# National Textile University,

# Faisalabad


## Department of Computer Science

| | |
|---|---|
| **Name:** | Adeen Asif |
| **Class:** | BSCS-A |
| **Registration No:** | 23-NTU-CS-1007 |
| **Course Name:** | Embedded IoT and Systems |
| **Submitted To:** | Sir Nasir Mehmood |

# Homework-01 – After mid
## Embedded IOT Systems

## 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);
```

**WebServer server (80)** means we are telling the ESP32 to act like small web site host that listens to the **HTTP** (Hypertext transfer protocol) requests. Port **80** is the default port of HTTP traffic. When we type an IP like 192.168.10.20 it automatically sends request to the port 80.

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

```
server.on("/", handleRoot);
```

This line tells the ESP32 to run the function **handleRoot ()** whenever the user accesses the root page as **'/'**. In simple words, it maps the root URL '/' to code that is responsible for generating the webpage content that is displayed in the browser of the user.

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

```
server.handleClient()
```

**server.handleClient()** constantly keeps checking if someone is trying to connect the ESP32 page. As it must run again and again means repeatedly so, it is placed inside the **loop ().** If it is removed ESP32 will stop responding and becomes silent. Also, webserver becomes unresponsive.

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

```
server.send(200, "text/html", html);
```

In the **handleRoot ()** the statement **server.send(200, "text/html", html)** is responsible for sending the response back to the browser of the user.

- **200** tells the success status.
- **"text/html"** tells the browser about the format and content of the webpage.
- **html** is the actual HTML code that we created

It delivers webpage content to user.

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 fresh DHT reading inside handleRoot ()** |
|---|---|
| • Webpage loads instantly | • Webpage may load slower. |
| • Minimal errors. | • High sensor and occasional errors. |
| • Shows the older values. | • Shows the most recent readings. |
| • More reliable. | • Frequent reads can cause timeout. |
| • Low sensor load as it uses the previous values. | • More sensor load because sensor is polled every time. |

## Part B: Long Question

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

Firstly, the **ESP32** is connected to the WiFi using the **ssid** and **password** of the router **WiFi.begin(ssid, password).** After the connection succeeds, the unique local IP address is assigned by the router like 192.168.20.10. This IP address allows the user to access the webserver. Now, as the WiFi is connected, the webserver is initialized on the port **80**. ESP32 is now capable of responding to the requests that are from any device connected to the same WiFi network. **server.handleClient()** constantly keeps checking if someone is trying to connect the ESP32 page. When someone open the IP of ESP32 in the browser, **handleRoot ()** function runs and generates the webpage content displaying the temperature and humidity. **DHT** (Digital Temperature and Humidity) sensor is used to measure the temperature and humidity. The button when pressed, makes the system to read the values despite of reading them continuously to avoid the sensor load. The button provides the manual control. Also, the values are stored in the variables and **OLED** is updated. These values are also displayed on the web page by using dynamically generated HTML. The meta-refresh tag is used to refresh the page so that the user sees the updated values.

**Problems:**

- Weak WiFi signal.
- Incorrect ssid/ password leading to failed connection
- Sensors wiring issue
- Unresponsiveness of webpage

**Solutions:**

- Make sure to use a strong WiFi network.
- Use correct credentials.
- Test wiring to ensure the proper working of the sensor.
- Check the working of server.handleClient () so the that webpage responses to the requests.

# 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?**

Blynk Template ID links the ESP32's project code to the Blynk cloud template that you have created. It must match the cloud template because if it wouldn't match, the cloud couldn't recognize which dashboard belongs to the device.

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

```
#define BLYNK_TEMPLATE_ID "TMPL6UCKQ_P6D"
#define BLYNK_AUTH_TOKEN    "S2P_gLWBb5E-DINBtsXmlOcndOq6xJnU"
```

**Blynk Template ID:**

- It identifies the project template.
- Template ID is same for all the devices under the same template.

**Blynk Auth token:**

- It identifies the specific device.
- It is unique for all devices.
-

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

```
#define DHTTYPE  DHT22
```

If you are using **DHT11** sensor instead of **DHT22**, it will give false reading as defined sensor has wide range of the temperature and humidity, DHT11 is less accurate. Before the implementation, you must change your sensor type for correct readings.

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

Virtual pins are the software-based channels that are used to send and receive data without using the GPIO pin. Virtual pins are preferred because they let us control and update the values through the cloud without makes changes to the physical circuit. These are flexible and no wiring is needed.

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

**delay ()** blocks the ESP32, stopping the WiFi and Blynk tasks. **BlynkTimer** is non-blocking so communication is smooth, it allows the tasks to run at intervals.

## Part-B: Long Question

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

In **Blynk cloud**, a new template is created by first going into the developer zone and then click on new template. Give name, hardware (ESP32) and connection (WiFi) to the template and then click done. Now, configure the virtual pin **V0** for **temperature** and **V1** for **humidity**. The Template ID, Template Name and Auth Token are generated by the Blynk use them in the ESP32's code. This will help to link the hardware with the cloud, and identify the project template and verify the specific device for authentication with Blynk cloud. On the hardware, the DHT's sensor correct type (**DHT11 Or DHT22**) is configured and connected for accurate readings. Sensor values are firstly read and then send to the cloud using **Blynk.virtualWrite(V0, t)** and **Blynk.virtualWrite(V1, h)**.

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

The BlynkTime is used to read and send the sensor values without blocking the communication.

**Problems:**

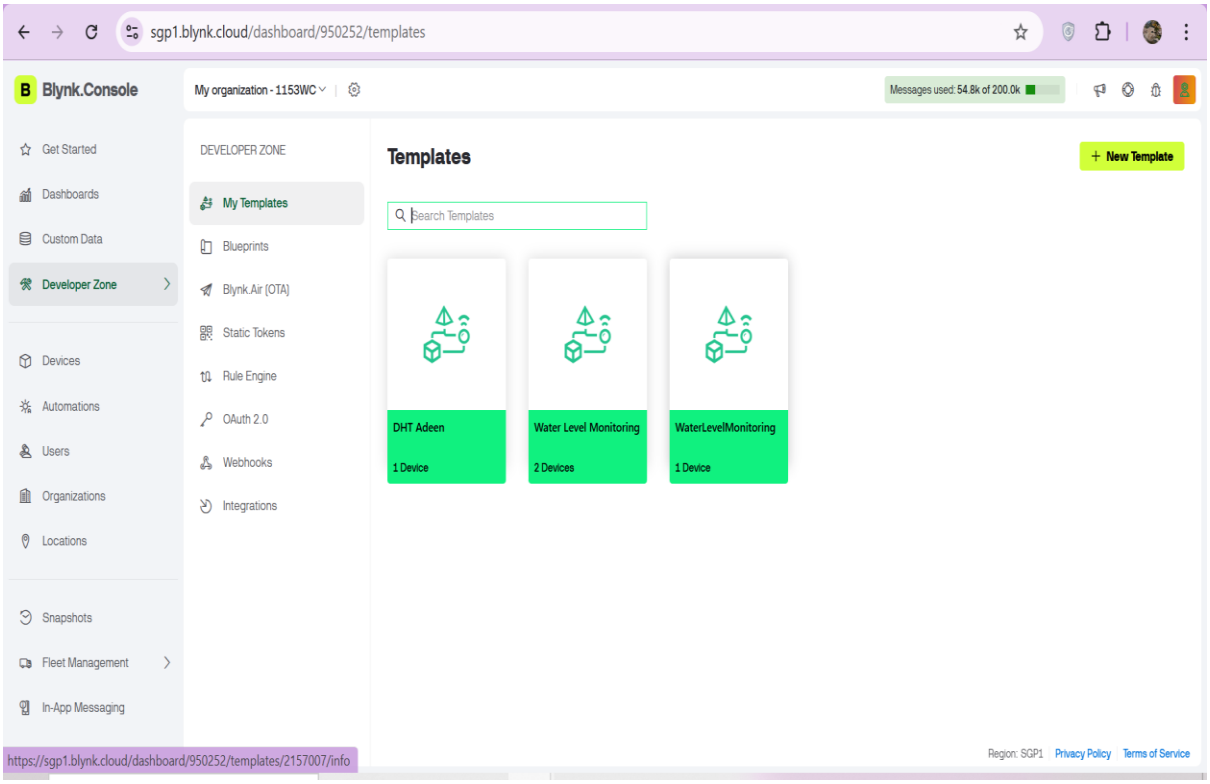
- Invalid **Template ID, Template Name or Auth Token**.
- Incorrect ssid/ password leading to failed connection.
- Wrong data stream pin numbers.
- **DHT** sensor type problem.

**Solution:**

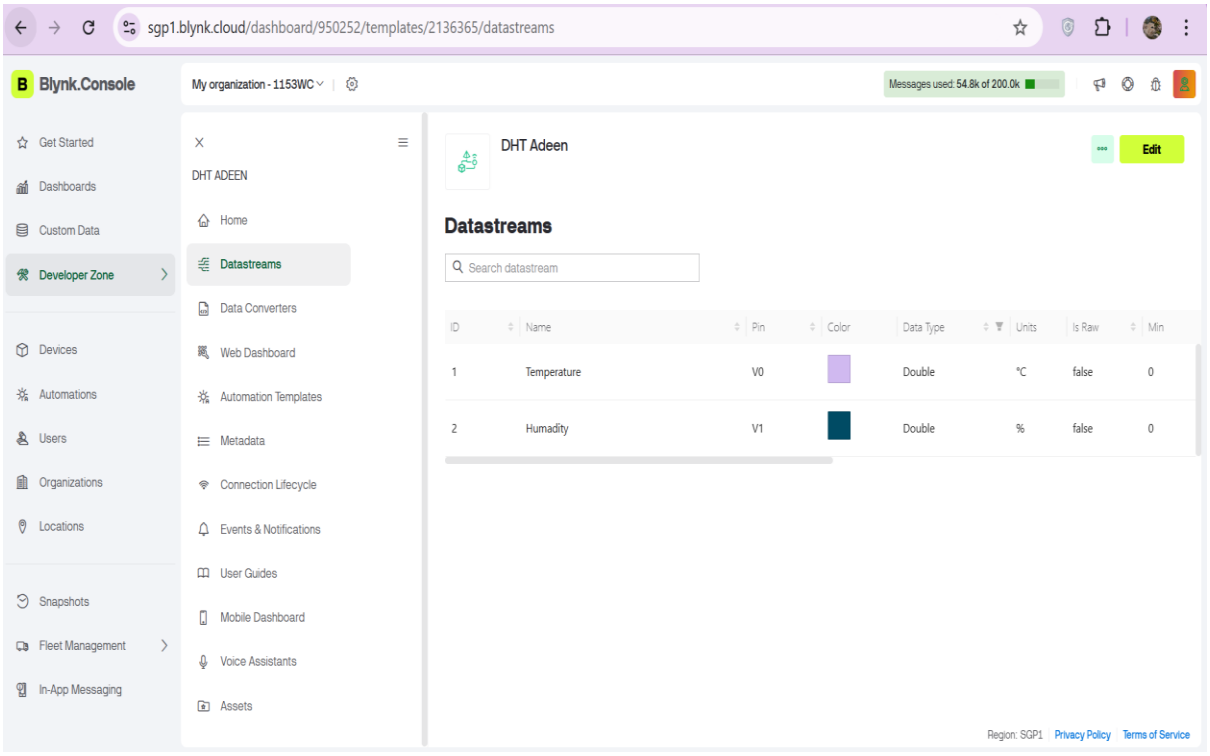
- Copy and paste Template ID, Template Name or Auth Token correctly.
- Use correct credentials.
- Use correct Virtual pins configuration.
- Use correct sensor type **DHT11 or DHT22.**

# Blynk Cloud (web):

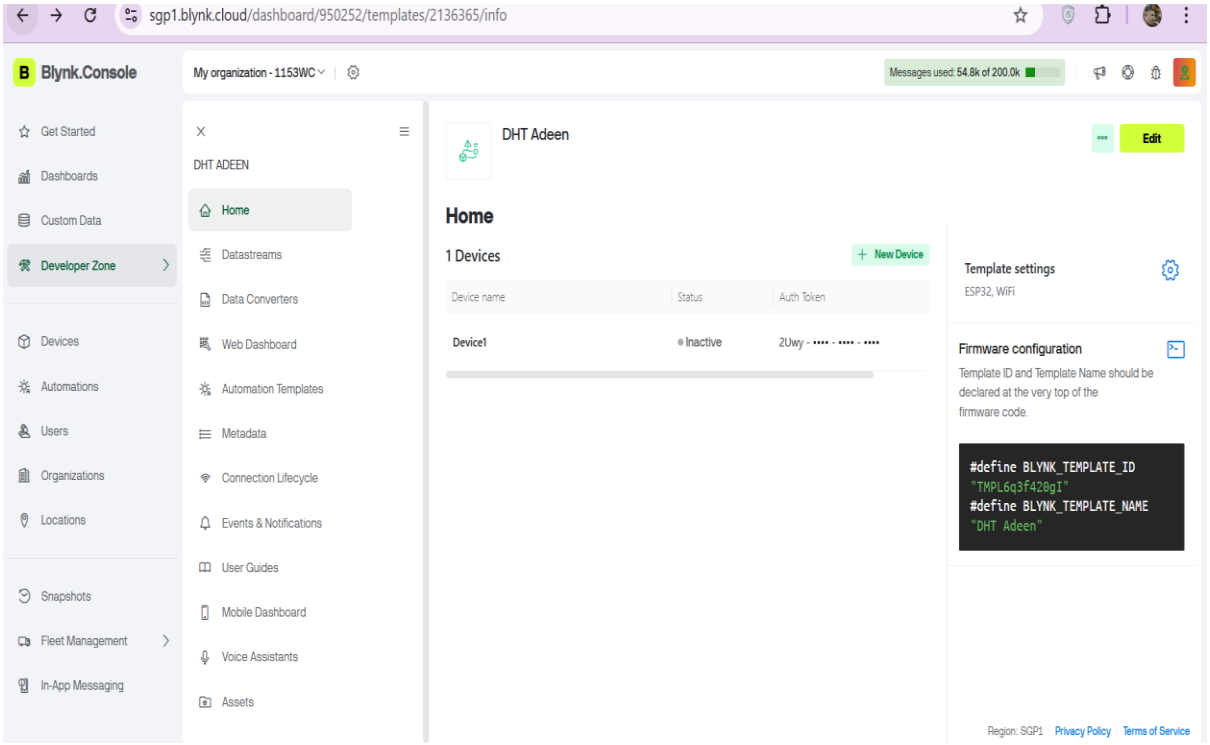
## Template:

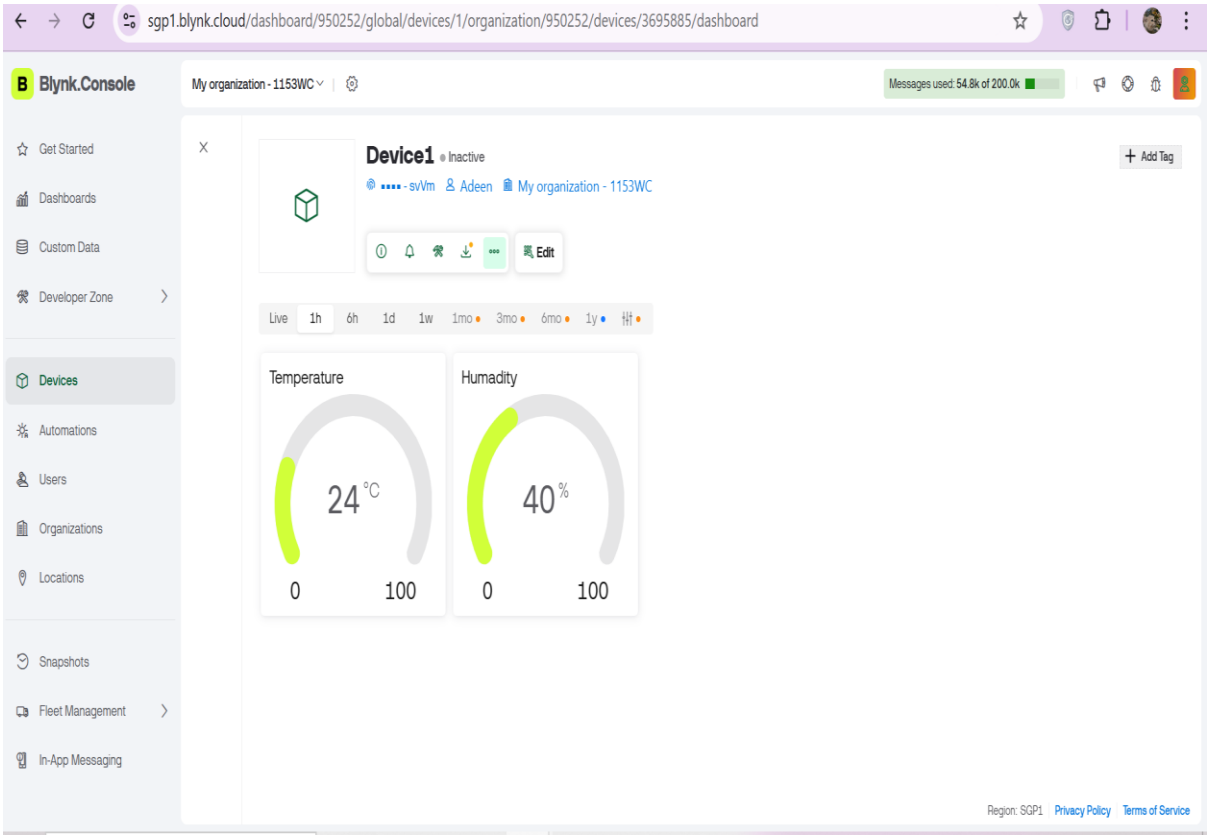



## DataStream:



## Device:


sgp1.blynk.cloud/dashboard/950252/templates/2136365/info
Blynk.Console
My organization - 1153WC
Messages used: 54.8k of 200.0k
Get Started
Dashboards
Custom Data
Developer Zone
Devices
Automations
Users
Organizations
Locations
Snapshots
Fleet Management
In-App Messaging
DHT ADEEN
Home
Datastreams
Data Converters
Web Dashboard
Automation Templates
Metadata
Connection Lifecycle
Events & Notifications
User Guides
Mobile Dashboard
Voice Assistants
Assets
DHT Adeen
Edit
Home
1 Devices
New Device
Device name
Status
Auth Token
Device1
Inactive
2Uwy - •••• - •••• - ••••
Template settings
ESP32, WiFi
Firmware configuration
Template ID and Template Name should be declared at the very top of the firmware code.
#define BLYNK_TEMPLATE_ID
"TMPL6q3f420gI"
#define BLYNK_TEMPLATE_NAME
"DHT Adeen"
Region: SGP1
Privacy Policy
Terms of Service


sgp1.blynk.cloud/dashboard/950252/global/devices/1/organization/950252/devices/3695885/dashboard
Blynk.Console
My organization - 1153WC
Messages used: 54.8k of 200.0k
Get Started
Dashboards
Custom Data
Developer Zone
Devices
Automations
Users
Organizations
Locations
Snapshots
Fleet Management
In-App Messaging
Device1
Inactive
•••• - svVm
Adeen
My organization - 1153WC
Add Tag
Edit
Live
1h
6h
1d
1w
1mo
3mo
6mo
1y
Temperature
24 °C
0
100
Humadity
40 %
0
100
Region: SGP1
Privacy Policy
Terms of Service

## Blynk mobile app: