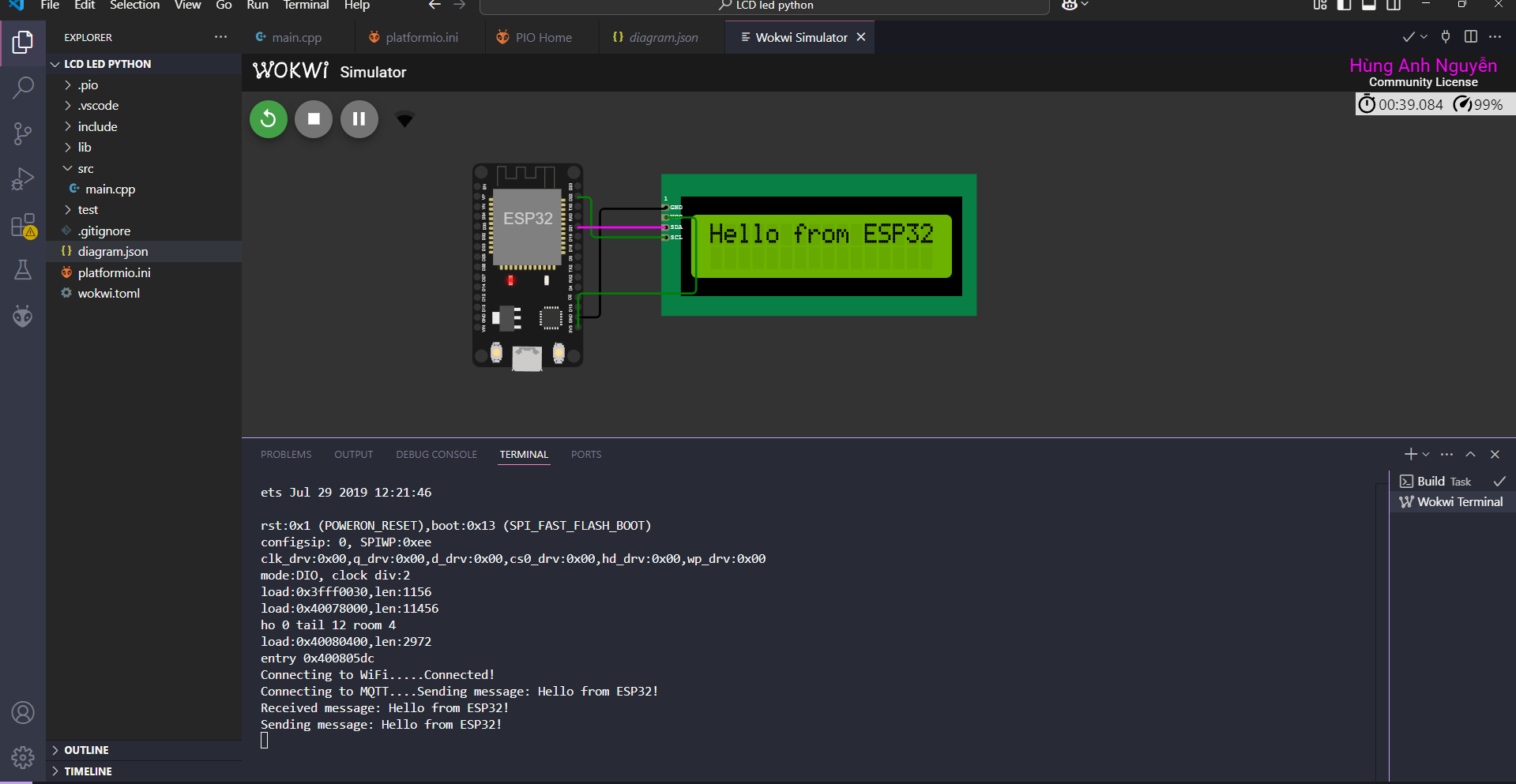
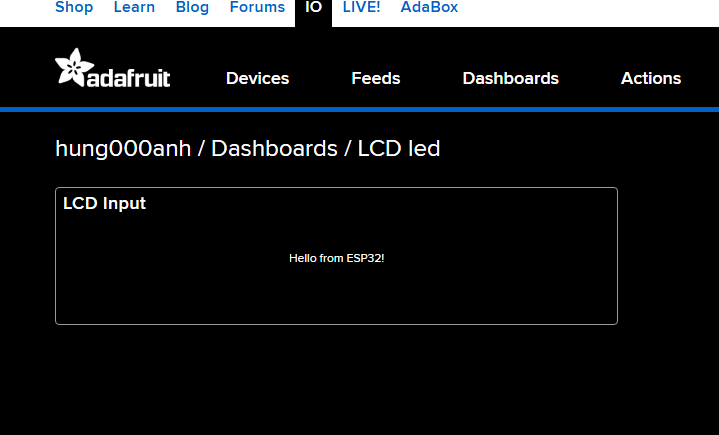
PRACTICE 6: MODERN IoT

1.

LCD Screen Python

Let’s write a Python program to configure ESp32 as MQTT Client to send messages on a LCD to the Adafruit dashboard.





**Code:**

#include <WiFi.h>

#include <Wire.h>

#include <LiquidCrystal\_I2C.h>

#include <Adafruit\_MQTT.h>

#include <Adafruit\_MQTT\_Client.h>

// WiFi & Adafruit IO Credentials

#define WIFI\_SSID "Wokwi-GUEST"

#define WIFI\_PASS ""

#define AIO\_SERVER "io.adafruit.com"

#define AIO\_SERVERPORT 1883 // Cổng MQTT

#define AIO\_USERNAME "hung000anh"

#define AIO\_KEY "aio\_ZjKv66YtWyQQuolAMYmIWicqicem"

// Khởi tạo WiFi Client & MQTT Client

WiFiClient client;

Adafruit\_MQTT\_Client mqtt(&client, AIO\_SERVER, AIO\_SERVERPORT, AIO\_USERNAME, AIO\_KEY);

// Topic MQTT gửi dữ liệu từ ESP32 đến Adafruit IO

Adafruit\_MQTT\_Publish lcdPublish = Adafruit\_MQTT\_Publish(&mqtt, AIO\_USERNAME "/feeds/LCD");

// Topic MQTT nhận dữ liệu từ Adafruit IO

Adafruit\_MQTT\_Subscribe lcdSubscribe = Adafruit\_MQTT\_Subscribe(&mqtt, AIO\_USERNAME "/feeds/LCD");

// Khởi tạo LCD I2C (Địa chỉ: 0x27 hoặc 0x3F)

LiquidCrystal\_I2C lcd(0x27, 16, 2);

void connectMQTT();

void setup() {

Serial.begin(115200);

// Kết nối WiFi

Serial.print("Connecting to WiFi...");

WiFi.begin(WIFI\_SSID, WIFI\_PASS);

while (WiFi.status() != WL\_CONNECTED) {

Serial.print(".");

delay(500);

}

Serial.println("Connected!");

// Kết nối MQTT Server

mqtt.subscribe(&lcdSubscribe);

connectMQTT();

// Khởi động LCD

lcd.init();

lcd.backlight();

lcd.setCursor(0, 0);

lcd.print("MQTT Ready");

}

// Hàm kết nối MQTT

void connectMQTT() {

Serial.print("Connecting to MQTT...");

while (mqtt.connected() == false) {

if (mqtt.connect()) {

Serial.println("Connected!");

} else {

Serial.print(".");

delay(500);

}

}

}

// Hàm gửi tin nhắn lên Adafruit IO

void sendMessage(String message) {

Serial.print("Sending message: ");

Serial.println(message);

lcdPublish.publish(message.c\_str()); // Gửi tin nhắn lên MQTT

}

// Nhận dữ liệu từ Adafruit IO và hiển thị lên LCD

void receiveMessage() {

Adafruit\_MQTT\_Subscribe \*subscription;

while ((subscription = mqtt.readSubscription(5000))) {

if (subscription == &lcdSubscribe) {

String message = (char \*)lcdSubscribe.lastread;

Serial.print("Received message: ");

Serial.println(message);

lcd.clear();

lcd.setCursor(0, 0);

lcd.print(message); // Hiển thị lên LCD

}

}

}

void loop() {

if (!mqtt.connected()) {

connectMQTT();

}

mqtt.processPackets(10000);

// Gửi tin nhắn lên Adafruit IO

sendMessage("Hello from ESP32!");

delay(5000); // Gửi tin nhắn mỗi 5 giây

// Nhận dữ liệu từ Adafruit IO

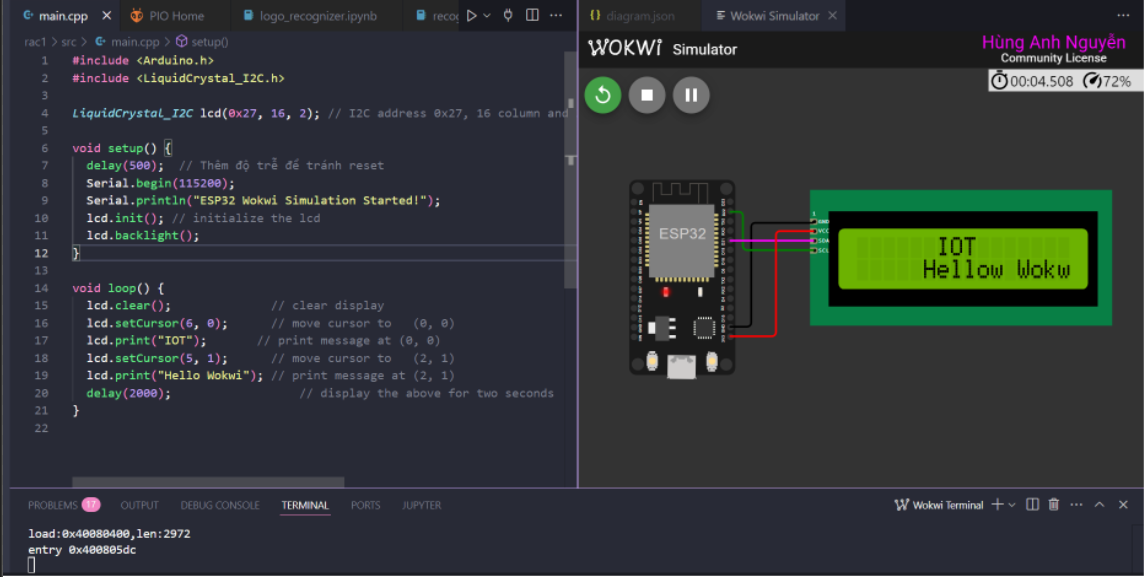
receiveMessage();

}

2 . LCD Screen C/C++

Create a project that displays messages on a LCD as the above program.

Program runs the simulation inside VS Code using C/C++ language.

****

**Code:**

#include <LiquidCrystal\_I2C.h>

LiquidCrystal\_I2C lcd(0x27, 16, 2); // I2C address 0x27, 16 column and 2 rows

void setup() {

lcd.init(); // initialize the lcd

lcd.backlight();

}

void loop() {

lcd.clear(); // clear display

lcd.setCursor(6, 0); // move cursor to (0, 0)

lcd.print("IOT"); // print message at (0, 0)

lcd.setCursor(5, 1); // move cursor to (2, 1)

lcd.print("Hellow Wokwi"); // print message at (2, 1)

delay(2000); // display the above for two seconds

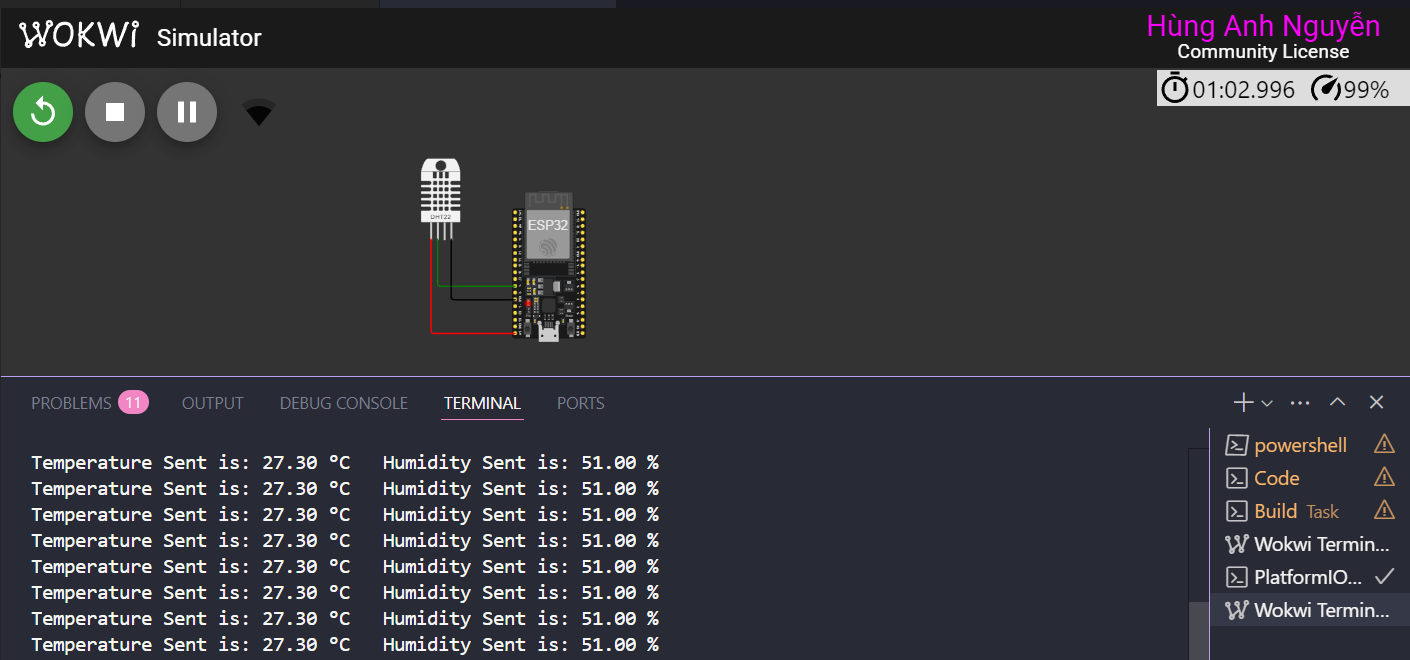
}

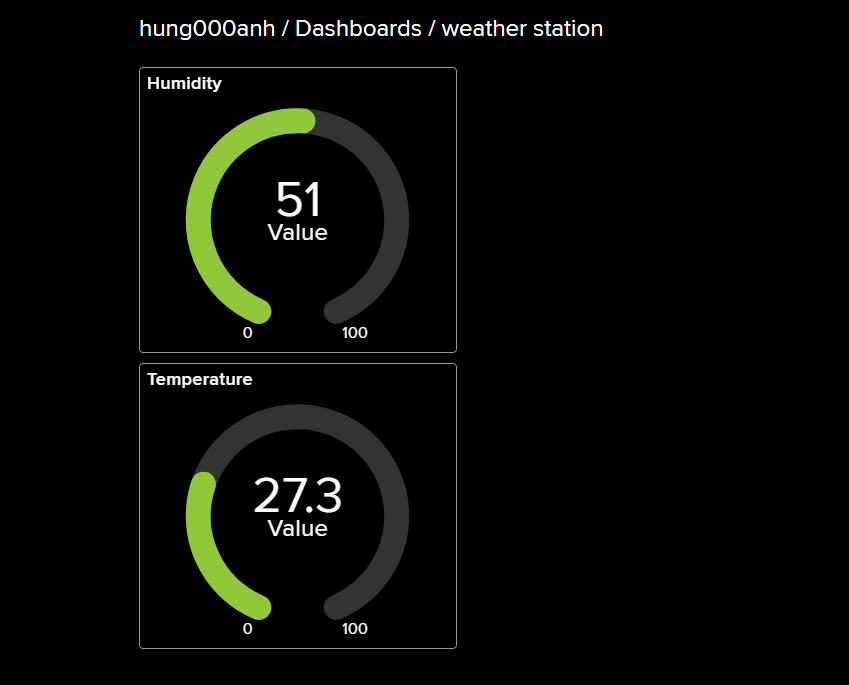
3. Weather Station Python

Develop an IoT Weather Station program, use MQTT communication protocol

with the ESP32 to publish messages and subscribe to topics by Python

language.





**Code**

import time

import json

import paho.mqtt.client as mqtt

import Adafruit\_DHT

import smbus

# WiFi & MQTT Credentials

AIO\_USERNAME = "hung000anh"

AIO\_KEY = "aio\_ZjKv66YtWyQQuolAMYmIWicqicem"

AIO\_SERVER = "io.adafruit.com"

AIO\_PORT = 1883

# MQTT Topics

TEMP\_FEED = f"{AIO\_USERNAME}/feeds/temperature"

HUMIDITY\_FEED = f"{AIO\_USERNAME}/feeds/humidity"

LED\_FEED = f"{AIO\_USERNAME}/feeds/led"

# DHT Sensor Configuration

DHT\_SENSOR = Adafruit\_DHT.DHT22

DHT\_PIN = 4

# LCD Configuration

I2C\_ADDR = 0x27 # Change to 0x3F if needed

bus = smbus.SMBus(1)

# MQTT Client Setup

client = mqtt.Client()

client.username\_pw\_set(AIO\_USERNAME, AIO\_KEY)

def on\_connect(client, userdata, flags, rc):

if rc == 0:

print("Connected to MQTT Broker!")

client.subscribe(LED\_FEED)

else:

print(f"Failed to connect, return code {rc}")

def on\_message(client, userdata, msg):

message = msg.payload.decode("utf-8")

print(f"Received from {msg.topic}: {message}")

if msg.topic == LED\_FEED:

handle\_led\_control(message)

def handle\_led\_control(state):

if state.upper() == "ON":

print("LED Turned ON")

elif state.upper() == "OFF":

print("LED Turned OFF")

def read\_dht\_data():

humidity, temperature = Adafruit\_DHT.read\_retry(DHT\_SENSOR, DHT\_PIN)

if humidity is not None and temperature is not None:

return round(temperature, 2), round(humidity, 2)

return None, None

def send\_weather\_data():

temperature, humidity = read\_dht\_data()

if temperature is None or humidity is None:

print("Failed to read from DHT sensor!")

return

print(f"Temperature: {temperature} °C, Humidity: {humidity} %")

client.publish(TEMP\_FEED, json.dumps(temperature))

client.publish(HUMIDITY\_FEED, json.dumps(humidity))

def main():

client.on\_connect = on\_connect

client.on\_message = on\_message

client.connect(AIO\_SERVER, AIO\_PORT, 60)

client.loop\_start()

while True:

send\_weather\_data()

time.sleep(10)

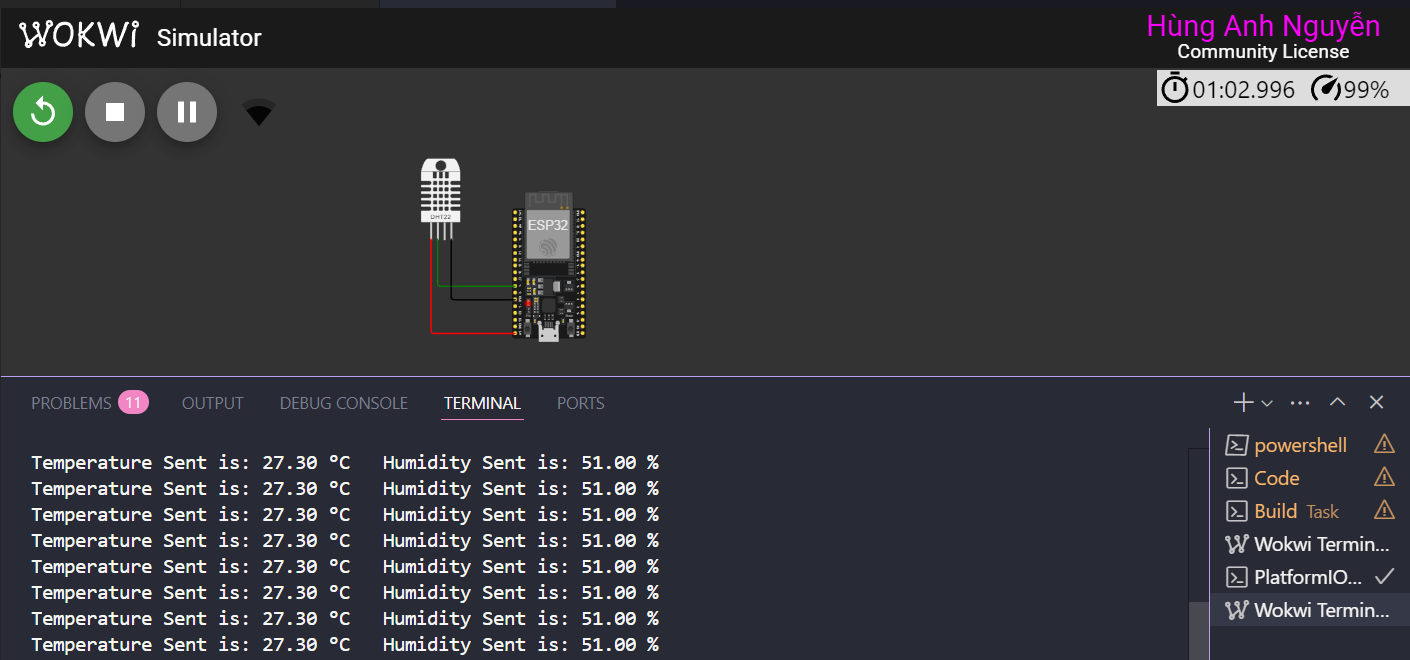
if \_\_name\_\_ == "\_\_main\_\_":

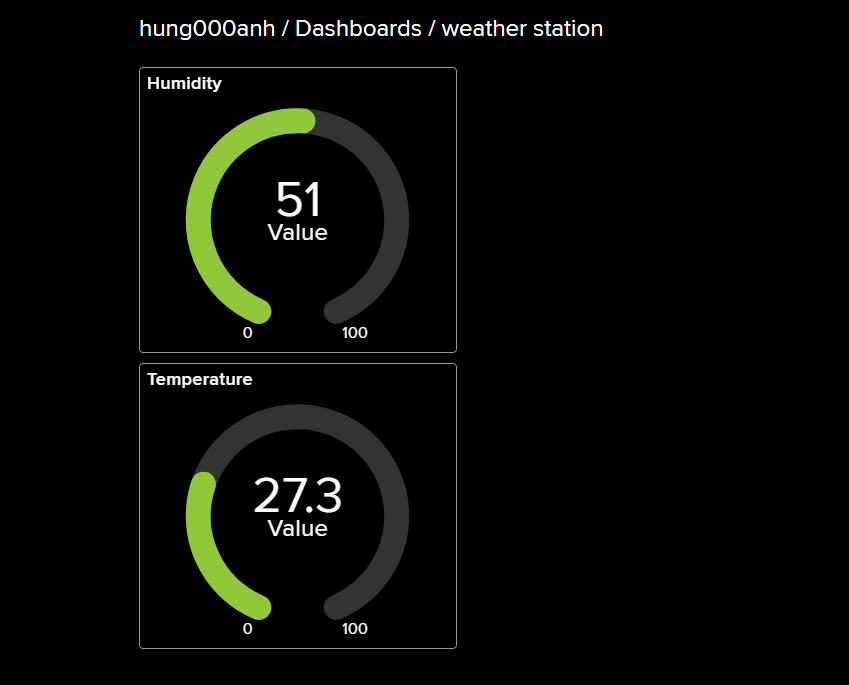
main()

4. Weather Station C/C++

Develop a program to run the simulation inside VS Code for the above Weather

Station program by C/C++ language





**Code**

#include <WiFi.h>

#include <Wire.h>

#include <Adafruit\_Sensor.h>

#include <DHT.h>

#include <LiquidCrystal\_I2C.h>

#include <Adafruit\_MQTT.h>

#include <Adafruit\_MQTT\_Client.h>

#define WIFI\_SSID "Wokwi-GUEST"

#define WIFI\_PASS ""

#define AIO\_SERVER "io.adafruit.com"

#define AIO\_SERVERPORT 1883

#define AIO\_USERNAME "hung000anh"

#define AIO\_KEY "aio\_ZjKv66YtWyQQuolAMYmIWicqicem"

#define DHTPIN 4

#define DHTTYPE DHT22

DHT dht(DHTPIN, DHTTYPE);

LiquidCrystal\_I2C lcd(0x27, 16, 2);

WiFiClient client;

Adafruit\_MQTT\_Client mqtt(&client, AIO\_SERVER, AIO\_SERVERPORT, AIO\_USERNAME, AIO\_KEY);

// 🌤️ MQTT Topics for Publishing Data

Adafruit\_MQTT\_Publish temperatureFeed = Adafruit\_MQTT\_Publish(&mqtt, AIO\_USERNAME "/feeds/temperature");

Adafruit\_MQTT\_Publish humidityFeed = Adafruit\_MQTT\_Publish(&mqtt, AIO\_USERNAME "/feeds/humidity");

// 🔔 MQTT Subscription (For Remote LED Control)

Adafruit\_MQTT\_Subscribe ledControl = Adafruit\_MQTT\_Subscribe(&mqtt, AIO\_USERNAME "/feeds/led");

// ⚡ LED Pin

#define LED\_PIN 5

// 🔗 Function Prototypes

void connectWiFi();

void connectMQTT();

void sendWeatherData();

void checkMQTTSubscriptions();

void setup() {

Serial.begin(115200);

connectWiFi();

mqtt.subscribe(&ledControl);

connectMQTT();

lcd.init();

lcd.backlight();

lcd.setCursor(0, 0);

lcd.print("Weather Station");

dht.begin();

pinMode(LED\_PIN, OUTPUT);

}

// 🔗 Connect to WiFi

void connectWiFi() {

Serial.print("Connecting to WiFi...");

WiFi.begin(WIFI\_SSID, WIFI\_PASS);

while (WiFi.status() != WL\_CONNECTED) {

Serial.print(".");

delay(500);

}

Serial.println("Connected!");

}

void connectMQTT() {

Serial.print("Connecting to MQTT...");

while (!mqtt.connected()) {

if (mqtt.connect()) {

Serial.println("Connected!");

} else {

Serial.print(".");

delay(1000);

}

}

}

void sendWeatherData() {

float temperature = dht.readTemperature();

float humidity = dht.readHumidity();

// Check if readings are valid

if (isnan(temperature) || isnan(humidity)) {

Serial.println("Failed to read from DHT sensor!");

return;

}

// Display on Serial Monitor

Serial.print("Temperature: "); Serial.print(temperature); Serial.println(" °C");

Serial.print("Humidity: "); Serial.print(humidity); Serial.println(" %");

// Display on LCD

lcd.clear();

lcd.setCursor(0, 0);

lcd.print("Temp: "); lcd.print(temperature); lcd.print("C");

lcd.setCursor(0, 1);

lcd.print("Humidity: "); lcd.print(humidity); lcd.print("%");

// Publish to MQTT

if (temperatureFeed.publish(temperature)) {

Serial.println("Temperature published!");

} else {

Serial.println("Temperature publish failed!");

}

if (humidityFeed.publish(humidity)) {

Serial.println("Humidity published!");

} else {

Serial.println("Humidity publish failed!");

}

}

// 📩 Check MQTT Subscriptions (LED Control)

void checkMQTTSubscriptions() {

Adafruit\_MQTT\_Subscribe \*subscription;

while ((subscription = mqtt.readSubscription(1000))) {

if (subscription == &ledControl) {

String message = (char \*)ledControl.lastread;

Serial.print("LED Control Received: ");

Serial.println(message);

if (message == "ON") {

digitalWrite(LED\_PIN, HIGH);

} else if (message == "OFF") {

digitalWrite(LED\_PIN, LOW);

}

}

}

}

void loop() {

if (!mqtt.connected()) {

connectMQTT();

}

mqtt.processPackets(10000);

mqtt.ping();

// 📤 Send weather data every 10 seconds

sendWeatherData();

delay(10000);

// 🔔 Check for MQTT messages

checkMQTTSubscriptions();

}

5. Light system

Design a light system that includes 5 buttons and 5 LEDs corresponding with

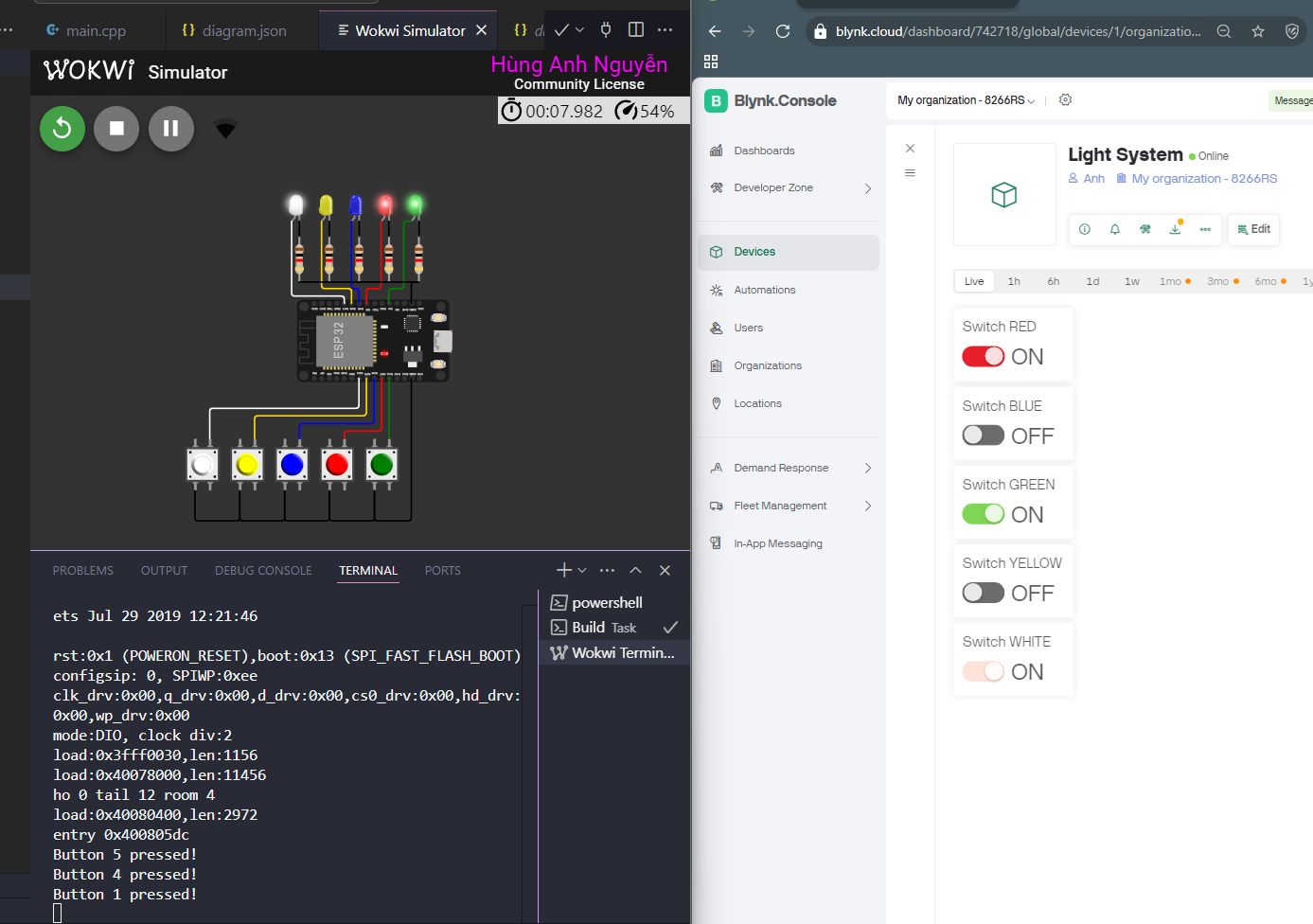
different colors. The system can communicate with the Blynk cloud. If lights are

on/off, then the dashboard of Blynk also turns on/off lights the same. In addition,

information about the number of LEDs that are turned on should be shown off.

Using MQTT communication protocol.

a) Implement in C/C++ language.



Code:

#define BLYNK\_TEMPLATE\_ID "TMPL6LEjHABTS"

#define BLYNK\_TEMPLATE\_NAME "Light System"

#define BLYNK\_AUTH\_TOKEN "P59keUFAjljCWu5O2Qmx4gksGWF77xAu"

#include <WiFi.h>

#include <BlynkSimpleEsp32.h>

char ssid[] = "Wokwi-GUEST";

char pass[] = "";

#define NUM\_LEDS 5

#define NUM\_BTNS 5

const int ledPins[NUM\_LEDS] = {21, 19, 18, 5, 4}; // Red, Blue, Green, Yellow, White

const int btnPins[NUM\_BTNS] = {33, 25, 26, 27, 14}; // Chân nút nhấn

const int blynkPins[NUM\_LEDS] = {V4, V3, V1, V0, V2}; // Chân ảo trên Blynk

bool ledStates[NUM\_LEDS] = {0}; // Trạng thái LED

bool btnStates[NUM\_BTNS] = {1}; // Lưu trạng thái trước đó của nút

BlynkTimer timer;

// Hàm kiểm tra nút bấm

void checkButtons() {

for (int i = 0; i < NUM\_BTNS; i++) {

bool currentState = digitalRead(btnPins[i]);

if (btnStates[i] == HIGH && currentState == LOW) { // Phát hiện nhấn

ledStates[i] = !ledStates[i]; // Đảo trạng thái LED

digitalWrite(ledPins[i], ledStates[i]);

Blynk.virtualWrite(blynkPins[i], ledStates[i]); // Gửi trạng thái lên Blynk

Serial.print("Button ");

Serial.print(i + 1);

Serial.println(" pressed!");

}

btnStates[i] = currentState;

}

}

// Điều khiển LED từ Blynk

BLYNK\_WRITE(V0) { ledStates[0] = param.asInt(); digitalWrite(ledPins[0], ledStates[0]); }

BLYNK\_WRITE(V1) { ledStates[1] = param.asInt(); digitalWrite(ledPins[1], ledStates[1]); }

BLYNK\_WRITE(V2) { ledStates[2] = param.asInt(); digitalWrite(ledPins[2], ledStates[2]); }

BLYNK\_WRITE(V3) { ledStates[3] = param.asInt(); digitalWrite(ledPins[3], ledStates[3]); }

BLYNK\_WRITE(V4) { ledStates[4] = param.asInt(); digitalWrite(ledPins[4], ledStates[4]); }

void setup() {

Serial.begin(115200);

WiFi.begin(ssid, pass);

Blynk.begin(BLYNK\_AUTH\_TOKEN, ssid, pass);

for (int i = 0; i < NUM\_LEDS; i++) {

pinMode(ledPins[i], OUTPUT);

digitalWrite(ledPins[i], LOW);

}

for (int i = 0; i < NUM\_BTNS; i++) {

pinMode(btnPins[i], INPUT\_PULLUP);

}

timer.setInterval(50L, checkButtons);

}

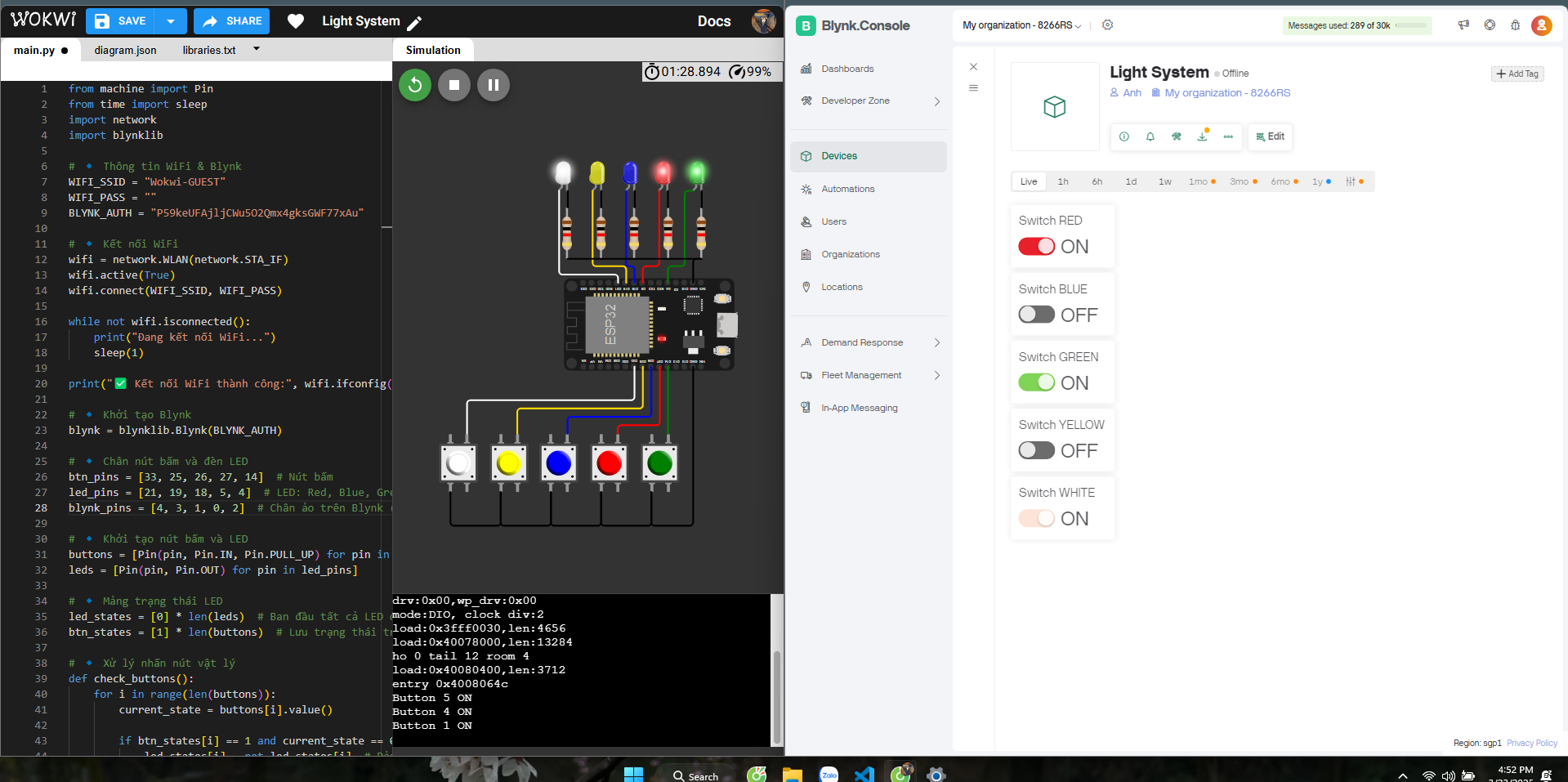
void loop() {

Blynk.run();

timer.run();

}

b) Implement in Python language.

Code:

from machine import Pin

from time import sleep

import network

import blynklib

# 🔹 Thông tin WiFi & Blynk

WIFI\_SSID = "Wokwi-GUEST"

WIFI\_PASS = ""

BLYNK\_AUTH = "P59keUFAjljCWu5O2Qmx4gksGWF77xAu"

# 🔹 Kết nối WiFi

wifi = network.WLAN(network.STA\_IF)

wifi.active(True)

wifi.connect(WIFI\_SSID, WIFI\_PASS)

while not wifi.isconnected():

print("Đang kết nối WiFi...")

sleep(1)

print("✅ Kết nối WiFi thành công:", wifi.ifconfig())

# 🔹 Khởi tạo Blynk

blynk = blynklib.Blynk(BLYNK\_AUTH)

# 🔹 Chân nút bấm và đèn LED

btn\_pins = [33, 25, 26, 27, 14] # Nút bấm

led\_pins = [21, 19, 18, 5, 4] # LED: Red, Blue, Green, Yellow, White

blynk\_pins = [4, 3, 1, 0, 2] # Chân ảo trên Blynk (V4 -> V0)

# 🔹 Khởi tạo nút bấm và LED

buttons = [Pin(pin, Pin.IN, Pin.PULL\_UP) for pin in btn\_pins]

leds = [Pin(pin, Pin.OUT) for pin in led\_pins]

# 🔹 Mảng trạng thái LED

led\_states = [0] \* len(leds) # Ban đầu tất cả LED đều tắt

btn\_states = [1] \* len(buttons) # Lưu trạng thái trước đó của nút

# 🔹 Xử lý nhấn nút vật lý

def check\_buttons():

for i in range(len(buttons)):

current\_state = buttons[i].value()

if btn\_states[i] == 1 and current\_state == 0: # Phát hiện nhấn

led\_states[i] = not led\_states[i] # Đảo trạng thái LED

leds[i].value(led\_states[i]) # Cập nhật trạng thái LED

blynk.virtual\_write(blynk\_pins[i], led\_states[i]) # Gửi lên Blynk

print(f"Button {i+1} {'ON' if led\_states[i] else 'OFF'}")

btn\_states[i] = current\_state # Cập nhật trạng thái nút

# 🔹 Xử lý khi điều khiển từ Blynk

@blynk.on("V0")

def v0\_handler(value):

led\_states[4] = int(value[0])

leds[4].value(led\_states[4])

@blynk.on("V1")

def v1\_handler(value):

led\_states[3] = int(value[0])

leds[3].value(led\_states[3])

@blynk.on("V2")

def v2\_handler(value):

led\_states[2] = int(value[0])

leds[2].value(led\_states[2])

@blynk.on("V3")

def v3\_handler(value):

led\_states[1] = int(value[0])

leds[1].value(led\_states[1])

@blynk.on("V4")

def v4\_handler(value):

led\_states[0] = int(value[0])

leds[0].value(led\_states[0])

# 🔹 Loop chính

while True:

check\_buttons() # Kiểm tra nút nhấn

blynk.run() # Chạy Blynk

sleep(0.1) # Giảm tải CPU