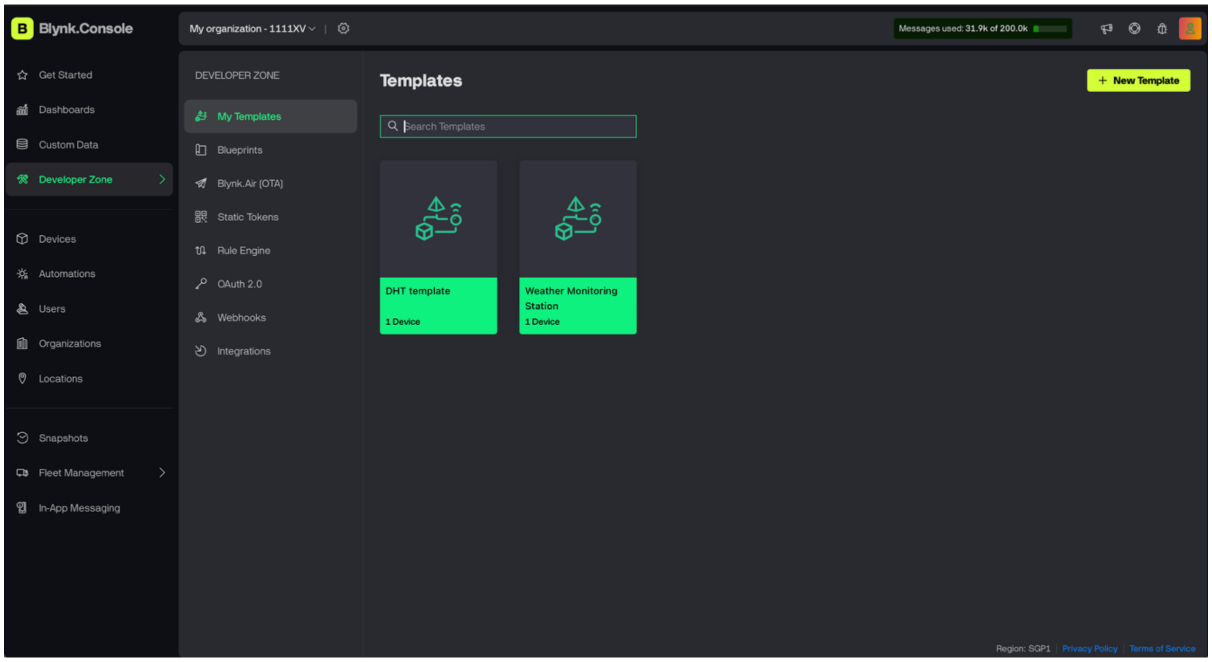
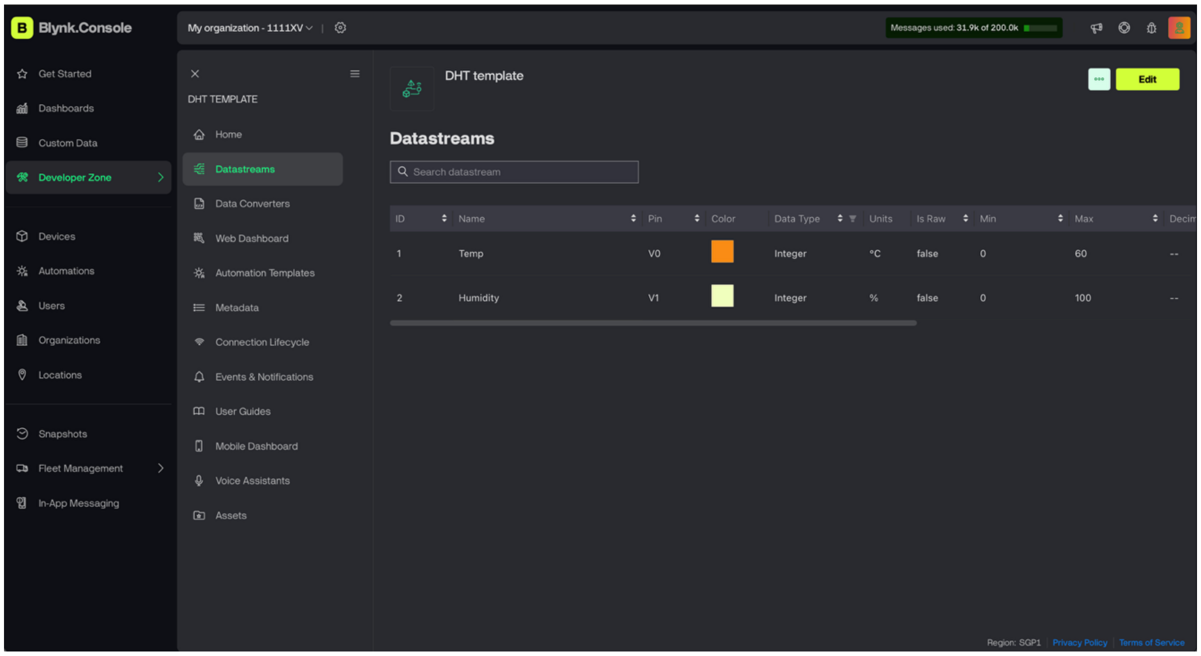


Figure 1 Template



Datastreams

ID	Name	Pin	Color	Data Type	Units	Is Raw	Min	Max	Decim
1	Temp	V0	🟧	Integer	°C	false	0	60	--
2	Humidity	V1	🟨	Integer	%	false	0	100	--

Figure 2 Data streams

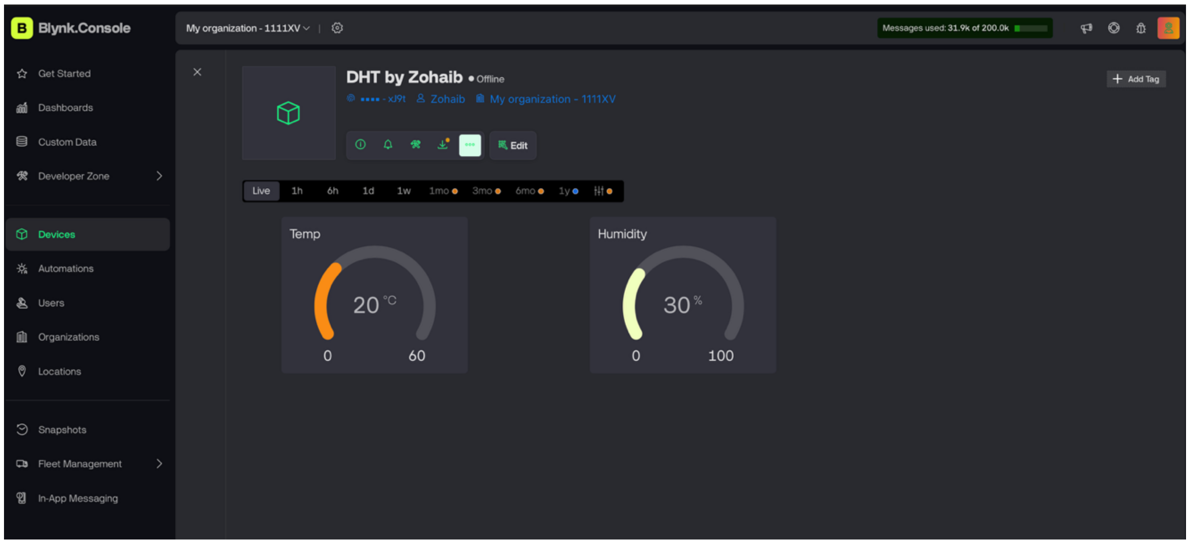


Figure 3 Blynk Cloud Web Dashboard

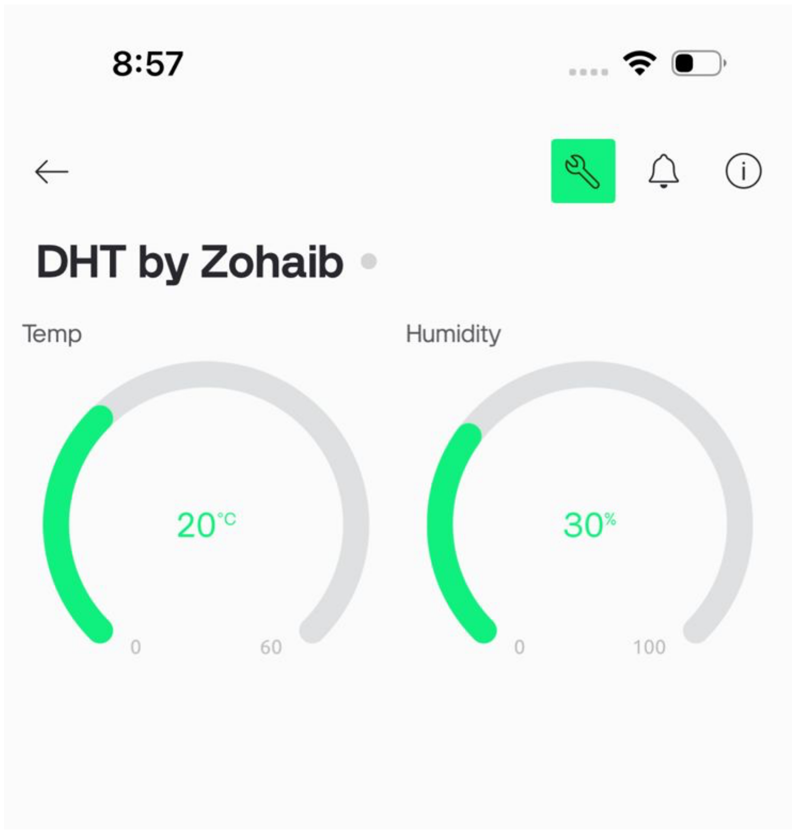


Figure 4 Blynk Cloud App Dashboard