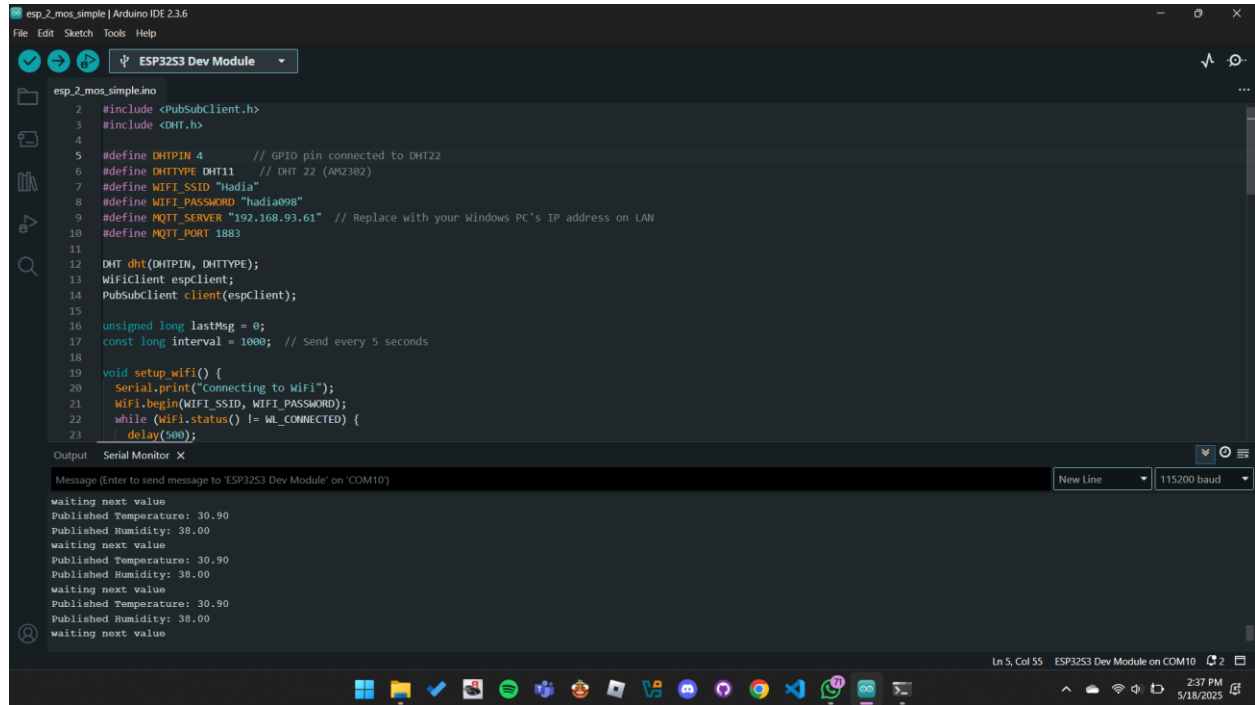




Name	Hadia Alvi
Class	BSAI – 6th
Reg. No	22-NTU-CS-1343
Course Name	IoT Fundamentals
Lab	Lab13
Submission Date	20 th May,2025.
Submitted To	Sir Nasir

Sending data via MQTT:

For sending data I am using ESP32S3 microcontroller that reads data from a DHT22 temperature and humidity sensor and sends it over WiFi to an MQTT broker.



```
esp_2_mos_simple.ino
1 #include <PubSubClient.h>
2 #include <DHT.h>
3
4
5 #define DHTPIN 4 // GPIO pin connected to DHT22
6 #define DHTTYPE DHT11 // DHT 22 (AM2302)
7 #define WIFI_SSID "Hadia"
8 #define WIFI_PASSWORD "hadia098"
9 #define MQTT_SERVER "192.168.93.61" // Replace with your Windows PC's IP address on LAN
10 #define MQTT_PORT 1883
11
12 DHT dht(DHTPIN, DHTTYPE);
13 WiFiClient espClient;
14 PubSubClient client(espClient);
15
16 unsigned long lastMsg = 0;
17 const long interval = 1000; // Send every 5 seconds
18
19 void setup_wifi() {
20   Serial.print("Connecting to WiFi");
21   WiFi.begin(WIFI_SSID, WIFI_PASSWORD);
22   while (WiFi.status() != WL_CONNECTED) {
23     delay(500);
24   }
25 }
26
27 void loop() {
28   client.connect("ESP32S3");
29   if (!client.connected()) {
30     Serial.println("WiFi disconnected. Retrying...");
31     delay(5000);
32     client.connect("ESP32S3");
33   }
34
35   float temperature = dht.readTemperature();
36   float humidity = dht.readHumidity();
37
38   if (isnan(temperature) || isnan(humidity)) {
39     Serial.println("Failed to read from DHT sensor!");
40     return;
41   }
42
43   String msg = "Temperature: " + String(temperature, 1) + " Humidity: " + String(humidity, 1);
44   client.publish("esp32s3_data", msg);
45   Serial.println(msg);
46
47   lastMsg = millis();
48   delay(interval);
49 }
```

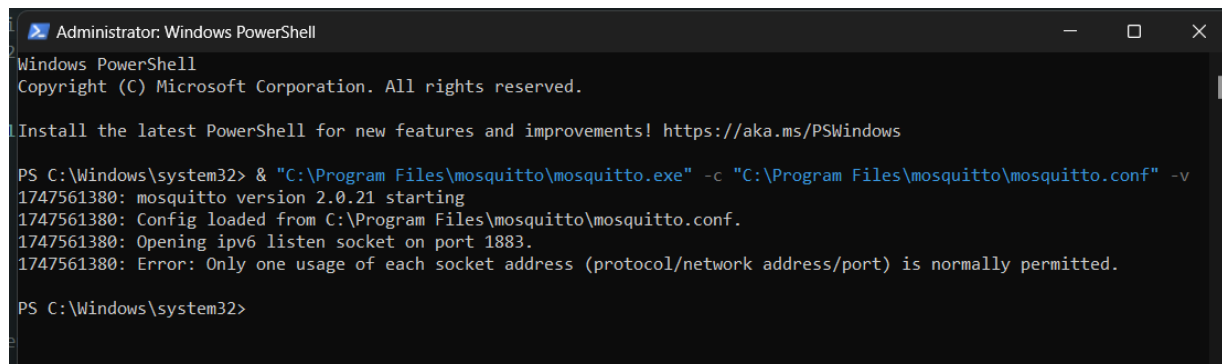
Output

```
Message (Enter to send message to 'ESP32S3 Dev Module' on 'COM10')
waiting next value
Published Temperature: 30.90
Published Humidity: 38.00
waiting next value
Published Temperature: 30.90
Published Humidity: 38.00
waiting next value
Published Temperature: 30.90
Published Humidity: 38.00
waiting next value
Published Temperature: 30.90
Published Humidity: 38.00
waiting next value
```

Now I run This command on powershell(run as administrator):

& "C:\Program Files\mosquitto\mosquitto.exe" -c "C:\Program Files\mosquitto\mosquitto.conf" -v

To Starts the Mosquitto broker manually



```
Administrator: Windows PowerShell
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\Windows\system32> & "C:\Program Files\mosquitto\mosquitto.exe" -c "C:\Program Files\mosquitto\mosquitto.conf" -v
1747561380: mosquitto version 2.0.21 starting
1747561380: Config loaded from C:\Program Files\mosquitto\mosquitto.conf.
1747561380: Opening ipv6 listen socket on port 1883.
1747561380: Error: Only one usage of each socket address (protocol/network address/port) is normally permitted.

PS C:\Windows\system32>
```

This screenshot shows the **InfluxDB** service starting up via the Windows Command Prompt. The logs confirm that the database service is successfully initializing its components, loading metadata and shards, and preparing to store and query time-series data.

```
C:\Windows\System32\cmd.exe
Microsoft Windows [Version 10.0.22621.5335]
(c) Microsoft Corporation. All rights reserved.

C:\Program Files\InfluxData\influxdb>.\\influxd.exe
2025-05-18T09:48:34.67987Z info Welcome to InfluxDB {"log_id": "0w_hZsxW000", "version": "v2.7.11", "commit": "bf65d4ab5e", "build_date": "2024-12-02T17:48:13Z", "log_level": "info"}
2025-05-18T09:48:34.695887Z info Resources opened {"log_id": "0w_hZsxW000", "service": "bolt", "path": "C:\\Users\\Admin\\influxdbv2\\influxd.bolt"}
2025-05-18T09:48:34.695887Z info Resources opened {"log_id": "0w_hZsxW000", "service": "sqlite", "path": "C:\\Users\\Admin\\influxdbv2\\influxd.sqlite"}
2025-05-18T09:48:34.729697Z info Checking InfluxDB metadata for prior version. {"log_id": "0w_hZsxW000", "bolt_path": "C:\\Users\\Admin\\influxdbv2\\influxd.bolt"}
2025-05-18T09:48:34.734372Z info Using data dir {"log_id": "0w_hZsxW000", "service": "storage-engine", "service": "store", "path": "C:\\Users\\Admin\\influxdbv2\\engine\\data"}
2025-05-18T09:48:34.734907Z info Compaction settings {"log_id": "0w_hZsxW000", "service": "storage-engine", "service": "store", "max_concurrent_compactions": 4, "throughput_bytes_per_second": 50331648, "throughput_bytes_per_second_burst": 50331648}
2025-05-18T09:48:34.734907Z info Open store (start) {"log_id": "0w_hZsxW000", "service": "storage-engine", "service": "store", "op_name": "tsdb_open", "op_event": "start"}
2025-05-18T09:48:34.770762Z info TSI log compaction (start) {"log_id": "0w_hZsxW000", "service": "storage-engine", "index": "tsi", "tsi_partition": "5", "op_name": "tsi_compact_log_file", "tsi_log_file_id": 1, "op_event": "start"}
2025-05-18T09:48:34.772247Z info index opened with 8 partitions {"log_id": "0w_hZsxW000", "service": "storage-engine", "index": "tsi", "tsi_partition": "5", "op_name": "tsi_compact_log_file", "tsi_log_file_id": 1, "elapsed": "10ms", "bytes": 507, "kb_per_sec": 47}
2025-05-18T09:48:34.776914Z info TSI log compaction (end) {"log_id": "0w_hZsxW000", "service": "storage-engine", "index": "tsi", "tsi_partition": "5", "op_name": "tsi_compact_log_file", "tsi_log_file_id": 1, "op_event": "end", "op_elapsed": "10.977ms"}
2025-05-18T09:48:34.777472Z info Loading changes (start) {"log_id": "0w_hZsxW000", "service": "storage-engine", "engine": "tsml", "op_name": "field_indices", "path": "C:\\Users\\Admin\\influxdbv2\\engine\\data\\6ab35a10a12f1703\\autogen\\2\\fields.idx", "op_event": "start"}
2025-05-18T09:48:34.779474Z info loading changes (end) {"log_id": "0w_hZsxW000", "service": "storage-engine", "engine": "tsml", "op_name": "field_indices", "path": "C:\\Users\\Admin\\influxdbv2\\engine\\data\\6ab35a10a12f1703\\autogen\\2\\fields.idx", "op_event": "end", "op_elapsed": "0.545ms"}
2025-05-18T09:48:34.785875Z info Opened file {"log_id": "0w_hZsxW000", "service": "storage-engine", "engine": "tsml", "service": "filestore", "path": "C:\\Users\\Admin\\influxdbv2\\engine\\data\\6ab35a10a12f1703\\autogen\\2\\000000002-000000001.tsml", "id": 1, "duration": "1.643ms"}
2025-05-18T09:48:34.785875Z info Opened file {"log_id": "0w_hZsxW000", "service": "storage-engine", "engine": "tsml", "service": "filestore", "path": "C:\\Users\\Admin\\influxdbv2\\engine\\data\\6ab35a10a12f1703\\autogen\\2\\000000003-000000001.tsml", "id": 0, "duration": "1.167ms"}
2025-05-18T09:48:34.785875Z info TSI log compaction (start) {"log_id": "0w_hZsxW000", "service": "storage-engine", "index": "tsi", "tsi_partition": "1", "op_name": "tsi_compact_log_file", "tsi_log_file_id": 1, "op_event": "start"}
2025-05-18T09:48:34.786194Z info Reading file {"log_id": "0w_hZsxW000", "service": "storage-engine", "engine": "tsml", "service": "cacheloader", "path": "C:\\Users\\Admin\\influxdbv2\\engine\\wal\\6ab35a10a12f1703\\autogen\\2\\00004.wal", "size": 15053}
2025-05-18T09:48:34.789947Z info Opened shard {"log_id": "0w_hZsxW000", "service": "storage-engine", "service": "store", "op_name": "tsdb_open", "db_shard_id": 2, "path": "C:\\Users\\Admin\\influxdbv2\\engine\\data\\6ab35a10a12f1703\\autogen\\2", "index_version": "tsi1", "duration": "35.363ms"}
2025-05-18T09:48:34.789579Z info Finished loading shard, current progress 50.0% shards (1 / 2). {"log_id": "0w_hZsxW000", "service": "storage-engine"}

```

This script subscribes to MQTT topics from the ESP32, receives DHT sensor data, and saves it into an InfluxDB database. The terminal output confirms real-time data collection and successful database writes.

[illegible]

- The screenshot shows the InfluxDB Data Explorer interface. The top navigation bar displays the URL: `localhost:8086/orgs/6cf62e3c781f3ed8/data-explorer?fluxScriptEditor`. The main header is "Data Explorer". Below the header, there's a "Table" view selector and a "CUSTOMIZE" button. The table displays sensor data with columns for time, measurement, field, device, and location. The first column shows timestamps from 2025-05-18 02:00:05 to 03:00:05. The second column shows measurements like "humidity" and "temperature". The third column shows field names like "_field". The fourth column shows device names like "dht_data" and "esp32". The fifth column shows location names like "esp32". Below the table, there's a "Query 1 (0.01s)" section with a "VIEW RAW DATA" button and a "CSV" button. The "FROM" section shows the data source as "InfluxData" and "Lab13". The "Filter" section shows filters for "_measurement" (dht_data), "_field" (humidity, temperature), and "device" (esp32). The "WINDOW PERIOD" section shows "CUSTOM" and "AUTO" buttons. The "AGGREGATE FUNCTION" section shows "CUSTOM" and "AUTO" buttons.

The image shows a Jupyter Notebook interface with a code editor on the left and a preview of the output on the right. The code in the notebook generates a dataset with noise and evaluates a classification model. The output includes a confusion matrix and a classification report.

Confusion Matrix

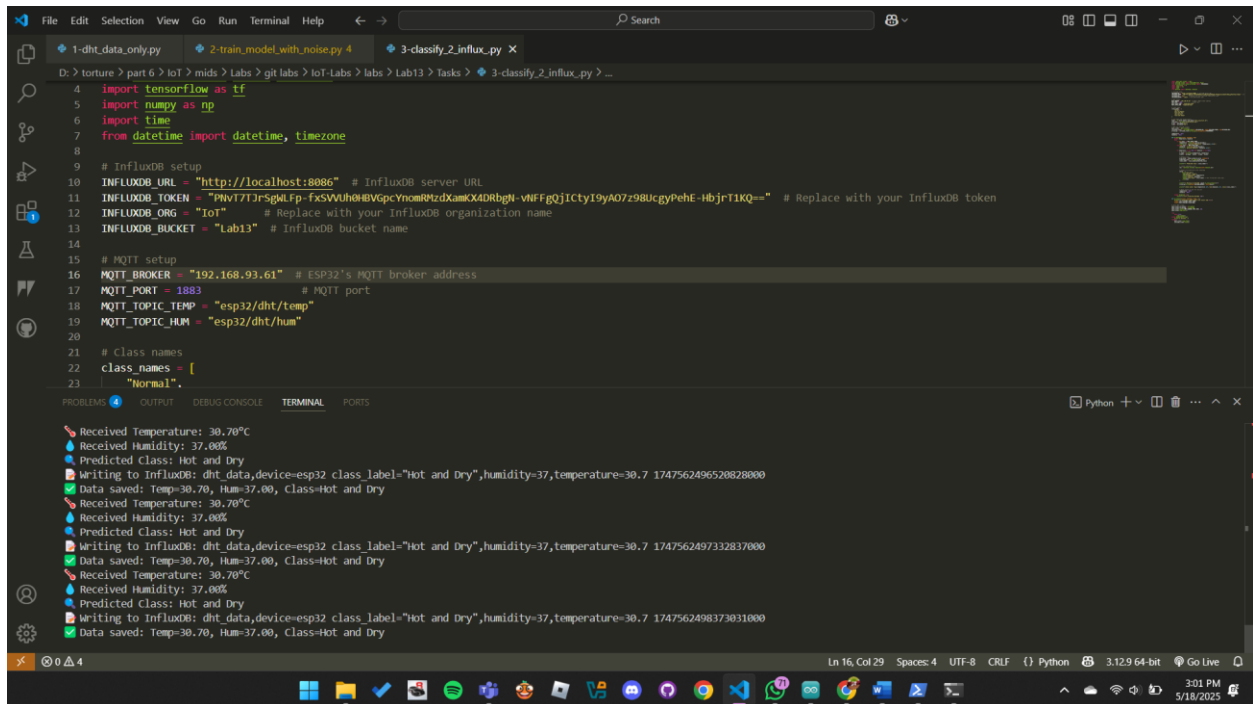
	Normal	HotHumid	ColdDry	HotDry	ColdHumid
Normal	777	1	54	0	0
HotHumid	0	742	0	0	0
Actual ColdDry	18	0	812	0	0
HotDry	0	0	0	771	0
ColdHumid	25	0	53	0	747
	Normal	HotHumid	ColdDry	HotDry	ColdHumid

Classification Report:

	precision	recall	f1-score	support
0	0.9476	0.9339	0.9407	832
1	0.9987	1.0000	0.9993	742
2	0.9836	0.9783	0.9809	830
3	1.0000	1.0000	1.0000	771
4	1.0000	0.9055	0.9504	825
accuracy	0.9660	0.9635	0.9638	4000
macro avg	0.9647	0.9623	0.9625	4000

This screenshot gives us a look at the real-time classification and logging system we've built using:

- An MQTT subscriber that listens to sensor data.
- A TensorFlow model to classify the environment.
- InfluxDB to store the classification results.



The screenshot shows a Visual Studio Code editor with a Python script named `3-classify_2_influx.py`. The script is configured to receive sensor data via MQTT, classify it using a TensorFlow model, and log the results to InfluxDB. The terminal output shows the script running successfully, receiving temperature and humidity data, predicting the class as "Hot and Dry", and writing the data to InfluxDB.

```
4 import tensorflow as tf
5 import numpy as np
6 import time
7 from datetime import datetime, timezone
8
9 # InfluxDB setup
10 INFLUXDB_URL = "http://localhost:8086" # InfluxDB server URL
11 INFLUXDB_TOKEN = "PNV17TJrSgWLFp-fxSVVUuHhHVGpcYnmRHzdXamCX4DRbghN-vNFFGqjICtyI9yA07z98UcgyPeHe-HbjrT1KQ==" # Replace with your InfluxDB token
12 INFLUXDB_ORG = "IoT" # Replace with your InfluxDB organization name
13 INFLUXDB_BUCKET = "Lab13" # InfluxDB bucket name
14
15 # MQTT setup
16 MQTT_BROKER = "192.168.93.61" # 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 names
22 class_names = [
23     "Normal",
24 ]
```

Terminal Output:

```
Received Temperature: 30.70°C
Received Humidity: 37.00%
Predicted Class: hot and Dry
Writing to InfluxDB: dht_data,device=esp32 class_label="hot and Dry",humidity=37,temperature=30.7 1747562496520828000
Data saved: Temp=30.70, Hum=37.00, Class=hot and Dry
Received Temperature: 30.70°C
Received Humidity: 37.00%
Predicted Class: Hot and Dry
Writing to InfluxDB: dht_data,device=esp32 class_label="Hot and Dry",humidity=37,temperature=30.7 1747562497332837000
Data saved: Temp=30.70, Hum=37.00, Class=Hot and Dry
Received Temperature: 30.70°C
Received Humidity: 37.00%
Predicted Class: hot and Dry
Writing to InfluxDB: dht_data,device=esp32 class_label="hot and Dry",humidity=37,temperature=30.7 1747562498373031000
Data saved: Temp=30.70, Hum=37.00, Class=hot and Dry
```