

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;
    }

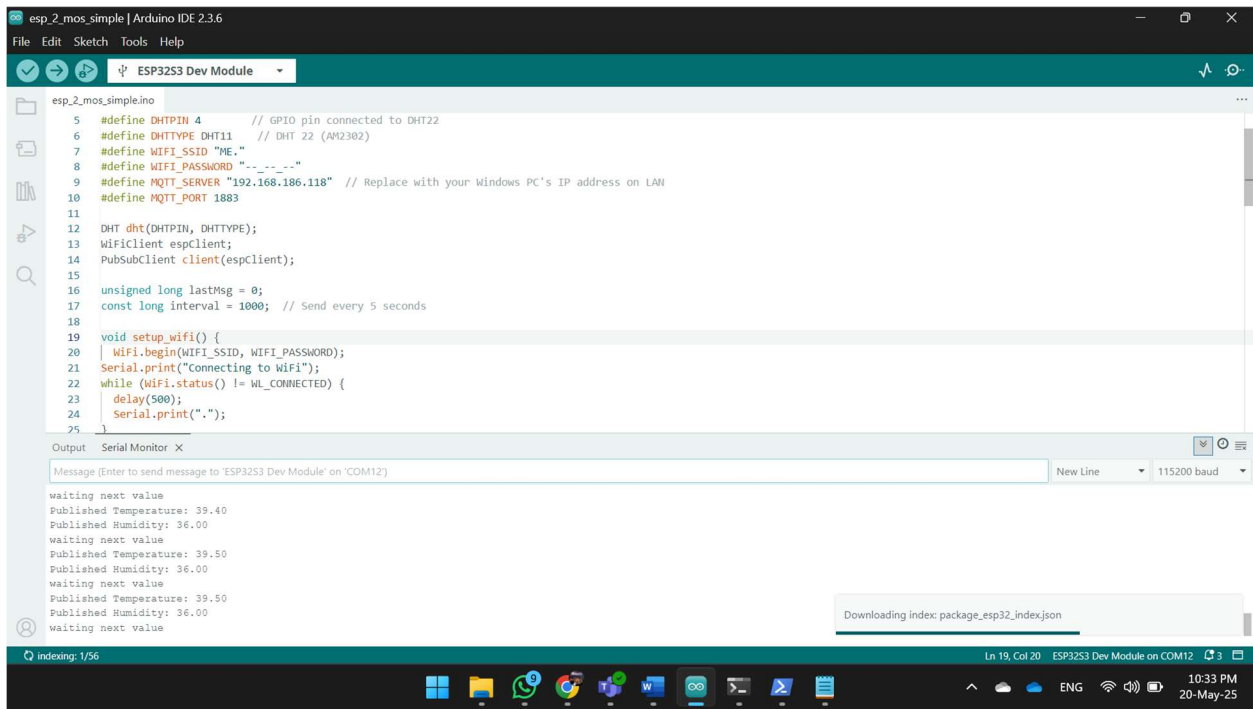
    String tempStr = String(temperature, 2);
    String humStr = String(humidity, 2);

    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

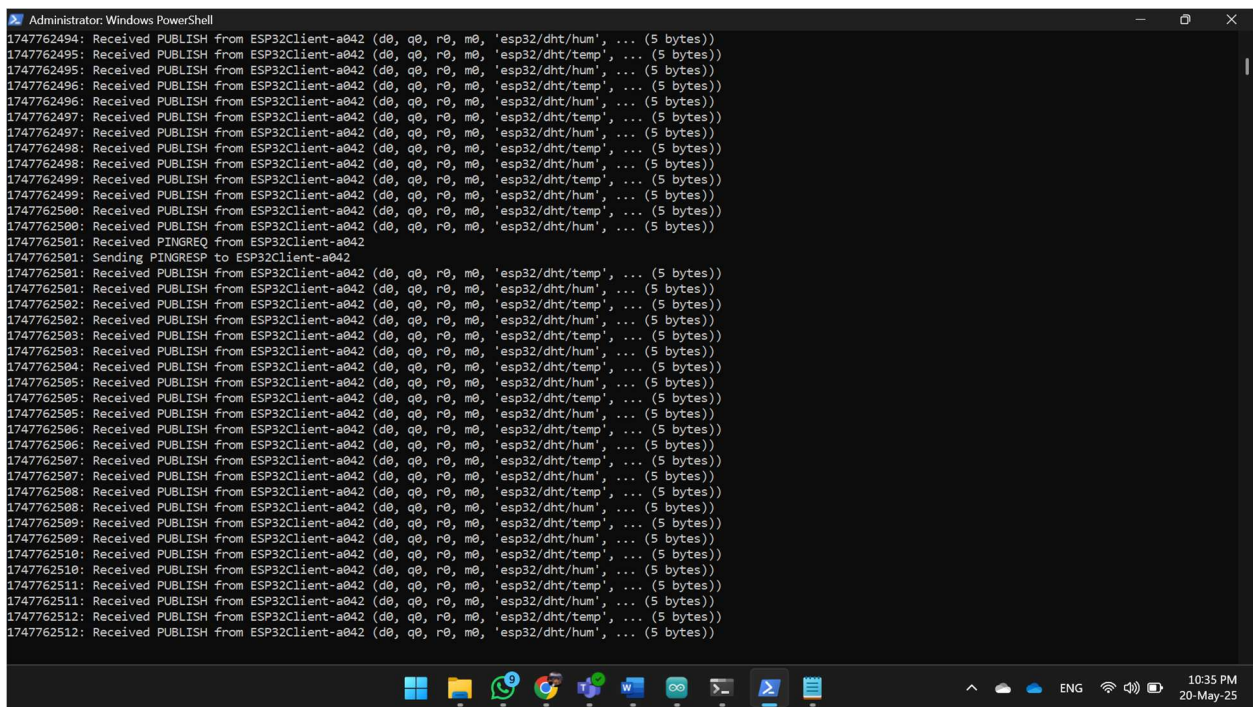


The screenshot shows the Arduino IDE interface with the file `esp_2_mos_simple.ino` open. The code defines a DHT sensor (DHT22) and an MQTT client. The output window shows the following messages:

```
waiting next value
Published Temperature: 39.40
Published Humidity: 36.00
waiting next value
Published Temperature: 39.50
Published Humidity: 36.00
waiting next value
Published Temperature: 39.50
Published Humidity: 36.00
waiting next value
```

The status bar at the bottom indicates the board is set to `ESP32S3 Dev Module` and the port is `COM12`. The system tray shows the time as 10:33 PM on 20-May-25.

Mosquitto MQTT Broker



The screenshot shows a Windows PowerShell window with the following output:

```
1747762494: Received PUBLISH from ESP32Client-a042 (d0, q0, r0, m0, 'esp32/dht/hum', ... (5 bytes))
1747762495: Received PUBLISH from ESP32Client-a042 (d0, q0, r0, m0, 'esp32/dht/temp', ... (5 bytes))
1747762495: Received PUBLISH from ESP32Client-a042 (d0, q0, r0, m0, 'esp32/dht/hum', ... (5 bytes))
1747762496: Received PUBLISH from ESP32Client-a042 (d0, q0, r0, m0, 'esp32/dht/temp', ... (5 bytes))
1747762496: Received PUBLISH from ESP32Client-a042 (d0, q0, r0, m0, 'esp32/dht/hum', ... (5 bytes))
1747762497: Received PUBLISH from ESP32Client-a042 (d0, q0, r0, m0, 'esp32/dht/temp', ... (5 bytes))
1747762497: Received PUBLISH from ESP32Client-a042 (d0, q0, r0, m0, 'esp32/dht/hum', ... (5 bytes))
1747762498: Received PUBLISH from ESP32Client-a042 (d0, q0, r0, m0, 'esp32/dht/temp', ... (5 bytes))
1747762498: Received PUBLISH from ESP32Client-a042 (d0, q0, r0, m0, 'esp32/dht/hum', ... (5 bytes))
1747762499: Received PUBLISH from ESP32Client-a042 (d0, q0, r0, m0, 'esp32/dht/temp', ... (5 bytes))
1747762499: Received PUBLISH from ESP32Client-a042 (d0, q0, r0, m0, 'esp32/dht/hum', ... (5 bytes))
1747762500: Received PUBLISH from ESP32Client-a042 (d0, q0, r0, m0, 'esp32/dht/temp', ... (5 bytes))
1747762500: Received PUBLISH from ESP32Client-a042 (d0, q0, r0, m0, 'esp32/dht/hum', ... (5 bytes))
1747762501: Received PINGREQ from ESP32Client-a042
1747762501: Sending PINGRESP to ESP32Client-a042
1747762501: Received PUBLISH from ESP32Client-a042 (d0, q0, r0, m0, 'esp32/dht/temp', ... (5 bytes))
1747762501: Received PUBLISH from ESP32Client-a042 (d0, q0, r0, m0, 'esp32/dht/hum', ... (5 bytes))
1747762502: Received PUBLISH from ESP32Client-a042 (d0, q0, r0, m0, 'esp32/dht/temp', ... (5 bytes))
1747762502: Received PUBLISH from ESP32Client-a042 (d0, q0, r0, m0, 'esp32/dht/hum', ... (5 bytes))
1747762503: Received PUBLISH from ESP32Client-a042 (d0, q0, r0, m0, 'esp32/dht/temp', ... (5 bytes))
1747762503: Received PUBLISH from ESP32Client-a042 (d0, q0, r0, m0, 'esp32/dht/hum', ... (5 bytes))
1747762504: Received PUBLISH from ESP32Client-a042 (d0, q0, r0, m0, 'esp32/dht/temp', ... (5 bytes))
1747762505: Received PUBLISH from ESP32Client-a042 (d0, q0, r0, m0, 'esp32/dht/hum', ... (5 bytes))
1747762505: Received PUBLISH from ESP32Client-a042 (d0, q0, r0, m0, 'esp32/dht/temp', ... (5 bytes))
1747762505: Received PUBLISH from ESP32Client-a042 (d0, q0, r0, m0, 'esp32/dht/hum', ... (5 bytes))
1747762506: Received PUBLISH from ESP32Client-a042 (d0, q0, r0, m0, 'esp32/dht/temp', ... (5 bytes))
1747762506: Received PUBLISH from ESP32Client-a042 (d0, q0, r0, m0, 'esp32/dht/hum', ... (5 bytes))
1747762507: Received PUBLISH from ESP32Client-a042 (d0, q0, r0, m0, 'esp32/dht/temp', ... (5 bytes))
1747762507: Received PUBLISH from ESP32Client-a042 (d0, q0, r0, m0, 'esp32/dht/hum', ... (5 bytes))
1747762508: Received PUBLISH from ESP32Client-a042 (d0, q0, r0, m0, 'esp32/dht/temp', ... (5 bytes))
1747762508: Received PUBLISH from ESP32Client-a042 (d0, q0, r0, m0, 'esp32/dht/hum', ... (5 bytes))
1747762509: Received PUBLISH from ESP32Client-a042 (d0, q0, r0, m0, 'esp32/dht/temp', ... (5 bytes))
1747762509: Received PUBLISH from ESP32Client-a042 (d0, q0, r0, m0, 'esp32/dht/hum', ... (5 bytes))
1747762510: Received PUBLISH from ESP32Client-a042 (d0, q0, r0, m0, 'esp32/dht/temp', ... (5 bytes))
1747762510: Received PUBLISH from ESP32Client-a042 (d0, q0, r0, m0, 'esp32/dht/hum', ... (5 bytes))
1747762511: Received PUBLISH from ESP32Client-a042 (d0, q0, r0, m0, 'esp32/dht/temp', ... (5 bytes))
1747762511: Received PUBLISH from ESP32Client-a042 (d0, q0, r0, m0, 'esp32/dht/hum', ... (5 bytes))
1747762512: Received PUBLISH from ESP32Client-a042 (d0, q0, r0, m0, 'esp32/dht/temp', ... (5 bytes))
1747762512: Received PUBLISH from ESP32Client-a042 (d0, q0, r0, m0, 'esp32/dht/hum', ... (5 bytes))
```

The status bar at the bottom shows the time as 10:35 PM on 20-May-25.

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-i06Gvar7nIMG5GI_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

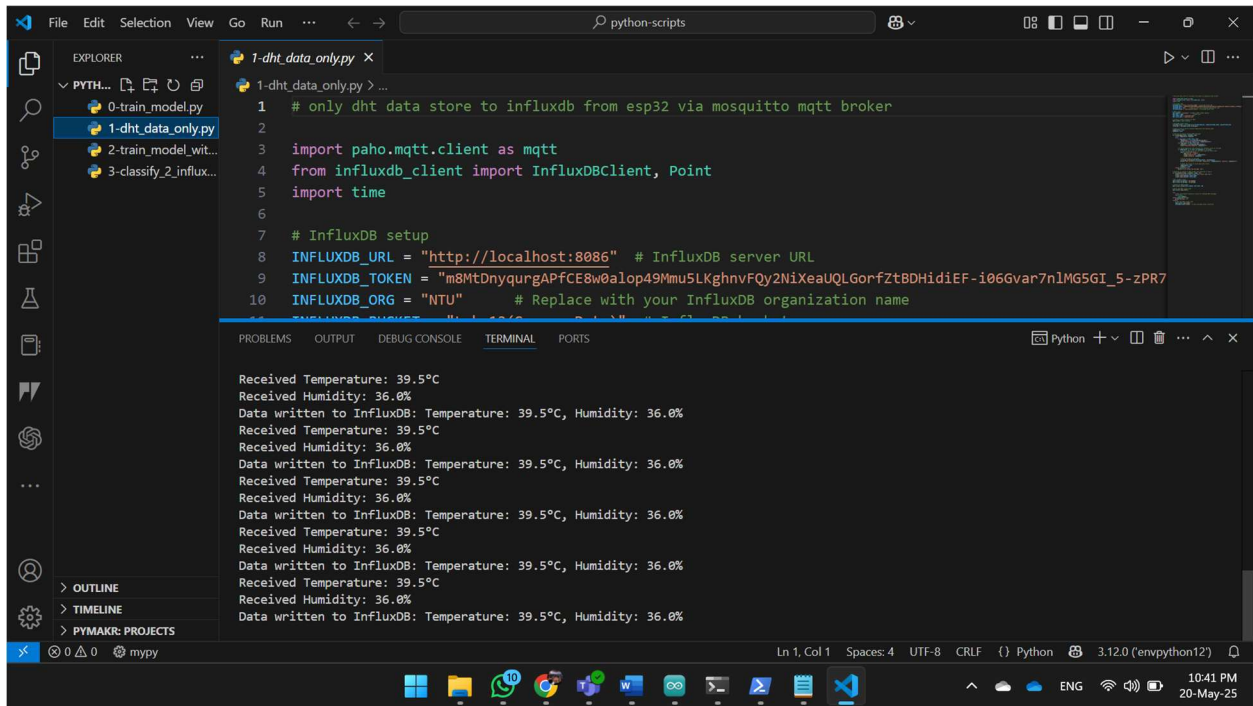
# Connect to MQTT broker
mqtt_client.connect(MQTT_BROKER, MQTT_PORT, 60)

# 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



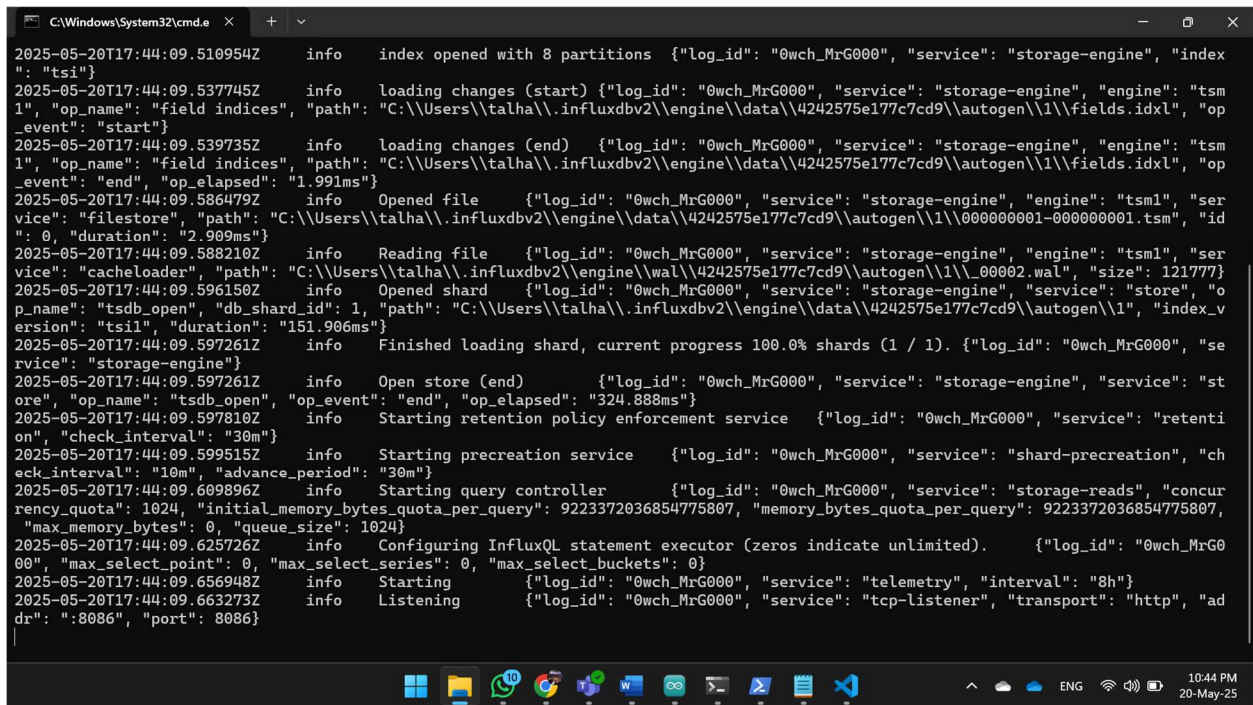
The screenshot shows the Visual Studio Code editor with a Python script named `1-dht_data_only.py` open. The script is configured to send DHT sensor data to InfluxDB via a Mosquitto MQTT broker. The terminal output shows the script running successfully, receiving temperature and humidity data, and writing it to InfluxDB.

```
1 # only dht data store to influxdb from esp32 via mosquitto mqtt broker
2
3 import paho.mqtt.client as mqtt
4 from influxdb_client import InfluxDBClient, Point
5 import time
6
7 # InfluxDB setup
8 INFLUXDB_URL = "http://localhost:8086" # InfluxDB server URL
9 INFLUXDB_TOKEN = "m8MtDnyqurgAPfCE8w0alop49Mmu5LKghnvFQy2NiXeaUQLGorfZtBDHidIEF-106Gvar7n1MG5GI_5-zPR7"
10 INFLUXDB_ORG = "NTU" # Replace with your InfluxDB organization name
```

Terminal Output:

```
Received Temperature: 39.5°C
Received Humidity: 36.0%
Data written to InfluxDB: Temperature: 39.5°C, Humidity: 36.0%
Received Temperature: 39.5°C
Received Humidity: 36.0%
Data written to InfluxDB: Temperature: 39.5°C, Humidity: 36.0%
Received Temperature: 39.5°C
Received Humidity: 36.0%
Data written to InfluxDB: Temperature: 39.5°C, Humidity: 36.0%
Received Temperature: 39.5°C
Received Humidity: 36.0%
Data written to InfluxDB: Temperature: 39.5°C, Humidity: 36.0%
Received Temperature: 39.5°C
Received Humidity: 36.0%
Data written to InfluxDB: Temperature: 39.5°C, Humidity: 36.0%
Received Temperature: 39.5°C
Received Humidity: 36.0%
Data written to InfluxDB: Temperature: 39.5°C, Humidity: 36.0%
```

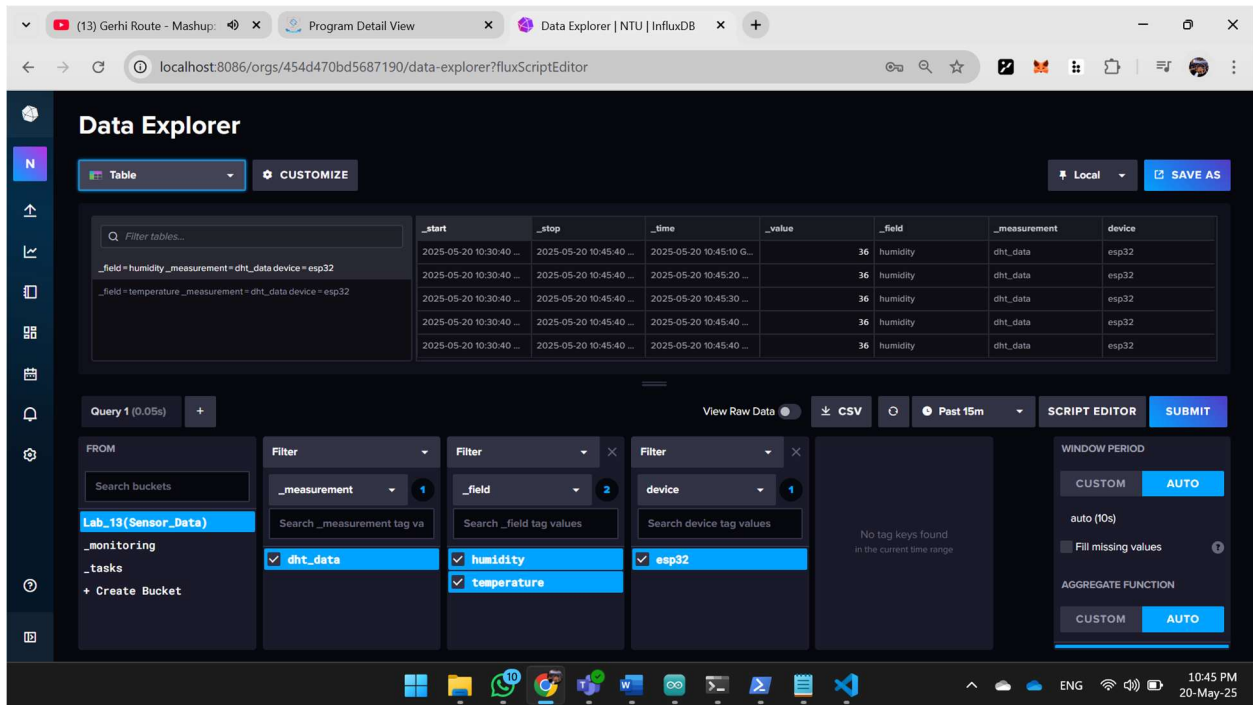
Starting InfluxDB port



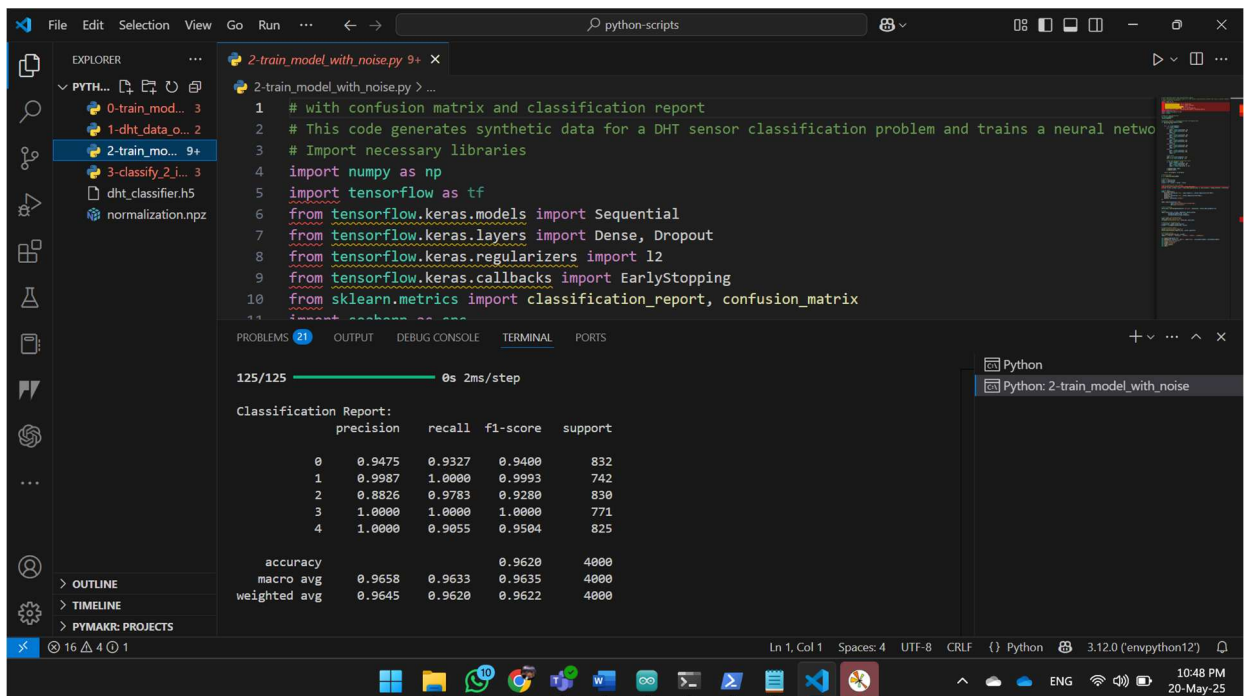
The screenshot shows a Windows command prompt window displaying the logs for starting InfluxDB. The logs indicate that the index is opened, changes are loaded, and the storage engine is initialized. The service is then started, and the retention policy enforcement service is also started.

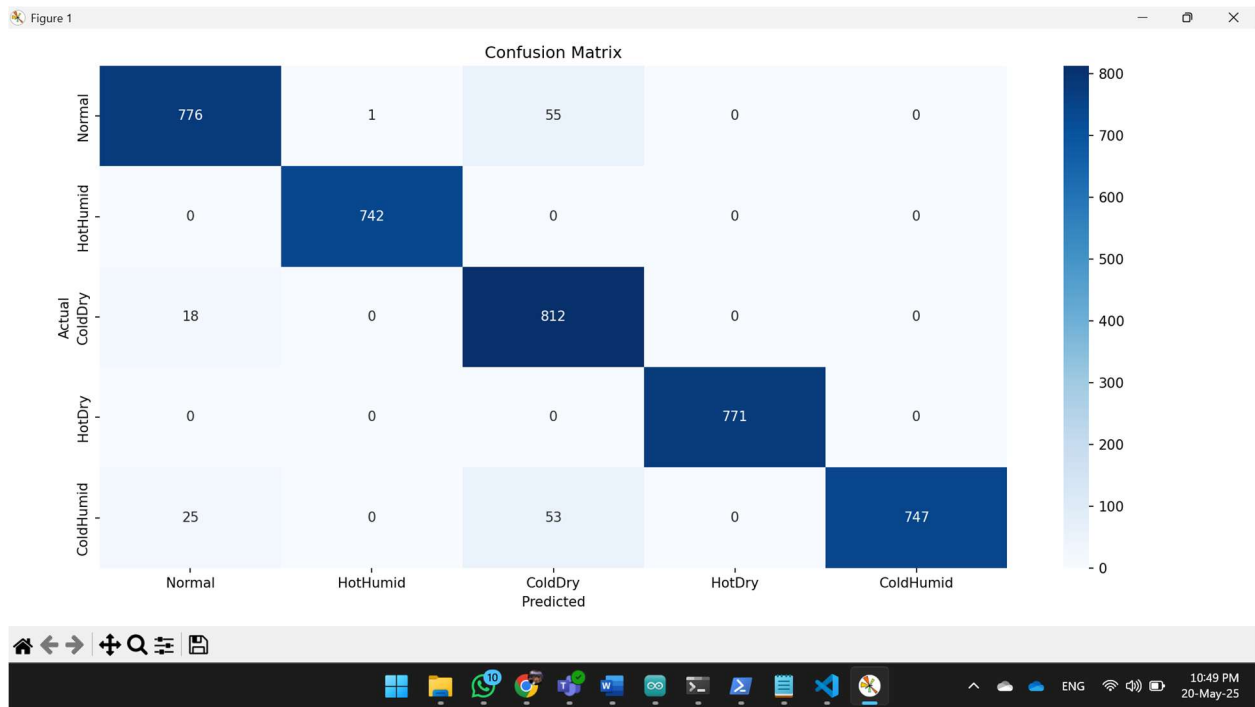
```
2025-05-20T17:44:09.510954Z info index opened with 8 partitions {"log_id": "0wch_MrG000", "service": "storage-engine", "index": "tsi"}
2025-05-20T17:44:09.537745Z info loading changes (start) {"log_id": "0wch_MrG000", "service": "storage-engine", "engine": "tsm1", "op_name": "field indices", "path": "C:\\Users\\talha\\.influxdbv2\\engine\\data\\4242575e177c7cd9\\autogen\\1\\fields.idx", "op_event": "start"}
2025-05-20T17:44:09.539735Z info loading changes (end) {"log_id": "0wch_MrG000", "service": "storage-engine", "engine": "tsm1", "op_name": "field indices", "path": "C:\\Users\\talha\\.influxdbv2\\engine\\data\\4242575e177c7cd9\\autogen\\1\\fields.idx", "op_event": "end", "op_elapsed": "1.991ms"}
2025-05-20T17:44:09.586479Z info Opened file {"log_id": "0wch_MrG000", "service": "storage-engine", "engine": "tsm1", "service": "filestore", "path": "C:\\Users\\talha\\.influxdbv2\\engine\\data\\4242575e177c7cd9\\autogen\\1\\000000001-000000001.tsm", "id": 0, "duration": "2.909ms"}
2025-05-20T17:44:09.588210Z info Reading file {"log_id": "0wch_MrG000", "service": "storage-engine", "engine": "tsm1", "service": "cacheloader", "path": "C:\\Users\\talha\\.influxdbv2\\engine\\wal\\4242575e177c7cd9\\autogen\\1\\00002.wal", "size": 121777}
2025-05-20T17:44:09.596150Z info Opened shard {"log_id": "0wch_MrG000", "service": "storage-engine", "service": "store", "op_name": "tsdb_open", "db_shard_id": 1, "path": "C:\\Users\\talha\\.influxdbv2\\engine\\data\\4242575e177c7cd9\\autogen\\1", "index_version": "tsi1", "duration": "151.906ms"}
2025-05-20T17:44:09.597261Z info Finished loading shard, current progress 100.0% shards (1 / 1). {"log_id": "0wch_MrG000", "service": "storage-engine"}
2025-05-20T17:44:09.597261Z info Open store (end) {"log_id": "0wch_MrG000", "service": "storage-engine", "service": "store", "op_name": "tsdb_open", "op_event": "end", "op_elapsed": "324.888ms"}
2025-05-20T17:44:09.597810Z info Starting retention policy enforcement service {"log_id": "0wch_MrG000", "service": "retention", "check_interval": "30m"}
2025-05-20T17:44:09.599515Z info Starting precreation service {"log_id": "0wch_MrG000", "service": "shard-precreation", "check_interval": "10m", "advance_period": "30m"}
2025-05-20T17:44:09.609896Z info Starting query controller {"log_id": "0wch_MrG000", "service": "storage-reads", "concurrency_quota": 1024, "initial_memory_bytes_quota_per_query": 9223372036854775807, "memory_bytes_quota_per_query": 9223372036854775807, "max_memory_bytes": 0, "queue_size": 1024}
2025-05-20T17:44:09.625726Z info Configuring InfluxQL statement executor (zeros indicate unlimited). {"log_id": "0wch_MrG000", "max_select_point": 0, "max_select_series": 0, "max_select_buckets": 0}
2025-05-20T17:44:09.656948Z info Starting {"log_id": "0wch_MrG000", "service": "telemetry", "interval": "8h"}
2025-05-20T17:44:09.663273Z info Listening {"log_id": "0wch_MrG000", "service": "tcp-listener", "transport": "http", "address": "8086", "port": 8086}
```

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.

```
File Edit Selection View Go Run ... python-scripts
3-classify_2_influx_py 3 x 1-dht_data_only.py 2
3-classify_2_influx_py > ...
11 INFLUXDB_TOKEN = "m8MtDnyqurgAPfCE8w0alop49Mmu5LKghnvFQy2NiXeaUQLGorfZtBDHidief-106Gvar7n1MG5GI_5-zPR7"
12 INFLUXDB_ORG = "NTU" # Replace with your InfluxDB organization name
13 INFLUXDB_BUCKET = "Lab_13(Sensor_Data)" # InfluxDB bucket name
14
15 # MQTT setup
16 MQTT_BROKER = "localhost" # ESP32's MQTT broker address
17 MQTT_PORT = 1883 # MQTT port
18 MQTT_TOPIC_TEMP = "esp32/dht/temp"
19 MQTT_TOPIC_HUM = "esp32/dht/hum"
20
21 # Class ...

747763631879809000
Data saved: Temp=39.50, Hum=36.00, Class=Hot and Dry
Received Humidity: 36.00%
Received Temperature: 39.50°C
Predicted Class: Hot and Dry
Writing to InfluxDB: dht_data,device=esp32 class_label="Hot and Dry",humidity=36,temperature=39.5 1
747763632908900000
Data saved: Temp=39.50, Hum=36.00, Class=Hot and Dry
Received Humidity: 36.00%
Received Temperature: 39.50°C
Predicted Class: Hot and Dry
Writing to InfluxDB: dht_data,device=esp32 class_label="Hot and Dry",humidity=36,temperature=39.5 1
747763633937184000
Data saved: Temp=39.50, Hum=36.00, Class=Hot and Dry
Received Humidity: 36.00%

Ln 16, Col 25 Spaces: 4 UTF-8 CRLF {} Python 3.12.0 (env:python12)
10:53 PM 20-May-25
```