National Textile University, Faisalabad

Department of Computer Science

Name:	Muhammad Talha Javed
Class:	BS Artificial Intelligence
Reg No:	22-NTU-CS-1366
Activity:	Lab 13 Report
Course Code:	**AIE-3079**
Course Name:	**Internet of Things Fundamentals**
Submitted To:	**Nasir Mahmood**
Submission Date:	20-May-2025

Lab 13 Tasks

Run the Arduino-based code to publish DHT sensor data to the Mosquitto MQTT broker.

Code:

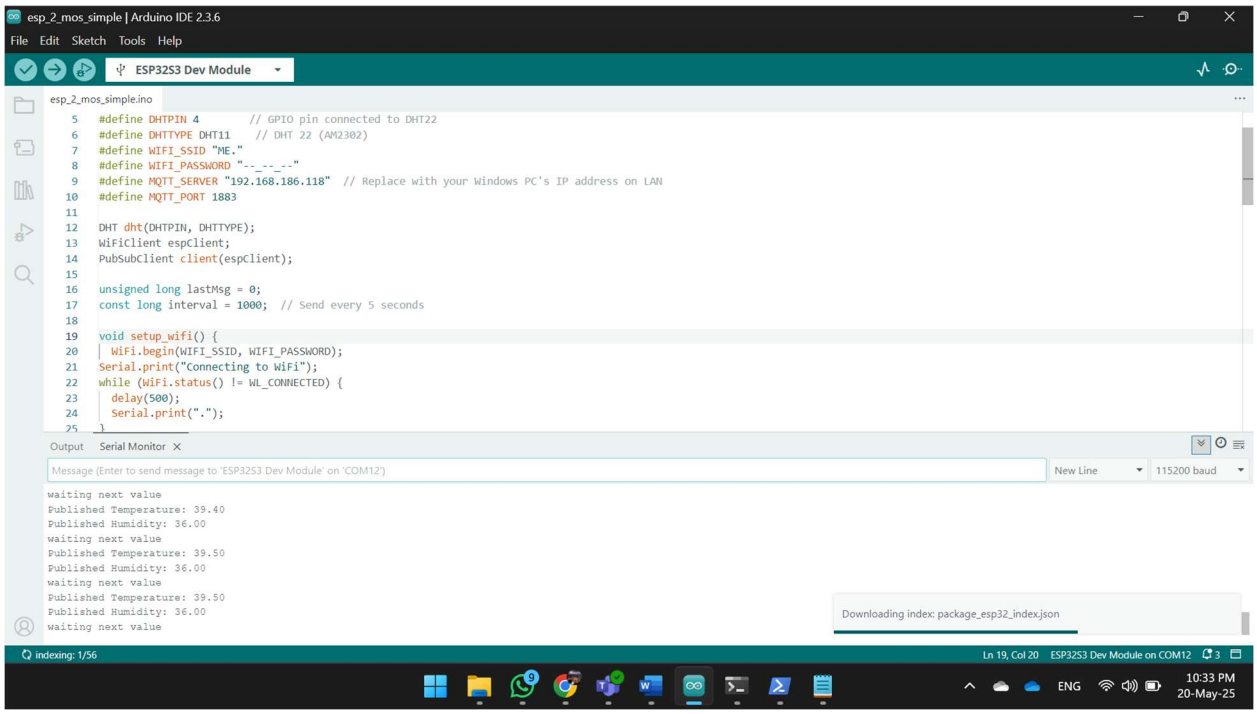
```
#include <WiFi.h>
#include <PubSubClient.h>
#include <DHT.h>

#define DHTPIN 4        // GPIO pin connected to DHT22
#define DHTTYPE DHT11    // DHT 22 (AM2302)
#define WIFI_SSID "ME."
#define WIFI_PASSWORD "--_--_--"
#define MQTT_SERVER "192.168.186.118"  // Replace with your Windows PC's IP address on LAN
#define MQTT_PORT 1883

DHT dht(DHTPIN, DHTTYPE);
WiFiClient espClient;
PubSubClient client(espClient);

unsigned long lastMsg = 0;
const long interval = 1000;  // Send every 5 seconds

void setup_wifi() {
  WiFi.begin(WIFI_SSID, WIFI_PASSWORD);
Serial.print("Connecting to WiFi");
while (WiFi.status() != WL_CONNECTED) {
  delay(500);
  Serial.print(".");
}
Serial.println();
```

```
Serial.println("Connected to WiFi");
Serial.println(WiFi.localIP());  // Print IP to confirm connection

}

void reconnect() {
  while (!client.connected()) {
    Serial.print("Attempting MQTT connection...");
    String clientId = "ESP32Client-";
    clientId += String(random(0xffff), HEX);
    if (client.connect(clientId.c_str())) {
      Serial.println("connected");
    } else {
      Serial.print("failed, rc=");
      Serial.print(client.state());
      Serial.println(" try again in 5 seconds");
      delay(5000);
    }
  }
}

void setup() {
  Serial.begin(115200);
  dht.begin();
  setup_wifi();
  client.setServer(MQTT_SERVER, MQTT_PORT);
}

void loop() {
  if (!client.connected()) {
```

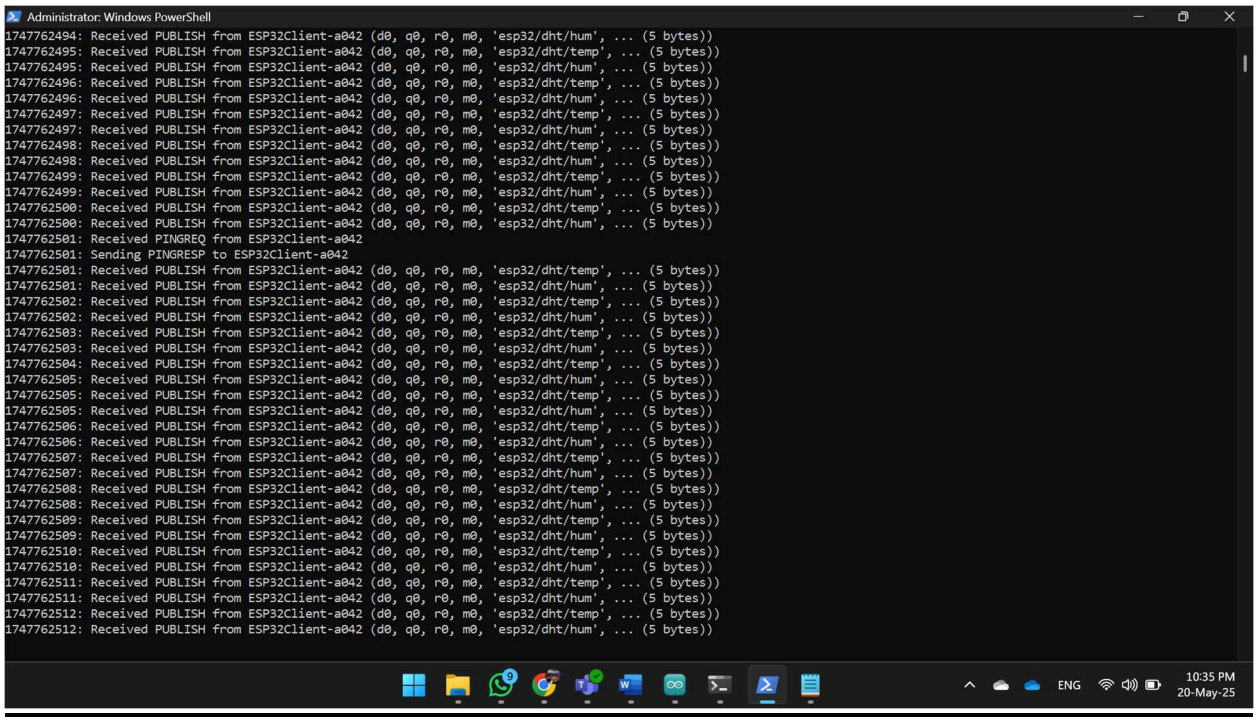
```
  reconnect();
 }
 client.loop();

 unsigned long now = millis();
 if (now - lastMsg > interval) {
  lastMsg = now;
  float temperature = dht.readTemperature();
  float humidity = dht.readHumidity();

  if (isnan(temperature) || isnan(humidity)) {
   Serial.println("Failed to read from DHT sensor!");
   return;
  }

  String tempStr = String(temperature, 2);
  String humStr = String(humidity, 2);

  client.publish("esp32/dht/temp", tempStr.c_str());
  client.publish("esp32/dht/hum", humStr.c_str());

  Serial.print("Published Temperature: ");
  Serial.println(tempStr);
  Serial.print("Published Humidity: ");
  Serial.println(humStr);
  Serial.println("waiting next value");
 }
}
```

Output:

Arduino IDE



Mosquitto MQTT Broker



Execute the Python script 1-dht_data_only.py to store MQTT data in InfluxDB.

Code:

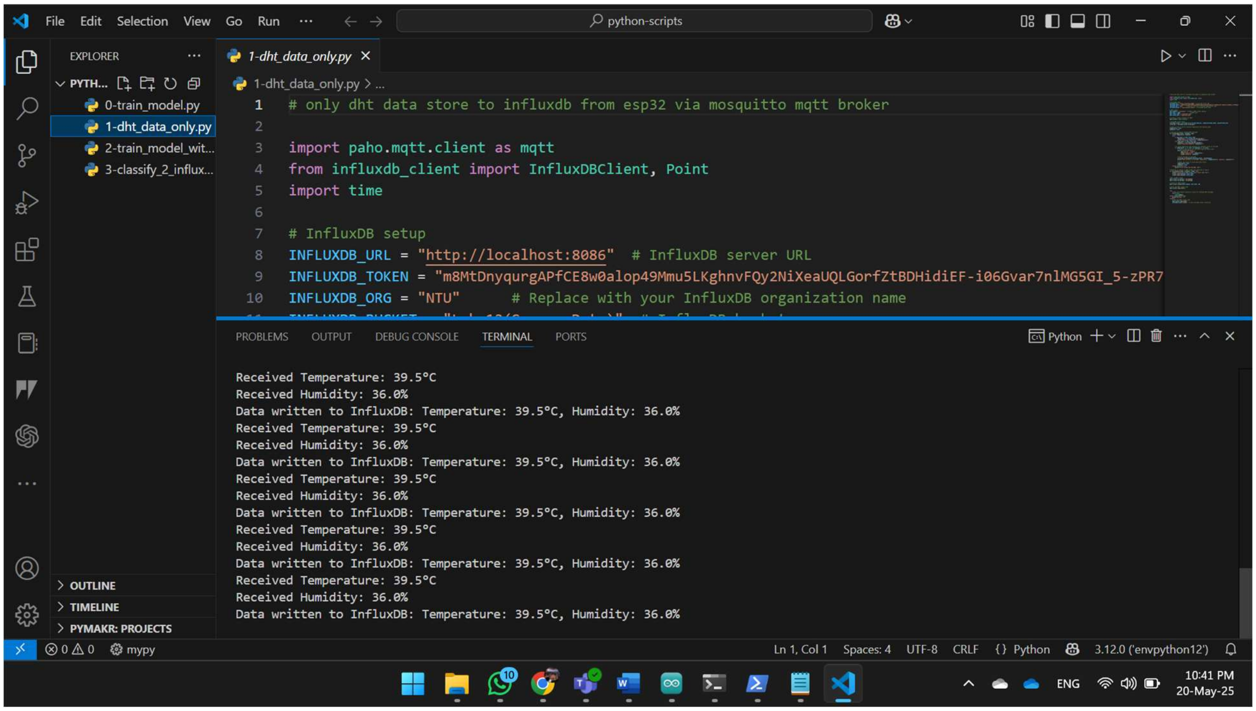
```
# only dht data store to influxdb from esp32 via mosquitto mqtt broker

import paho.mqtt.client as mqtt
from influxdb_client import InfluxDBClient, Point
import time

# InfluxDB setup
INFLUXDB_URL = "http://localhost:8086"  # InfluxDB server URL
INFLUXDB_TOKEN = "m8MtDnyqurgAPfCE8w0alop49Mmu5LKghnvFQy2NiXeaUQLGorfZtBDHidiEF-i06Gvar7nlMG5GI_5-zPR7Ug==" # Replace with your InfluxDB token
INFLUXDB_ORG = "NTU"      # Replace with your InfluxDB organization name
INFLUXDB_BUCKET = "Lab_13(Sensor_Data)"  # InfluxDB bucket name

# MQTT setup
MQTT_BROKER = "localhost"  # ESP32's MQTT broker address
MQTT_PORT = 1883              # MQTT port
MQTT_TOPIC_TEMP = "esp32/dht/temp"
MQTT_TOPIC_HUM = "esp32/dht/hum"

# Create a client instance for MQTT
mqtt_client = mqtt.Client()

# InfluxDB client setup
influxdb_client = InfluxDBClient(url=INFLUXDB_URL, token=INFLUXDB_TOKEN, org=INFLUXDB_ORG)
write_api = influxdb_client.write_api()

# Flag to track if we've received temperature and humidity data
temperature = None
```

```
humidity = None

# Function to handle incoming MQTT messages
def on_message(client, userdata, msg):
    global temperature, humidity
    try:
        if msg.topic == MQTT_TOPIC_TEMP:
            temperature = float(msg.payload.decode())
            print(f"Received Temperature: {temperature}°C")
        elif msg.topic == MQTT_TOPIC_HUM:
            humidity = float(msg.payload.decode())
            print(f"Received Humidity: {humidity}%")

        # If both temperature and humidity are received, write to InfluxDB
        if temperature is not None and humidity is not None:
            # Create a data point for InfluxDB using the Point class
            point = Point("dht_data") \
                .tag("device", "esp32") \
                .field("temperature", temperature) \
                .field("humidity", humidity)

            # Write the data to InfluxDB
            write_api.write(bucket=INFLUXDB_BUCKET, record=point)
            print(f"Data written to InfluxDB: Temperature: {temperature}°C, Humidity: {humidity}%")

            # Reset the values to avoid duplicate writes
            temperature = None
            humidity = None
    except Exception as e:
        print(f"Error processing message: {e}")
```

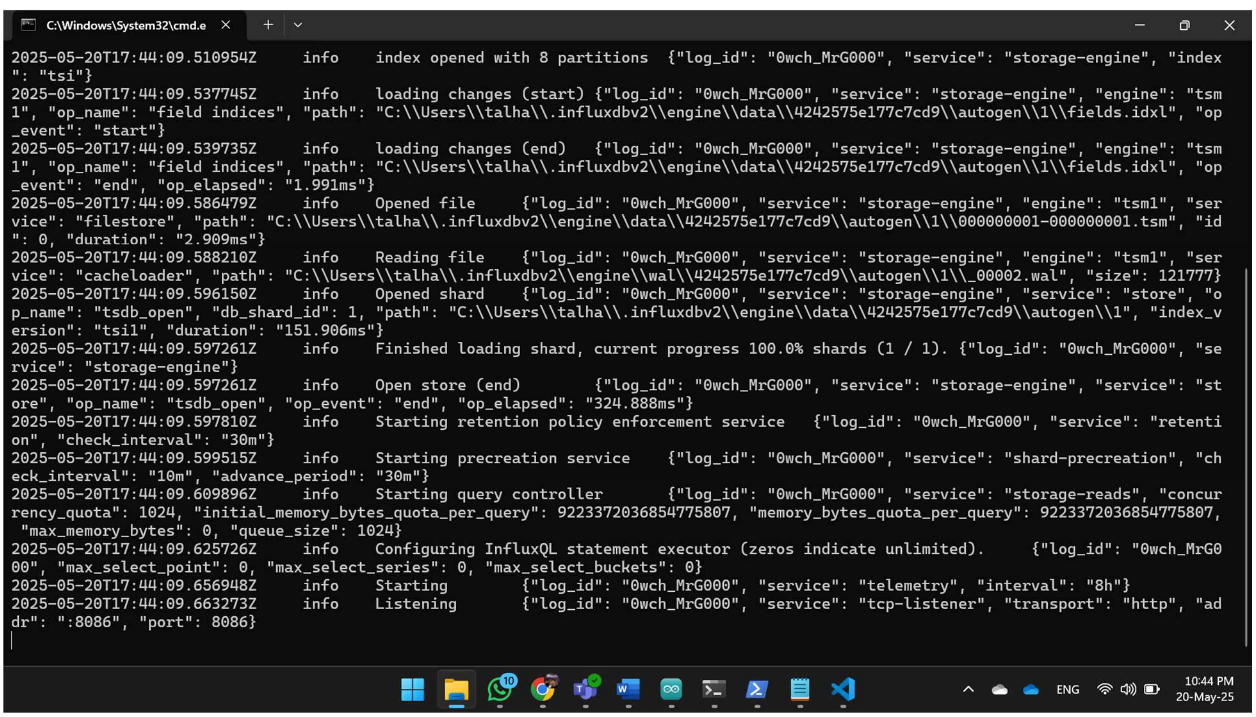
```
# Function to connect to MQTT broker and subscribe to topics
def on_connect(client, userdata, flags, rc):
    print(f"Connected to MQTT broker with result code {rc}")
    client.subscribe(MQTT_TOPIC_TEMP)
    client.subscribe(MQTT_TOPIC_HUM)

# Set up MQTT client
mqtt_client.on_connect = on_connect
mqtt_client.on_message = on_message

# Connect to MQTT broker
mqtt_client.connect(MQTT_BROKER, MQTT_PORT, 60)

# Start the MQTT client loop
mqtt_client.loop_start()

try:
    # Keep the program running to listen for incoming MQTT messages
    while True:
        time.sleep(1)
except KeyboardInterrupt:
    print("Exiting...")
finally:
    # Stop the MQTT client loop
    mqtt_client.loop_stop()
    influxdb_client.close()  # Close InfluxDB client connection
```

Output:

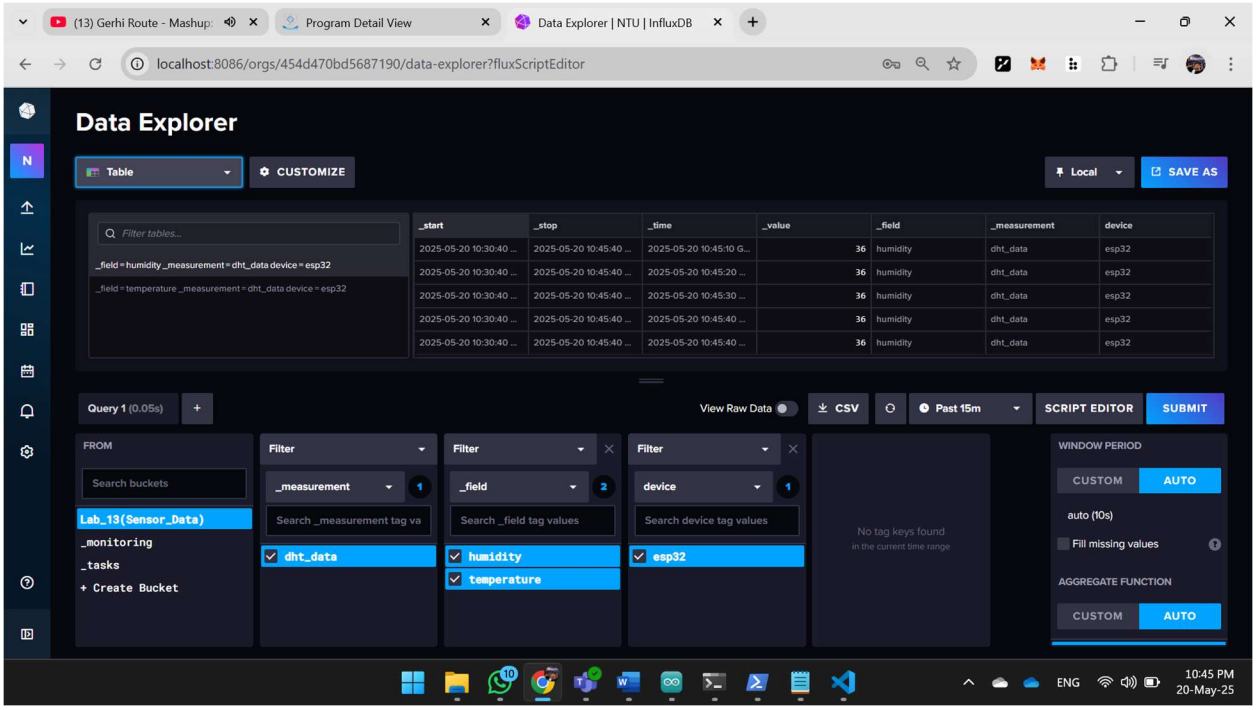
VS Code



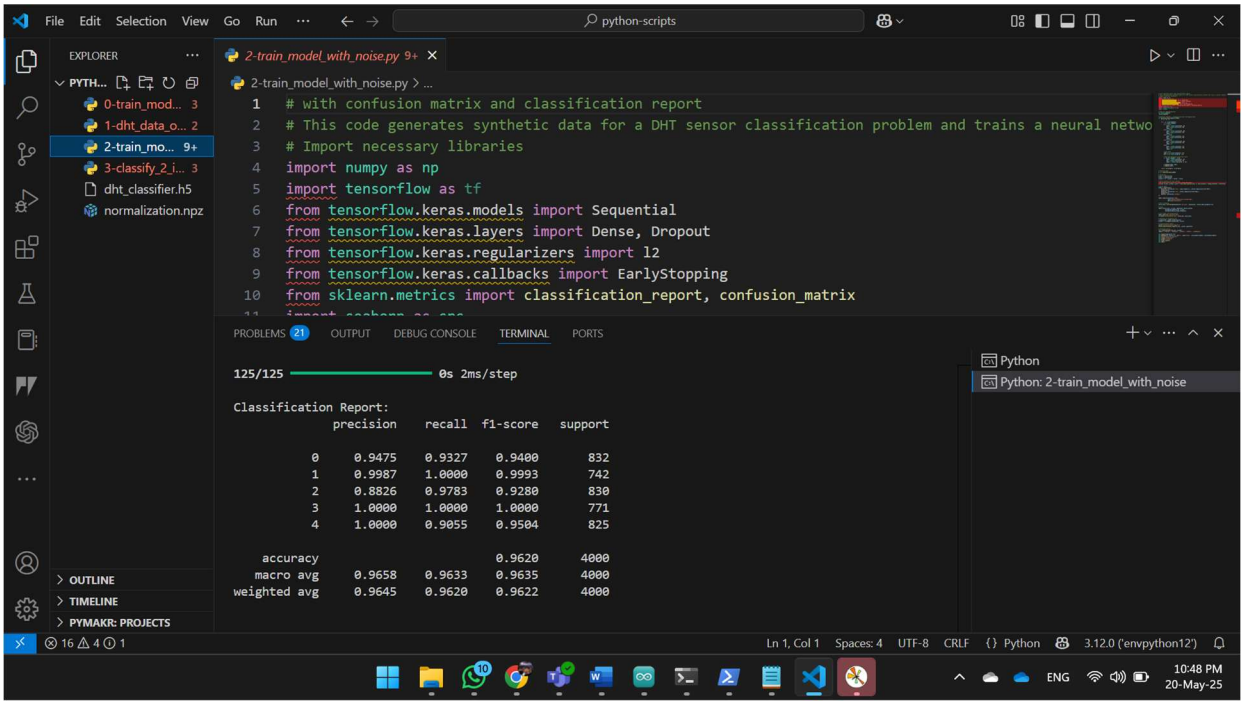
Starting InfluxDB port

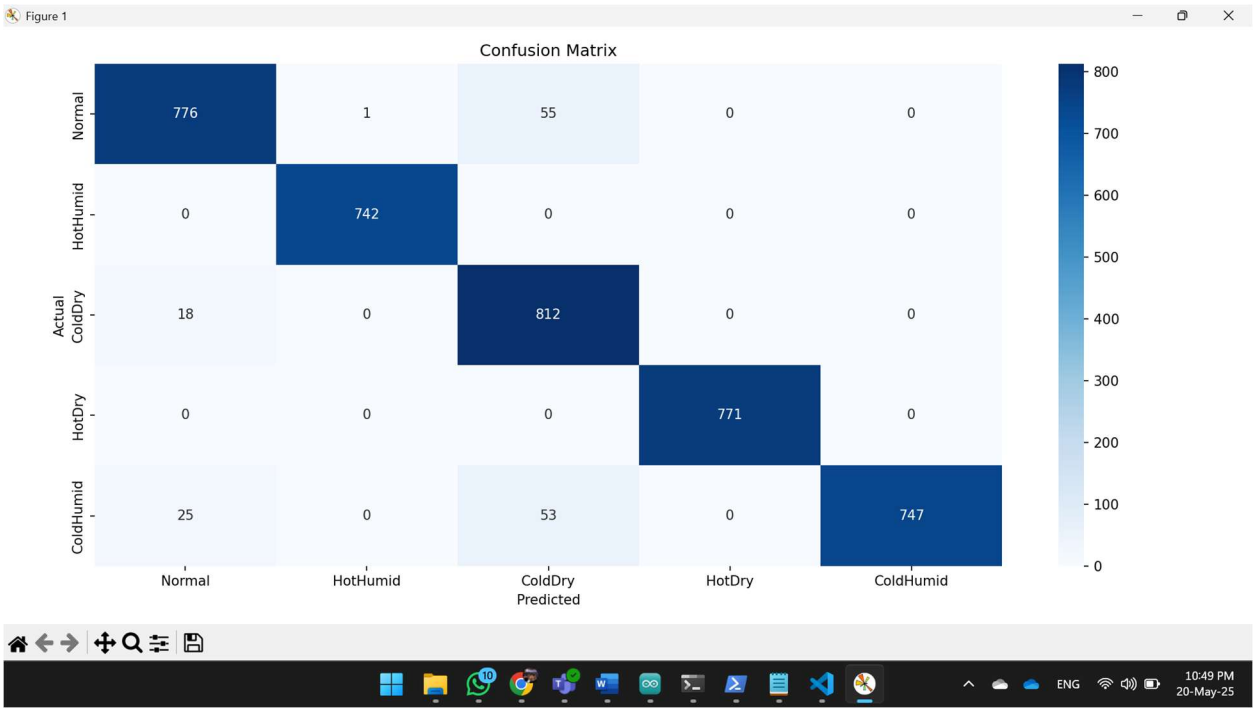


InfluxDB Dashboard



Run 2-train_model_with_noise.py and record the confusion matrix and classification report.





Execute 3-classify_2_influx.py and verify InfluxDB data for temperature, humidity, and classification results.