Name: Laiba Fatima

Reg# 23-ntu-cs-1257

Homework-01 – After mid

Question-1 ESP32 Webserver (webserver.cpp)

Part A:

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

Purpose: WebServer server(80) creates a server object.

Port 80 is the standard communication channel for web traffic (HTTP).

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

It tells the ESP32: "When a user visits the main page (/), run the handleRoot function."

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

It keeps the ESP32 listening for requests. If **removed**, the web page will never load because the ESP32 won't "hear" the browser.

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

It sends three things to the browser: a success code (**200**), the data format (**text/html**), and the actual web page content (**html variable**)

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

Last values are faster to show but might be old. **Fresh reading** gives real-time data but makes the page load slightly slower.

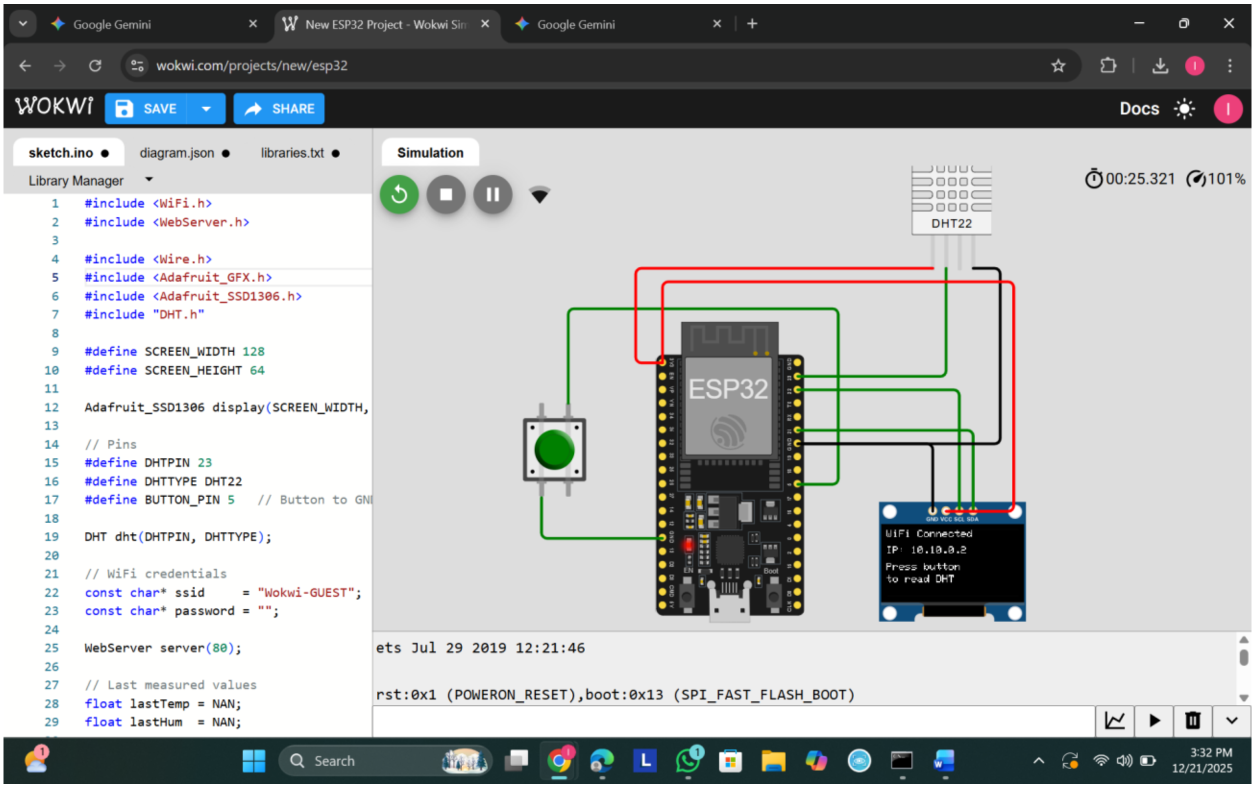
Part B:

Working:

I started this project by setting up the ESP32 to work as a local web server to monitor temperature and humidity. The process begins with the Wi-Fi connection, where the ESP32 uses the WiFi.begin() command to connect to my router; once connected, it is automatically assigned a unique IP address, which I saw on the Serial Monitor and used to access the webpage. After the connection is stable, the web server initialization happens, where the code starts listening for incoming requests from any browser on the same network.

For the hardware part, I used a button-based mechanism to trigger the sensor; whenever I press the button on the ESP32, the code reads the latest data from the DHT22 sensor and immediately updates the OLED display with the current temperature and humidity.

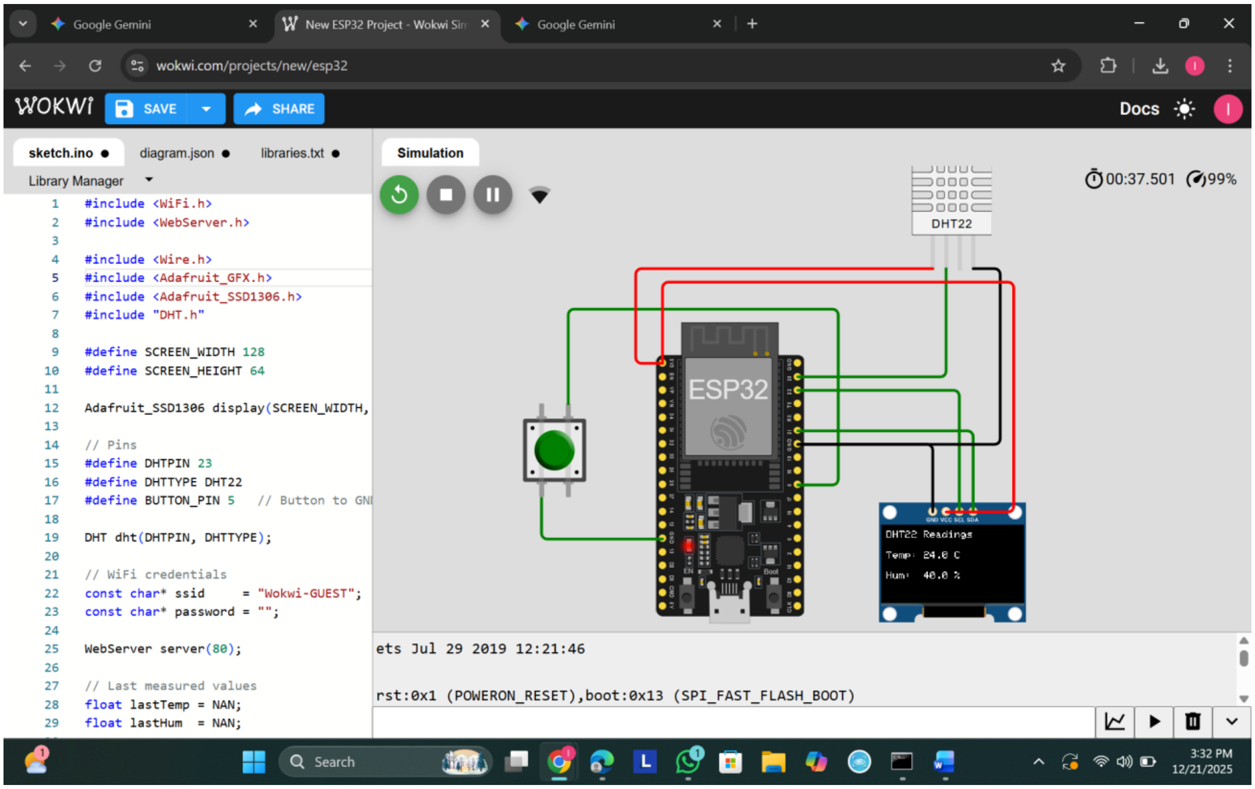
The web server is successfully initialized at IP 10.10.0.2. Due to the virtual private network of the Wokwi simulator, the webpage is being served internally and can be verified via the Serial Monitor and OLED output.".



```
#include <WiFi.h>
#include <WebServer.h>

#include <Wire.h>
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>
#include "DHT.h"

#define SCREEN_WIDTH 128
#define SCREEN_HEIGHT 64

Adafruit_SSD1306 display(SCREEN_WIDTH,

// Pins
#define DHTPIN 23
#define DHTTYPE DHT22
#define BUTTON_PIN 5    // Button to GN

DHT dht(DHTPIN, DHTTYPE);

// WiFi credentials
const char* ssid     = "Wokwi-GUEST";
const char* password = "";

WebServer server(80);

// Last measured values
float lastTemp = NAN;
float lastHum  = NAN;
```

```
WiFi Connected
IP: 10.10.0.2
Press button
to read DHT
```

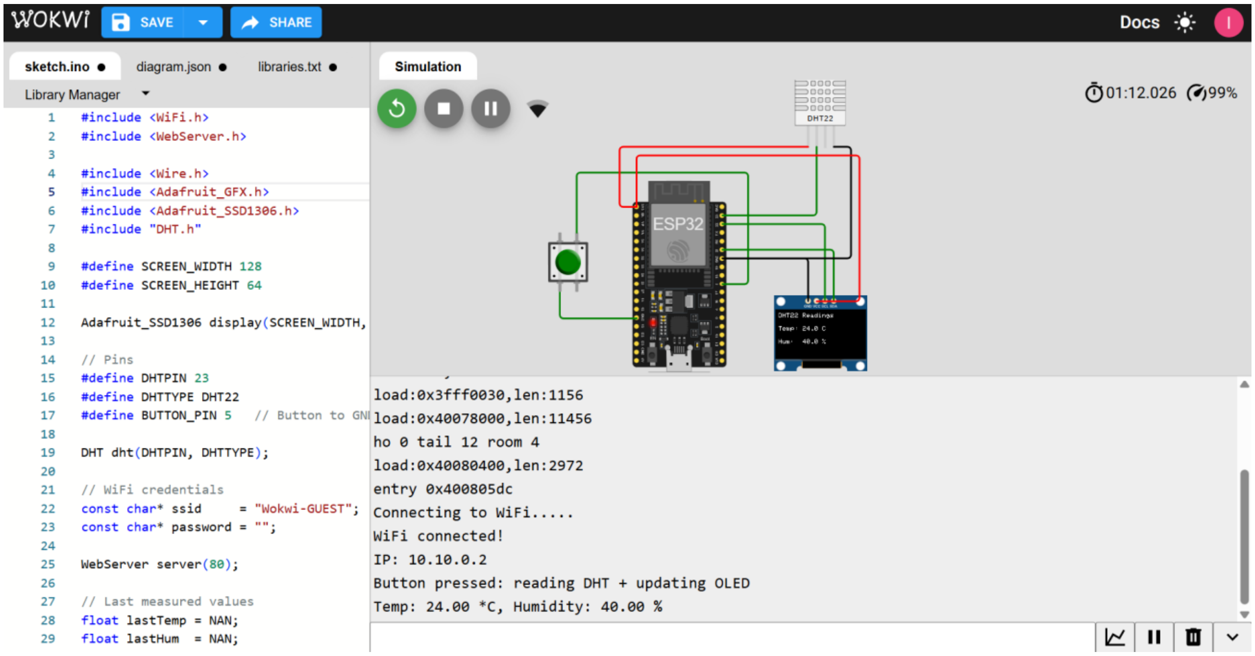
```
ets Jul 29 2019 12:21:46

rst:0x1 (POWERON_RESET),boot:0x13 (SPI_FAST_FLASH_BOOT)
```



```
#include <WiFi.h>
#include <WebServer.h>

#include <Wire.h>
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>
#include "DHT.h"

#define SCREEN_WIDTH 128
#define SCREEN_HEIGHT 64

Adafruit_SSD1306 display(SCREEN_WIDTH,

// Pins
#define DHTPIN 23
#define DHTTYPE DHT22
#define BUTTON_PIN 5    // Button to GN

DHT dht(DHTPIN, DHTTYPE);

// WiFi credentials
const char* ssid     = "Wokwi-GUEST";
const char* password = "";

WebServer server(80);

// Last measured values
float lastTemp = NAN;
float lastHum  = NAN;
```

```
DHT22 Readings
Temp: 24.0 C
Hum:  40.0 %
```

```
ets Jul 29 2019 12:21:46

rst:0x1 (POWERON_RESET),boot:0x13 (SPI_FAST_FLASH_BOOT)
```



```
#include <WiFi.h>
#include <WebServer.h>

#include <Wire.h>
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>
#include "DHT.h"

#define SCREEN_WIDTH 128
#define SCREEN_HEIGHT 64

Adafruit_SSD1306 display(SCREEN_WIDTH,

// Pins
#define DHTPIN 23
#define DHTTYPE DHT22
#define BUTTON_PIN 5   // Button to GN

DHT dht(DHTPIN, DHTTYPE);

// WiFi credentials
const char* ssid     = "Wokwi-GUEST";
const char* password = "";

WebServer server(80);

// Last measured values
float lastTemp = NAN;
float lastHum  = NAN;
```

```
load:0x3fff0030,len:1156
load:0x40078000,len:11456
ho 0 tail 12 room 4
load:0x40080400,len:2972
entry 0x400805dc
Connecting to WiFi.....
WiFi connected!
IP: 10.10.0.2
Button pressed: reading DHT + updating OLED
Temp: 24.00 *C, Humidity: 40.00 %
```

Question 2:

Part A:

What is the role of Blynk Template ID in an ESP32 IoT project? Why must it match the cloud template?

It is a unique identifier that tells the Blynk cloud which project and dashboard the hardware belongs to. It must match so the server knows how to handle the data coming from the ESP32.

Differentiate between Blynk Template ID and Blynk Auth Token.

The Template ID defines the project's structure and hardware type. The Auth Token is a private key that connects a specific device to your account.

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

The readings fail because both sensors use different communication protocols and timings. A key difference is that DHT22 is more accurate and provides decimal values (e.g., 24.5), while DHT11 does not.

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

Virtual Pins (V0, V1, etc.) are digital channels used to send data to the cloud without using physical wires. They are preferred because they don't waste physical GPIO pins and can handle any data type.

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

Delay() freezes the ESP32 and breaks the cloud connection. BlynkTimer is used to run tasks at intervals without stopping the background internet connection.

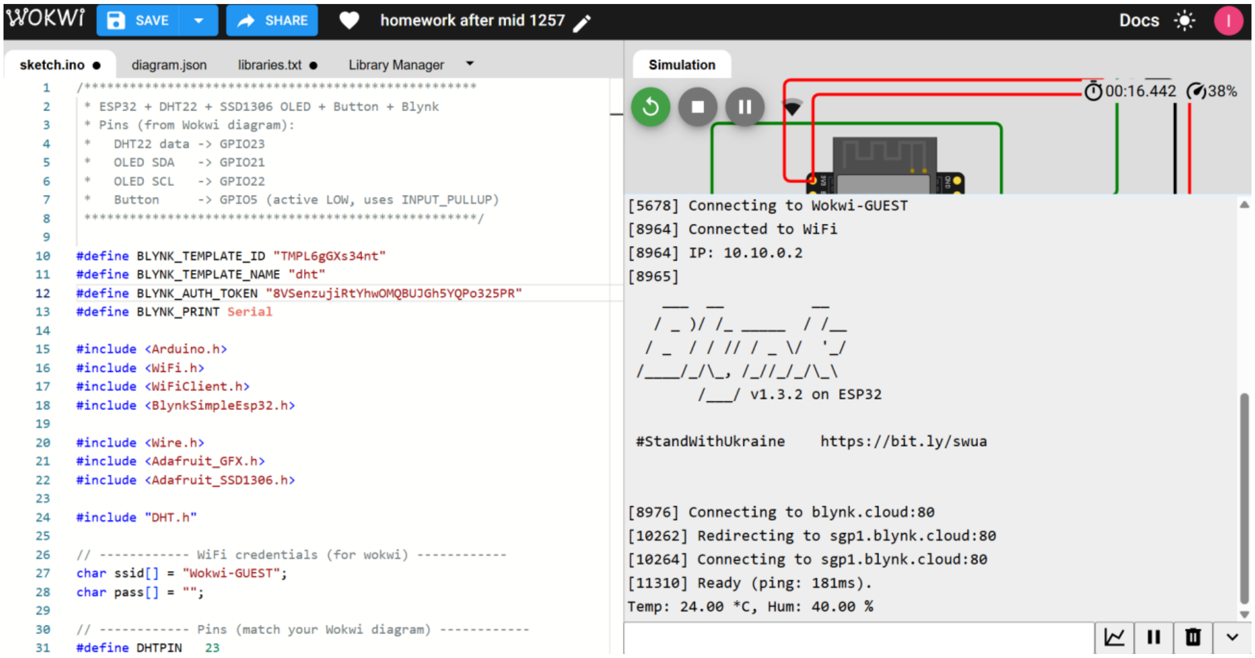
Part B:

I started the project by logging into the Blynk Cloud website to set up my digital dashboard. First, I created a Template named "dht" and set up two Datastreams (V0 for temperature and V1 for humidity) with a range of 0 to 100. Once the template was ready, I created a "New Device" which gave me the Template ID, Template Name, and Auth Token. These three details are the most important because they act as the project's identity and secret password to link my ESP32 hardware to the cloud.

In Wokwi, I wrote the code and selected the DHT22 sensor settings, because using DHT11 code would have caused wrong readings due to different data signals. I pasted my cloud credentials into the code so the ESP32 knew exactly where to send the data. To show the values on my dashboard, I used the Blynk.virtualWrite() command, which sends live sensor data to the virtual pins instead of using physical ones.

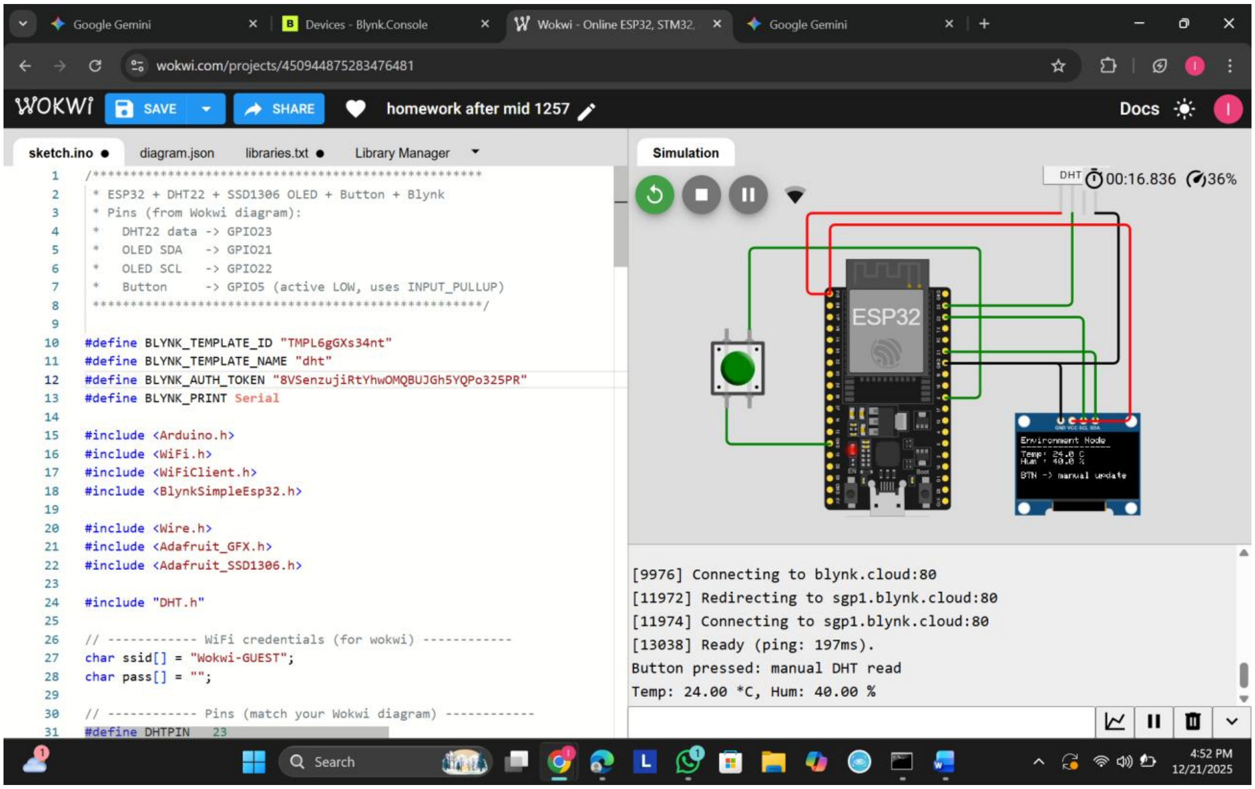
I faced a few common issues, like a fatal error because the Blynk and DHT libraries were missing in Wokwi, so I had to add them to the libraries.txt file. Another problem was the gauges showing only "1" at first, but I fixed this by adjusting the gauge range to 100 and clicking the main Save button on the dashboard. To keep the connection stable, I used BlynkTimer to send data every 5 seconds instead of using delay(), which would have frozen the ESP32 and disconnected it from the internet. Once the simulation started and showed "Ready" in the Serial Monitor, the live data appeared perfectly on both the web dashboard and my mobile app.

I faced some common issues during the process, like a fatal error because the Blynk and DHT libraries were missing in Wokwi, so I had to add them to the libraries.txt file. Another problem was the gauges showing only "1" at first, but I fixed this by changing the gauge range to 100 and hitting the main Save button on the dashboard. To keep the connection stable, I used BlynkTimer to send data every 5 seconds instead of using delay(), which would have frozen the ESP32 and disconnected it from the internet. Once I started the simulation and saw "Ready" in the Serial Monitor, the live temperature and humidity appeared perfectly on the Blynk web dashboard and my mobile app.



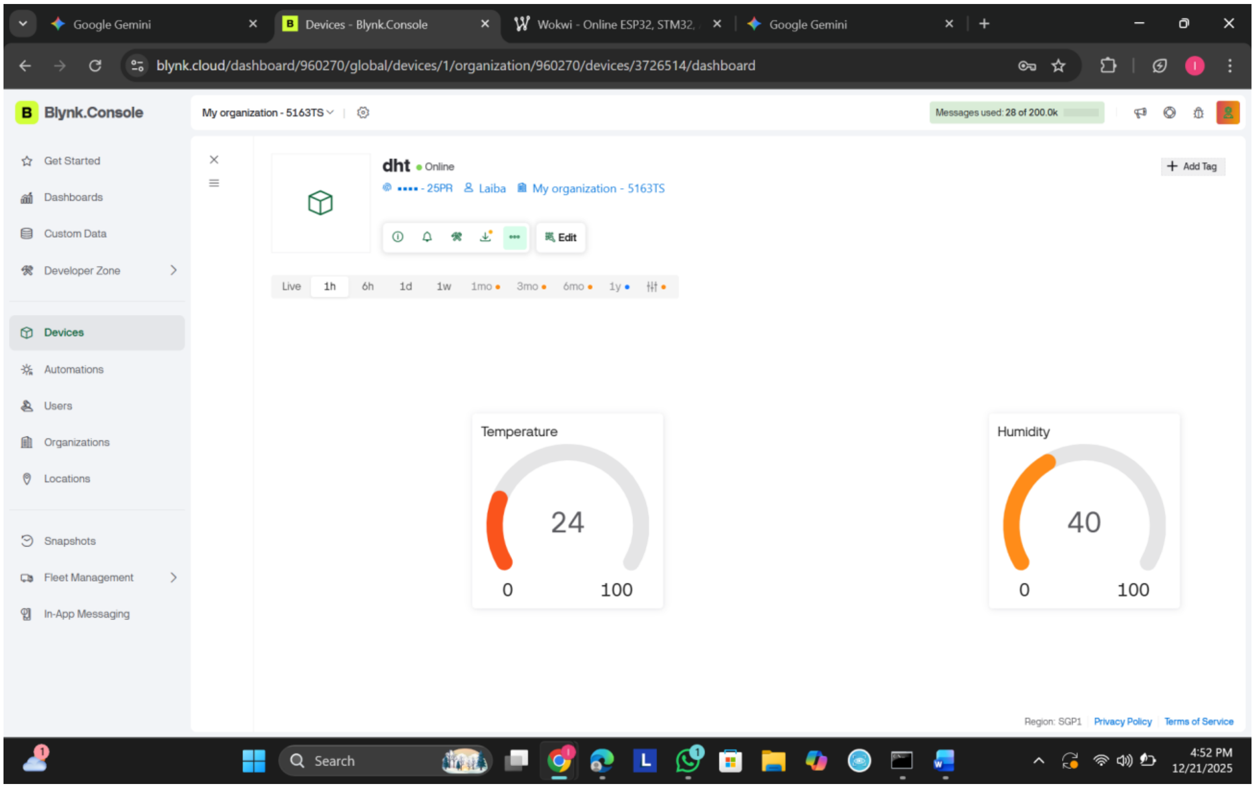
```
/*****************************************************
 * ESP32 + DHT22 + SSD1306 OLED + Button + Blynk
 * Pins (from Wokwi diagram):
 *   DHT22 data -> GPIO23
 *   OLED SDA   -> GPIO21
 *   OLED SCL   -> GPIO22
 *   Button     -> GPIO5 (active LOW, uses INPUT_PULLUP)
 *****************************************************/

#define BLYNK_TEMPLATE_ID "TMPL6gGXs34nt"
#define BLYNK_TEMPLATE_NAME "dht"
#define BLYNK_AUTH_TOKEN "8VSenzujiRtYhwOMQBUJGh5YQPo325PR"
#define BLYNK_PRINT Serial

#include <Arduino.h>
#include <WiFi.h>
#include <WiFiClient.h>
#include <BlynkSimpleEsp32.h>

#include <Wire.h>
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>

#include "DHT.h"

// ------------ WiFi credentials (for wokwi) ------------
char ssid[] = "Wokwi-GUEST";
char pass[] = "";

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

```
[5678] Connecting to Wokwi-GUEST
[8964] Connected to WiFi
[8964] IP: 10.10.0.2
[8965]
    ___  __          __
   / _ )/ /_ _____  / /__
  / _  / / // / _ \/  '_/
 /____/_/\_, /_//_/_/\_\
        /___/ v1.3.2 on ESP32

 #StandWithUkraine    https://bit.ly/swua


[8976] Connecting to blynk.cloud:80
[10262] Redirecting to sgp1.blynk.cloud:80
[10264] Connecting to sgp1.blynk.cloud:80
[11310] Ready (ping: 181ms).
Temp: 24.00 *C, Hum: 40.00 %
```



```
/*****************************************************
 * ESP32 + DHT22 + SSD1306 OLED + Button + Blynk
 * Pins (from Wokwi diagram):
 *   DHT22 data -> GPIO23
 *   OLED SDA   -> GPIO21
 *   OLED SCL   -> GPIO22
 *   Button     -> GPIO5 (active LOW, uses INPUT_PULLUP)
 *****************************************************/

#define BLYNK_TEMPLATE_ID "TMPL6gGXs34nt"
#define BLYNK_TEMPLATE_NAME "dht"
#define BLYNK_AUTH_TOKEN "8VSenzujiRtYhwOMQBUJGh5YQPo325PR"
#define BLYNK_PRINT Serial

#include <Arduino.h>
#include <WiFi.h>
#include <WiFiClient.h>
#include <BlynkSimpleEsp32.h>

#include <Wire.h>
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>

#include "DHT.h"

// ------------ WiFi credentials (for wokwi) ------------
char ssid[] = "Wokwi-GUEST";
char pass[] = "";

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

```
[9976] Connecting to blynk.cloud:80
[11972] Redirecting to sgp1.blynk.cloud:80
[11974] Connecting to sgp1.blynk.cloud:80
[13038] Ready (ping: 197ms).
Button pressed: manual DHT read
Temp: 24.00 *C, Hum: 40.00 %
```



blynk.cloud/dashboard/960270/global/devices/1/organization/960270/devices/3726514/dashboard

dht ● Online

•••• - 25PR Laiba My organization - 5163TS

Temperature

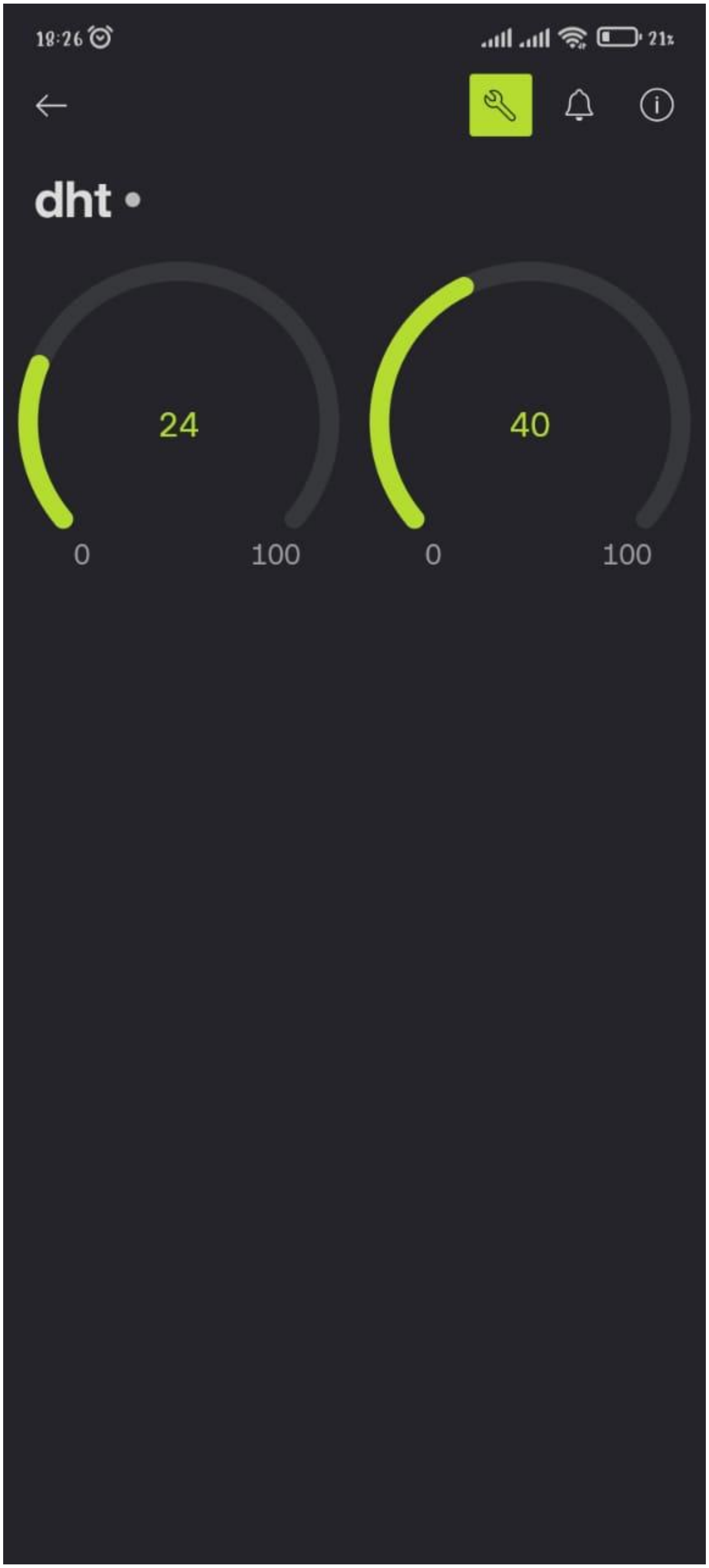
24

0 100

Humidity

40

0 100


18:26
21%
dht •
24
0
100
40
0
100