University of Moratuwa, Sri Lanka

Faculty of Engineering

Department of Electronics and Telecommunication Engineering
Semester 5 (Intake 2021)

EN3251 - Internet of Things

Information transfer with MQTT and HTTP using JSON

Gunawardana W.N.M.	210199D
Dilshan N.L.	210129P
Sehara G.M.M.	210583B

This report is submitted as a partial fulfillment for the moduleEN3251 - Internet of Things, Department of Electronic and Telecommunication Engineering, University of Moratuwa.

1 Step 3: MQTT Publisher and Subscriber with JSON

In **Step 3A**, the MQTT publisher reads a JSON object from a file and publishes it to a broker on a specific topic (`JACK1234`). The file is read using the `json.load()` function, and the data is serialized into a JSON string using `json.dumps()`. This encoded JSON object is then published to the MQTT broker. The network traffic can be captured using Wireshark to observe the MQTT publish operation.

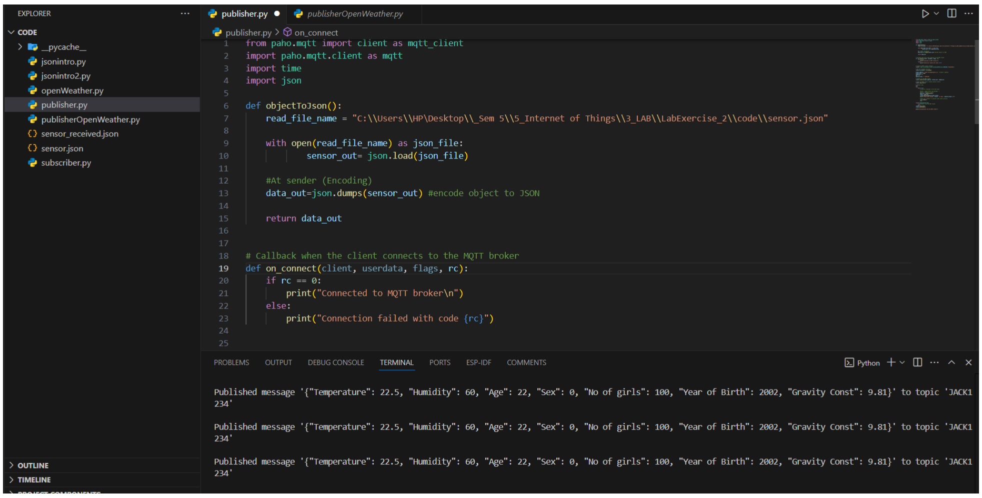
Key Observations

- **JSON Serialization**: The JSON object is serialized (converted to a string) before it is sent over MQTT. This step is crucial since MQTT messages require payloads to be transmitted in a binary format.
- **Wireshark Analysis**: In Wireshark, the MQTT publish packet contains the serialized JSON object. By expanding the packet, one can observe the actual message payload containing the human-readable JSON object.

```python
from paho.mqtt import client as mqtt_client
import paho.mqtt.client as mqtt
import time
import json

def objectToJson():
    read_file_name = "C:\\Users\\HP\Desktop\\_Sem 5\\5_Internet of Things\\3_LAB\\LabExercise_2\\code\\sensor.json"

    with open(read_file_name) as json_file:
            sensor_out= json.load(json_file)

    #At sender (Encoding)
    data_out=json.dumps(sensor_out) #encode object to JSON

    return data_out


# Callback when the client connects to the MQTT broker
def on_connect(client, userdata, flags, rc):
    if rc == 0:
        print("Connected to MQTT broker\n")
    else:
        print("Connection failed with code {rc}")


# Create an MQTT client instance
client = mqtt.Client(mqtt_client.CallbackAPIVersion.VERSION1,"PythonPub")

# Set the callback function
client.on_connect = on_connect

broker_address = "mqtt.eclipseprojects.io"  # broker's address
broker_port = 1883
keepalive = 5
qos = 0
```

```python
publish_topic = "JACK1234"

# Connect to the MQTT broker
client.connect(broker_address, broker_port, keepalive)

# Start the MQTT loop to handle network traffic
client.loop_start()

# Publish loop
n=0
try:
    while True:
        # Publish a message to the send topic

        #value = input(’Enter the message: ’)
        #value = "Hellow" + str(n)
        data_out = objectToJson()
        value = data_out
        client.publish(publish_topic,value)
        print(f"Published message ’{value}’ to topic ’{publish_topic}’\n")

        # Wait for a moment to simulate some client activity
        time.sleep(2)
        n+=1

except KeyboardInterrupt:
    # Disconnect from the MQTT broker
    pass
client.loop_stop()
client.disconnect()

print("Disconnected from the MQTT broker")
```

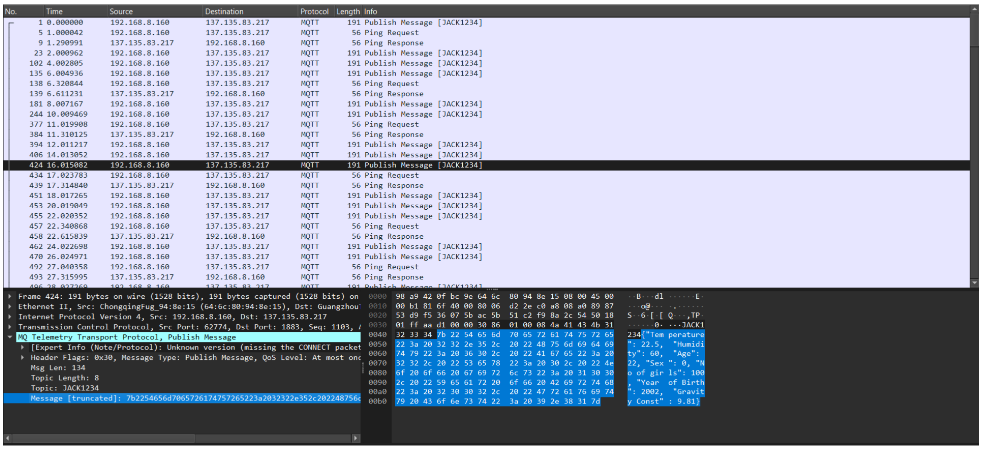
Listing 1: Publisher.py



```python
from paho.mqtt import client as mqtt_client
import paho.mqtt.client as mqtt
import time
import json

def objectToJson():
    read_file_name = "C:\\Users\\HP\Desktop\\_Sem 5\\5_Internet of Things\\3_LAB\\LabExercise_2\\code\\sensor.json"

    with open(read_file_name) as json_file:
            sensor_out= json.load(json_file)

    #At sender (Encoding)
    data_out=json.dumps(sensor_out) #encode object to JSON

    return data_out


# Callback when the client connects to the MQTT broker
def on_connect(client, userdata, flags, rc):
    if rc == 0:
        print("Connected to MQTT broker\n")
    else:
        print("Connection failed with code {rc}")
```

```
Published message '{"Temperature": 22.5, "Humidity": 60, "Age": 22, "Sex": 0, "No of girls": 100, "Year of Birth": 2002, "Gravity Const": 9.81}' to topic 'JACK1
234'

Published message '{"Temperature": 22.5, "Humidity": 60, "Age": 22, "Sex": 0, "No of girls": 100, "Year of Birth": 2002, "Gravity Const": 9.81}' to topic 'JACK1
234'

Published message '{"Temperature": 22.5, "Humidity": 60, "Age": 22, "Sex": 0, "No of girls": 100, "Year of Birth": 2002, "Gravity Const": 9.81}' to topic 'JACK1
234'
```

Figure 1: Python code and the output for publisher



```
No.   Time       Source           Destination      Protocol Length Info
    1 0.000000   192.168.8.160    137.135.83.217   MQTT     191 Publish Message [JACK1234]
    5 1.000042   192.168.8.160    137.135.83.217   MQTT      56 Ping Request
    9 1.290991   137.135.83.217   192.168.8.160    MQTT      56 Ping Response
   23 2.000962   192.168.8.160    137.135.83.217   MQTT     191 Publish Message [JACK1234]
  102 4.002805   192.168.8.160    137.135.83.217   MQTT     191 Publish Message [JACK1234]
  135 6.004936   192.168.8.160    137.135.83.217   MQTT     191 Publish Message [JACK1234]
  138 6.320844   192.168.8.160    137.135.83.217   MQTT      56 Ping Request
  139 6.611231   137.135.83.217   192.168.8.160    MQTT      56 Ping Response
  181 8.007167   192.168.8.160    137.135.83.217   MQTT     191 Publish Message [JACK1234]
  244 10.009469  192.168.8.160    137.135.83.217   MQTT     191 Publish Message [JACK1234]
  377 11.019908  192.168.8.160    137.135.83.217   MQTT      56 Ping Request
  384 11.310125  137.135.83.217   192.168.8.160    MQTT      56 Ping Response
  394 12.011217  192.168.8.160    137.135.83.217   MQTT     191 Publish Message [JACK1234]
  406 14.013052  192.168.8.160    137.135.83.217   MQTT     191 Publish Message [JACK1234]
  424 16.015082  192.168.8.160    137.135.83.217   MQTT     191 Publish Message [JACK1234]
  434 17.023783  192.168.8.160    137.135.83.217   MQTT      56 Ping Request
  439 17.314840  137.135.83.217   192.168.8.160    MQTT      56 Ping Response
  451 18.017265  192.168.8.160    137.135.83.217   MQTT     191 Publish Message [JACK1234]
  453 20.019049  192.168.8.160    137.135.83.217   MQTT     191 Publish Message [JACK1234]
  455 22.020352  192.168.8.160    137.135.83.217   MQTT     191 Publish Message [JACK1234]
  457 22.340868  192.168.8.160    137.135.83.217   MQTT      56 Ping Request
  458 22.615839  137.135.83.217   192.168.8.160    MQTT      56 Ping Response
  462 24.022698  192.168.8.160    137.135.83.217   MQTT     191 Publish Message [JACK1234]
  470 26.024971  192.168.8.160    137.135.83.217   MQTT     191 Publish Message [JACK1234]
  492 27.040358  192.168.8.160    137.135.83.217   MQTT      56 Ping Request
  493 27.315995  137.135.83.217   192.168.8.160    MQTT      56 Ping Response
  496 28.027269  192.168.8.160    137.135.83.217   MQTT     191 Publish Message [JACK1234]
```

```
> Frame 424: 191 bytes on wire (1528 bits), 191 bytes captured (1528 bits) on
> Ethernet II, Src: ChongqingFug_94:8e:15 (64:6c:80:94:8e:15), Dst: Guangzhou
> Internet Protocol Version 4, Src: 192.168.8.160, Dst: 137.135.83.217
> Transmission Control Protocol, Src Port: 62774, Dst Port: 1883, Seq: 1103, A
v MQ Telemetry Transport Protocol, Publish Message
  > [Expert Info (Note/Protocol): Unknown version (missing the CONNECT packet
  > Header Flags: 0x30, Message Type: Publish Message, QoS Level: At most onc
    Msg Len: 134
    Topic Length: 8
    Topic: JACK1234
    Message [truncated]: 7b2254656d706572617475726522 3a2032322e352c202248756c
```

```
0000  98 a9 42 0f bc 9e 64 6c  80 94 8e 15 08 00 45 00   ··B···dl ······E·
0010  00 b1 81 6f 40 00 80 06  d2 2e c0 a8 08 a0 89 87   ···o@··· ·.······
0020  53 d9 f5 36 07 5b ac 5b  51 c2 f9 8a 2c 54 50 18   S··6·[·[ Q···,TP·
0030  01 ff aa d1 00 00 30 86  01 00 08 4a 41 43 4b 31   ······0· ···JACK1
0040  32 33 34 7b 22 54 65 6d  70 65 72 61 74 75 72 65   234{"Tem perature
0050  22 3a 20 32 32 2e 35 2c  20 22 48 75 6d 69 64 69   ": 22.5,  "Humidi
0060  74 79 22 3a 20 36 30 2c  20 22 41 67 65 22 3a 20   ty": 60,  "Age": 
0070  32 32 2c 20 22 53 65 78  22 3a 20 30 2c 20 22 4e   22, "Sex ": 0, "N
0080  6f 20 6f 66 20 67 69 72  6c 73 22 3a 20 31 30 30   o of gir ls": 100
0090  2c 20 22 59 65 61 72 20  6f 66 20 42 69 72 74 68   , "Year  of Birth
00a0  22 3a 20 32 30 30 32 2c  20 22 47 72 61 76 69 74   ": 2002,  "Gravit
00b0  79 20 43 6f 6e 73 74 22  3a 20 39 2e 38 31 7d      y Const" : 9.81}
```

Figure 2: Wireshark capture for the publisher

In **Step 3B**, the MQTT subscriber subscribes to the same topic (`JACK1234`) and receives the JSON object published in part A. Upon receiving the message, the payload is decoded using `json.loads()` to convert the JSON string back into a Python dictionary. The deserialized data is then written to a file using `json.dump()`.

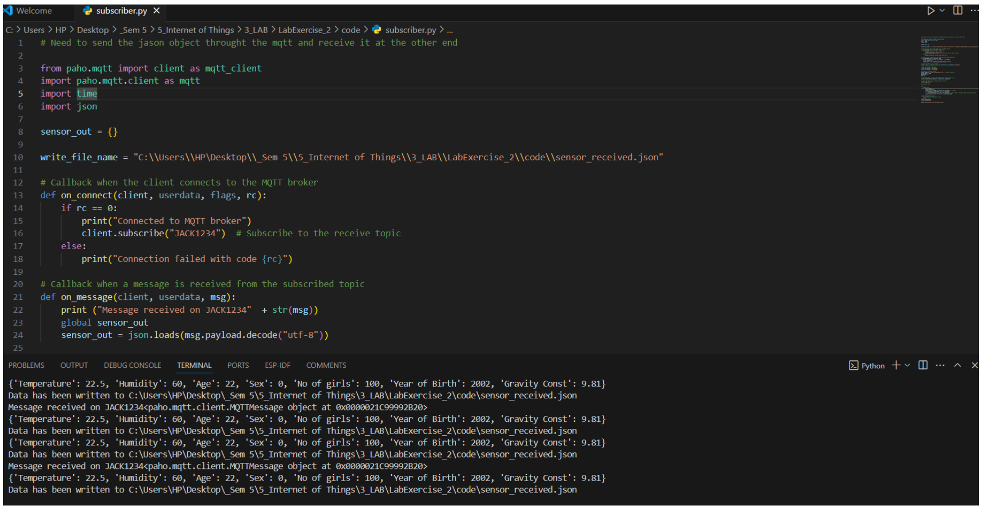
Key Observations

- **Deserialization**: The received JSON object is deserialized (converted back to a dictionary) to allow the subscriber to utilize the structured data.
- **Wireshark Observation**: Wireshark captures the MQTT subscribe packet and displays the received message containing the JSON object.

```
# Need to send the jason object throught the mqtt and receive it at the other end

from paho.mqtt import client as mqtt_client
import paho.mqtt.client as mqtt
import time
import json

sensor_out = {}

write_file_name = "C:\\Users\\HP\Desktop\\_Sem 5\\5_Internet of Things\\3_LAB\\LabExercise_2\\code\\sensor_received.json"

# Callback when the client connects to the MQTT broker
def on_connect(client, userdata, flags, rc):
    if rc == 0:
        print("Connected to MQTT broker")
        client.subscribe("JACK1234")  # Subscribe to the receive topic
    else:
        print("Connection failed with code {rc}")

# Callback when a message is received from the subscribed topic
def on_message(client, userdata, msg):
    print ("Message received on JACK1234"  + str(msg))
```

```python
    global sensor_out
    sensor_out = json.loads(msg.payload.decode("utf-8"))

# Create an MQTT client instance
client = mqtt.Client(mqtt_client.CallbackAPIVersion.VERSION1,"PythonSub")

# Set the callback functions
client.on_connect = on_connect
client.on_message = on_message

# Connect to the MQTT broker
broker_address = "mqtt.eclipseprojects.io"  # broker's address
broker_port = 1883
keepalive = 5
qos = 0

# subscribe_topic = input ('Enter the topic to subscribe to: ')
client.connect(broker_address, broker_port, keepalive)

# Start the MQTT loop to handle network traffic
client.loop_start()

# Subscribe loop

try:
    while True:
        time.sleep(1)
        print(sensor_out)
        sensor_in=json.loads(str(sensor_out).replace("'", "\""))
        with open(write_file_name, 'w') as json_file:
            json.dump(sensor_in, json_file, indent=4)  # The 'indent' parameter adds pretty formatting
        print("Data has been written to", write_file_name)

except KeyboardInterrupt:
    # Disconnect from the MQTT broker
    pass
client.loop_stop()
client.disconnect()

print("Disconnected from the MQTT broker")
```

Listing 2: Subscriber.py

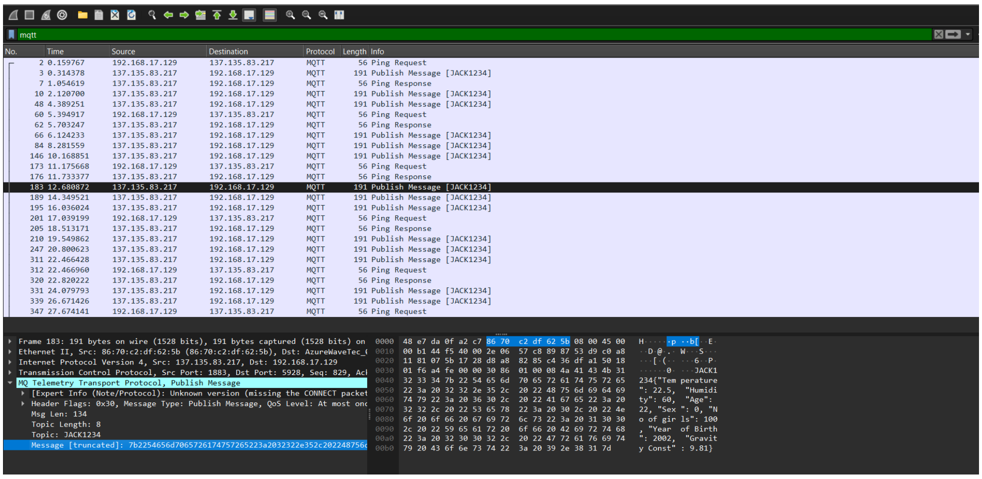


Welcome
subscriber.py
C: > Users > HP > Desktop > _Sem 5 > 5_Internet of Things > 3_LAB > LabExercise_2 > code > subscriber.py > ...

```python
# Need to send the jason object throught the mqtt and receive it at the other end

from paho.mqtt import client as mqtt_client
import paho.mqtt.client as mqtt
import time
import json

sensor_out = {}

write_file_name = "C:\\Users\\HP\Desktop\\_Sem 5\\5_Internet of Things\\3_LAB\\LabExercise_2\\code\\sensor_received.json"

# Callback when the client connects to the MQTT broker
def on_connect(client, userdata, flags, rc):
    if rc == 0:
        print("Connected to MQTT broker")
        client.subscribe("JACK1234")  # Subscribe to the receive topic
    else:
        print("Connection failed with code {rc}")

# Callback when a message is received from the subscribed topic
def on_message(client, userdata, msg):
    print ("Message received on JACK1234"  + str(msg))
    global sensor_out
    sensor_out = json.loads(msg.payload.decode("utf-8"))

```

```
{'Temperature': 22.5, 'Humidity': 60, 'Age': 22, 'Sex': 0, 'No of girls': 100, 'Year of Birth': 2002, 'Gravity Const': 9.81}
Data has been written to C:\Users\HP\Desktop\_Sem 5\5_Internet of Things\3_LAB\LabExercise_2\code\sensor_received.json
Message received on JACK1234<paho.mqtt.client.MQTTMessage object at 0x0000021C99992B20>
{'Temperature': 22.5, 'Humidity': 60, 'Age': 22, 'Sex': 0, 'No of girls': 100, 'Year of Birth': 2002, 'Gravity Const': 9.81}
Data has been written to C:\Users\HP\Desktop\_Sem 5\5_Internet of Things\3_LAB\LabExercise_2\code\sensor_received.json
{'Temperature': 22.5, 'Humidity': 60, 'Age': 22, 'Sex': 0, 'No of girls': 100, 'Year of Birth': 2002, 'Gravity Const': 9.81}
Data has been written to C:\Users\HP\Desktop\_Sem 5\5_Internet of Things\3_LAB\LabExercise_2\code\sensor_received.json
Message received on JACK1234<paho.mqtt.client.MQTTMessage object at 0x0000021C99992B20>
{'Temperature': 22.5, 'Humidity': 60, 'Age': 22, 'Sex': 0, 'No of girls': 100, 'Year of Birth': 2002, 'Gravity Const': 9.81}
Data has been written to C:\Users\HP\Desktop\_Sem 5\5_Internet of Things\3_LAB\LabExercise_2\code\sensor_received.json
```

Figure 3: Python code and the output for subscriber



```
No.     Time          Source            Destination       Protocol  Length Info
    2   0.159767      192.168.17.129    137.135.83.217    MQTT         56 Ping Request
    3   0.314378      137.135.83.217    192.168.17.129    MQTT        191 Publish Message [JACK1234]
    7   1.054619      137.135.83.217    192.168.17.129    MQTT         56 Ping Response
   10   2.120700      137.135.83.217    192.168.17.129    MQTT        191 Publish Message [JACK1234]
   48   4.389251      137.135.83.217    192.168.17.129    MQTT        191 Publish Message [JACK1234]
   60   5.394917      192.168.17.129    137.135.83.217    MQTT         56 Ping Request
   62   5.703247      137.135.83.217    192.168.17.129    MQTT         56 Ping Response
   66   6.124233      137.135.83.217    192.168.17.129    MQTT        191 Publish Message [JACK1234]
   84   8.281559      137.135.83.217    192.168.17.129    MQTT        191 Publish Message [JACK1234]
  146  10.168851      137.135.83.217    192.168.17.129    MQTT        191 Publish Message [JACK1234]
  173  11.175668      192.168.17.129    137.135.83.217    MQTT         56 Ping Request
  176  11.733377      137.135.83.217    192.168.17.129    MQTT         56 Ping Response
  183  12.680872      137.135.83.217    192.168.17.129    MQTT        191 Publish Message [JACK1234]
  189  14.349521      137.135.83.217    192.168.17.129    MQTT        191 Publish Message [JACK1234]
  195  16.036024      137.135.83.217    192.168.17.129    MQTT        191 Publish Message [JACK1234]
  201  17.039199      192.168.17.129    137.135.83.217    MQTT         56 Ping Request
  205  18.513171      137.135.83.217    192.168.17.129    MQTT         56 Ping Response
  210  19.549862      137.135.83.217    192.168.17.129    MQTT        191 Publish Message [JACK1234]
  247  20.800623      137.135.83.217    192.168.17.129    MQTT        191 Publish Message [JACK1234]
  311  22.466428      137.135.83.217    192.168.17.129    MQTT        191 Publish Message [JACK1234]
  312  22.466960      192.168.17.129    137.135.83.217    MQTT         56 Ping Request
  320  22.820222      137.135.83.217    192.168.17.129    MQTT         56 Ping Response
  331  24.079793      137.135.83.217    192.168.17.129    MQTT        191 Publish Message [JACK1234]
  339  26.671426      137.135.83.217    192.168.17.129    MQTT        191 Publish Message [JACK1234]
  347  27.674141      192.168.17.129    137.135.83.217    MQTT         56 Ping Request
```

```
> Frame 183: 191 bytes on wire (1528 bits), 191 bytes captured (1528 bits) on
> Ethernet II, Src: 86:70:c2:df:62:5b (86:70:c2:df:62:5b), Dst: AzureWaveTec_(
> Internet Protocol Version 4, Src: 137.135.83.217, Dst: 192.168.17.129
> Transmission Control Protocol, Src Port: 1883, Dst Port: 5928, Seq: 829, Ack
v MQ Telemetry Transport Protocol, Publish Message
  > [Expert Info (Note/Protocol): Unknown version (missing the CONNECT packet
  > Header Flags: 0x30, Message Type: Publish Message, QoS Level: At most onc
    Msg Len: 134
    Topic Length: 8
    Topic: JACK1234
    Message [truncated]: 7b2254656d70657261747572652223a2032322e352c202248756c
```

```
0000  48 e7 da 0f a2 c7 86 70  c2 df 62 5b 08 00 45 00   H·····p ··b[··E·
0010  00 b1 44 f5 40 00 2e 06  57 c8 89 87 53 d9 c0 a8   ··D·@·.· W···S···
0020  11 81 07 5b 17 28 d8 a8  82 85 c4 36 df a1 50 18   ···[·(·· ···6··P·
0030  01 f6 a4 fe 00 00 30 86  01 00 08 4a 41 43 4b 31   ······0· ···JACK1
0040  32 33 34 7b 22 54 65 6d  70 65 72 61 74 75 72 65   234{"Tem perature
0050  22 3a 20 32 32 2e 35 2c  20 22 48 75 6d 69 64 69   ": 22.5,  "Humidi
0060  74 79 22 3a 20 36 30 2c  20 22 41 67 65 22 3a 20   ty": 60,  "Age":
0070  32 32 2c 20 22 53 65 78  22 3a 20 30 2c 20 22 4e   22, "Sex ": 0, "N
0080  6f 20 6f 66 20 67 69 72  6c 73 22 3a 20 31 30 30   o of gir ls": 100
0090  2c 20 22 59 65 61 72 20  6f 66 20 42 69 72 74 68   , "Year  of Birth
00a0  22 3a 20 32 30 30 32 2c  20 22 47 72 61 76 69 74   ": 2002,  "Gravit
00b0  79 20 43 6f 6e 73 74 22  3a 20 39 2e 38 31 7d      y Const" : 9.81}
```

Figure 4: Wireshark capture for the subscriber

2 Step 4: HTTP Request to OpenWeather API

In **Step 4**, we retrieve real-time weather data from the OpenWeather API using an HTTP request. The city and country are provided by the user, and a request is made using the `requests` library. The response is received in JSON format, which contains information such as temperature, humidity, and weather description.

Key Points of Interest

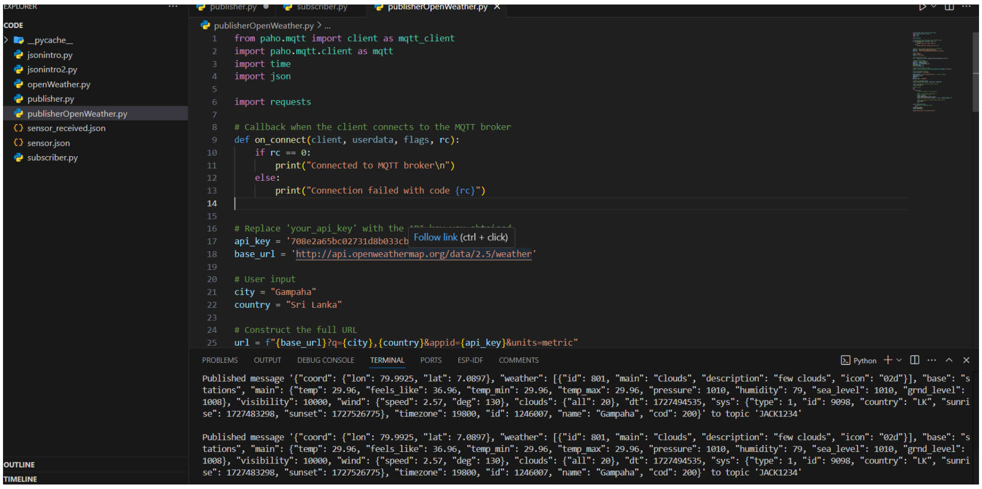
- **API Request**: The request to the OpenWeather API is made using the city and country as input parameters, and the response is returned in JSON format. The data is then parsed and displayed.

- **JSON Response**: The response from the API contains key-value pairs with details such as temperature, humidity, pressure, and general weather conditions.
- **MQTT Publishing**: Once the weather data is retrieved, it is serialized into a JSON string and published to the MQTT broker on topic `JACK1234`. This allows for easy dissemination of weather data across multiple MQTT clients.

```python
from paho.mqtt import client as mqtt_client
import paho.mqtt.client as mqtt
import time
import json

import requests

# Callback when the client connects to the MQTT broker
def on_connect(client, userdata, flags, rc):
    if rc == 0:
        print("Connected to MQTT broker\n")
    else:
        print("Connection failed with code {rc}")


# Replace 'your_api_key' with the API key you obtained
api_key = '708e2a65bc02731d8b033cb105a5382c'
base_url = 'http://api.openweathermap.org/data/2.5/weather'

# User input
city = "Gampaha"
country = "Sri Lanka"

# Construct the full URL
url = f"{base_url}?q={city},{country}&appid={api_key}&units=metric"

# Make the HTTP request
response = requests.get(url)
data_out = response.json()
data_out = json.dumps(data_out)
print(data_out)
print(type(data_out))

# Create an MQTT client instance
client = mqtt.Client(mqtt_client.CallbackAPIVersion.VERSION1,"PythonPub
    ")

# Set the callback function
client.on_connect = on_connect

broker_address = "mqtt.eclipseprojects.io"  # broker's address
broker_port = 1883
keepalive = 5
qos = 0
publish_topic = "JACK1234"

# Connect to the MQTT broker
client.connect(broker_address, broker_port, keepalive)

# Start the MQTT loop to handle network traffic
```

```python
client.loop_start()

# Publish loop
n=0
try:
    while True:
        # Publish a message to the send topic

        #value = input('Enter the message: ')
        #value = "Hellow" + str(n)
        value = data_out
        client.publish(publish_topic,value)
        print(f"Published message '{value}' to topic '{publish_topic}'\n")

        # Wait for a moment to simulate some client activity
        time.sleep(2)
        n+=1

except KeyboardInterrupt:
    # Disconnect from the MQTT broker
    pass
client.loop_stop()
client.disconnect()

print("Disconnected from the MQTT broker")
```

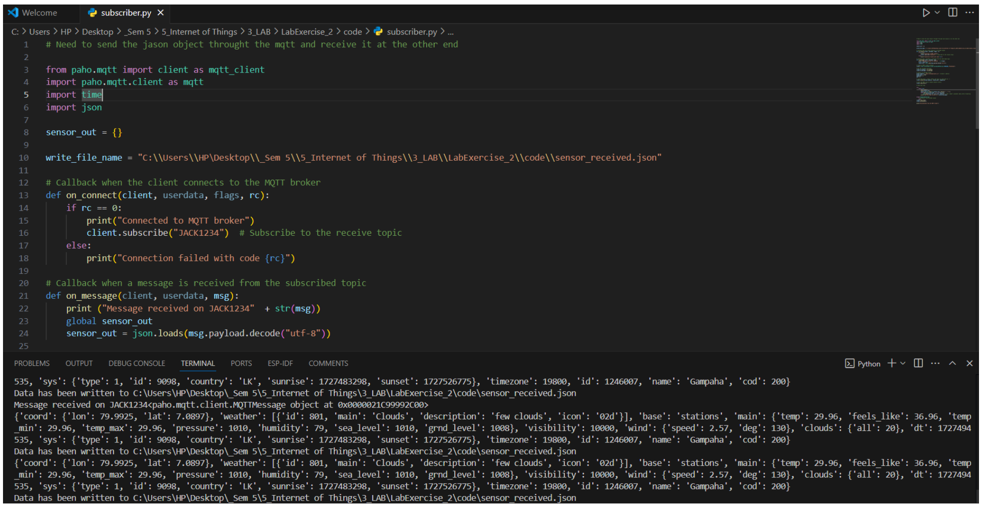
Listing 3: publisherOpenWeather.py



```python
from paho.mqtt import client as mqtt_client
import paho.mqtt.client as mqtt
import time
import json

import requests

# Callback when the client connects to the MQTT broker
def on_connect(client, userdata, flags, rc):
    if rc == 0:
        print("Connected to MQTT broker\n")
    else:
        print("Connection failed with code {rc}")


# Replace 'your_api_key' with the
api_key = '708e2a65bc02731d8b033cb
base_url = 'http://api.openweathermap.org/data/2.5/weather'

# User input
city = "Gampaha"
country = "Sri Lanka"

# Construct the full URL
url = f"{base_url}?q={city},{country}&appid={api_key}&units=metric"
```

```
Published message '{"coord": {"lon": 79.9925, "lat": 7.0897}, "weather": [{"id": 801, "main": "Clouds", "description": "few clouds", "icon": "02d"}], "base": "s
tations", "main": {"temp": 29.96, "feels_like": 36.96, "temp_min": 29.96, "temp_max": 29.96, "pressure": 1010, "humidity": 79, "sea_level": 1010, "grnd_level":
1008}, "visibility": 10000, "wind": {"speed": 2.57, "deg": 130}, "clouds": {"all": 20}, "dt": 1727494535, "sys": {"type": 1, "id": 9098, "country": "LK", "sunri
se": 1727483298, "sunset": 1727526775}, "timezone": 19800, "id": 1246007, "name": "Gampaha", "cod": 200}' to topic 'JACK1234'

Published message '{"coord": {"lon": 79.9925, "lat": 7.0897}, "weather": [{"id": 801, "main": "Clouds", "description": "few clouds", "icon": "02d"}], "base": "s
tations", "main": {"temp": 29.96, "feels_like": 36.96, "temp_min": 29.96, "temp_max": 29.96, "pressure": 1010, "humidity": 79, "sea_level": 1010, "grnd_level":
1008}, "visibility": 10000, "wind": {"speed": 2.57, "deg": 130}, "clouds": {"all": 20}, "dt": 1727494535, "sys": {"type": 1, "id": 9098, "country": "LK", "sunri
se": 1727483298, "sunset": 1727526775}, "timezone": 19800, "id": 1246007, "name": "Gampaha", "cod": 200}' to topic 'JACK1234'
```

Figure 5: Python code and the output for openWeatherPublisher

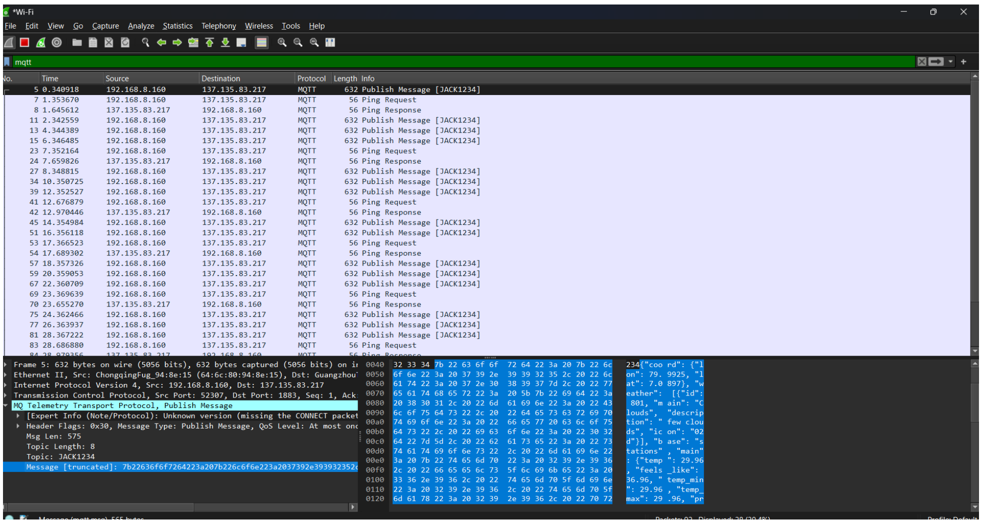


```python
# Need to send the jason object throught the mqtt and receive it at the other end

from paho.mqtt import client as mqtt_client
import paho.mqtt.client as mqtt
import time
import json

sensor_out = {}

write_file_name = "C:\\Users\\HP\Desktop\\_Sem 5\\5_Internet of Things\\3_LAB\\LabExercise_2\\code\\sensor_received.json"

# Callback when the client connects to the MQTT broker
def on_connect(client, userdata, flags, rc):
    if rc == 0:
        print("Connected to MQTT broker")
        client.subscribe("JACK1234")  # Subscribe to the receive topic
    else:
        print("Connection failed with code {rc}")

# Callback when a message is received from the subscribed topic
def on_message(client, userdata, msg):
    print ("Message received on JACK1234"  + str(msg))
    global sensor_out
    sensor_out = json.loads(msg.payload.decode("utf-8"))

```

```
535, 'sys': {'type': 1, 'id': 9098, 'country': 'LK', 'sunrise': 1727483298, 'sunset': 1727526775}, 'timezone': 19800, 'id': 1246007, 'name': 'Gampaha', 'cod': 200}
Data has been written to C:\Users\HP\Desktop\_Sem 5\5_Internet of Things\3_LAB\LabExercise_2\code\sensor_received.json
Message received on JACK1234<paho.mqtt.client.MQTTMessage object at 0x0000021C99992C00>
{'coord': {'lon': 79.9925, 'lat': 7.0897}, 'weather': [{'id': 801, 'main': 'Clouds', 'description': 'few clouds', 'icon': '02d'}], 'base': 'stations', 'main': {'temp': 29.96, 'feels_like': 36.96, 'temp
_min': 29.96, 'temp_max': 29.96, 'pressure': 1010, 'humidity': 79, 'sea_level': 1010, 'grnd_level': 1008}, 'visibility': 10000, 'wind': {'speed': 2.57, 'deg': 130}, 'clouds': {'all': 20}, 'dt': 1727494
535, 'sys': {'type': 1, 'id': 9098, 'country': 'LK', 'sunrise': 1727483298, 'sunset': 1727526775}, 'timezone': 19800, 'id': 1246007, 'name': 'Gampaha', 'cod': 200}
Data has been written to C:\Users\HP\Desktop\_Sem 5\5_Internet of Things\3_LAB\LabExercise_2\code\sensor_received.json
{'coord': {'lon': 79.9925, 'lat': 7.0897}, 'weather': [{'id': 801, 'main': 'Clouds', 'description': 'few clouds', 'icon': '02d'}], 'base': 'stations', 'main': {'temp': 29.96, 'feels_like': 36.96, 'temp
_min': 29.96, 'temp_max': 29.96, 'pressure': 1010, 'humidity': 79, 'sea_level': 1010, 'grnd_level': 1008}, 'visibility': 10000, 'wind': {'speed': 2.57, 'deg': 130}, 'clouds': {'all': 20}, 'dt': 1727494
535, 'sys': {'type': 1, 'id': 9098, 'country': 'LK', 'sunrise': 1727483298, 'sunset': 1727526775}, 'timezone': 19800, 'id': 1246007, 'name': 'Gampaha', 'cod': 200}
Data has been written to C:\Users\HP\Desktop\_Sem 5\5_Internet of Things\3_LAB\LabExercise_2\code\sensor_received.json
```

Welcome
subscriber.py
C: > Users > HP > Desktop > _Sem 5 > 5_Internet of Things > 3_LAB > LabExercise_2 > code > subscriber.py > ...
PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
PORTS
ESP-IDF
COMMENTS
Python

Figure 6: Python code and the output for subscriber(Use same Subscriber.py code)



```
No.  Time       Source          Destination     Protocol Length Info
  5 0.340918    192.168.8.160   137.135.83.217  MQTT     632 Publish Message [JACK1234]
  7 1.353670    192.168.8.160   137.135.83.217  MQTT      56 Ping Request
  8 1.645612    137.135.83.217  192.168.8.160   MQTT      56 Ping Response
 11 2.342559    192.168.8.160   137.135.83.217  MQTT     632 Publish Message [JACK1234]
 13 4.344389    192.168.8.160   137.135.83.217  MQTT     632 Publish Message [JACK1234]
 15 6.346485    192.168.8.160   137.135.83.217  MQTT     632 Publish Message [JACK1234]
 23 7.352164    192.168.8.160   137.135.83.217  MQTT      56 Ping Request
 24 7.659826    137.135.83.217  192.168.8.160   MQTT      56 Ping Response
 27 8.348815    192.168.8.160   137.135.83.217  MQTT     632 Publish Message [JACK1234]
 34 10.350725   192.168.8.160   137.135.83.217  MQTT     632 Publish Message [JACK1234]
 39 12.352527   192.168.8.160   137.135.83.217  MQTT     632 Publish Message [JACK1234]
 41 12.676879   192.168.8.160   137.135.83.217  MQTT      56 Ping Request
 42 12.970446   137.135.83.217  192.168.8.160   MQTT      56 Ping Response
 45 14.354984   192.168.8.160   137.135.83.217  MQTT     632 Publish Message [JACK1234]
 51 16.356118   192.168.8.160   137.135.83.217  MQTT     632 Publish Message [JACK1234]
 53 17.366523   192.168.8.160   137.135.83.217  MQTT      56 Ping Request
 54 17.689302   137.135.83.217  192.168.8.160   MQTT      56 Ping Response
 57 18.357326   192.168.8.160   137.135.83.217  MQTT     632 Publish Message [JACK1234]
 59 20.359053   192.168.8.160   137.135.83.217  MQTT     632 Publish Message [JACK1234]
 67 22.360709   192.168.8.160   137.135.83.217  MQTT     632 Publish Message [JACK1234]
 69 23.369639   192.168.8.160   137.135.83.217  MQTT      56 Ping Request
 70 23.655270   137.135.83.217  192.168.8.160   MQTT      56 Ping Response
 75 24.362466   192.168.8.160   137.135.83.217  MQTT     632 Publish Message [JACK1234]
 77 26.363937   192.168.8.160   137.135.83.217  MQTT     632 Publish Message [JACK1234]
 81 28.367222   192.168.8.160   137.135.83.217  MQTT     632 Publish Message [JACK1234]
 83 28.686880   192.168.8.160   137.135.83.217  MQTT      56 Ping Request
 84 28.979356   137.135.83.217  192.168.8.160   MQTT      56 Ping Response
```

```
Frame 5: 632 bytes on wire (5056 bits), 632 bytes captured (5056 bits) on ir
Ethernet II, Src: ChongqingFug_94:8e:15 (64:6c:80:94:8e:15), Dst: GuangzhouT
Internet Protocol Version 4, Src: 192.168.8.160, Dst: 137.135.83.217
Transmission Control Protocol, Src Port: 52307, Dst Port: 1883, Seq: 1, Ack:
MQ Telemetry Transport Protocol, Publish Message
    [Expert Info (Note/Protocol): Unknown version (missing the CONNECT packet
    Header Flags: 0x30, Message Type: Publish Message, QoS Level: At most onc
    Msg Len: 575
    Topic Length: 8
    Topic: JACK1234
    Message [truncated]: 7b22636f6f7264223a207b226c6f6e223a2037392e393932352c
```

```
0040  32 33 34 7b 22 63 6f 6f  72 64 22 3a 20 7b 22 6c   234{"coo rd": {"l
0050  6f 6e 22 3a 20 37 39 2e  39 39 32 35 2c 20 22 6c   on": 79. 9925, "l
0060  61 74 22 3a 20 37 2e 30  38 39 37 7d 2c 20 22 77   at": 7.0 897}, "w
0070  65 61 74 68 65 72 22 3a  20 5b 7b 22 69 64 22 3a   eather":  [{"id":
0080  20 38 30 31 2c 20 22 6d  61 69 6e 22 3a 20 22 43    801, "m ain": "C
0090  6c 6f 75 64 73 22 2c 20  22 64 65 73 63 72 69 70   louds",  "descrip
00a0  74 69 6f 6e 22 3a 20 22  66 65 77 20 63 6c 6f 75   tion": " few clou
00b0  64 73 22 2c 20 22 69 63  6f 6e 22 3a 20 22 30 32   ds", "ic on": "02
00c0  64 22 7d 5d 2c 20 22 62  61 73 65 22 3a 20 22 73   d"}], "b ase": "s
00d0  74 61 74 69 6f 6e 73 22  2c 20 22 6d 61 69 6e 22   tations" , "main"
00e0  3a 20 7b 22 74 65 6d 70  22 3a 20 32 39 2e 39 36   : {"temp ": 29.96
00f0  2c 20 22 66 65 65 6c 73  5f 6c 69 6b 65 22 3a 20   , "feels _like": 
0100  33 36 2e 39 36 2c 20 22  74 65 6d 70 5f 6d 69 6e   36.96, " temp_min
0110  22 3a 20 32 39 2e 39 36  2c 20 22 74 65 6d 70 5f   ": 29.96 , "temp_
0120  6d 61 78 22 3a 20 32 39  2e 39 36 2c 20 22 70 72   max": 29 .96, "pr
```

Figure 7: Wireshark capture for the openWeatherPublisher

3 Node-Red flow

3.1 Using MQTT Broker

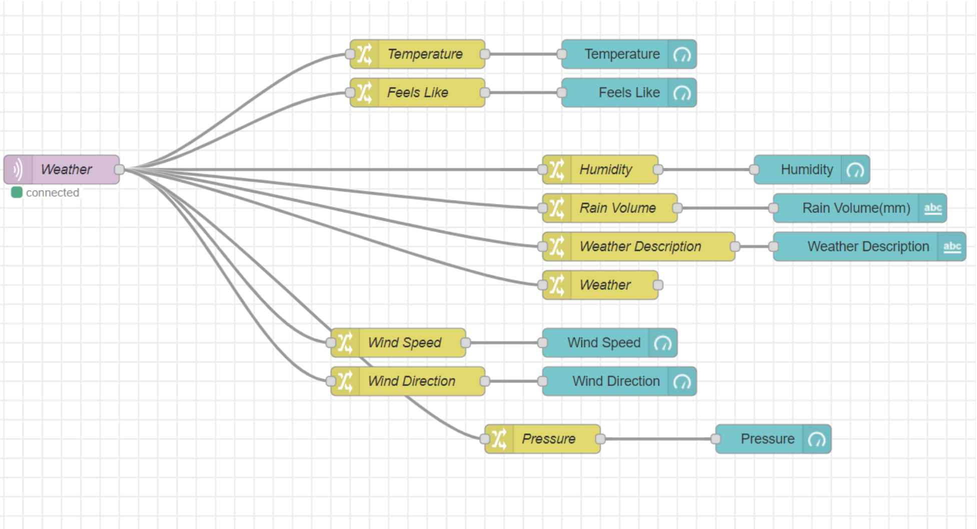


Figure 8: Node-Red flow

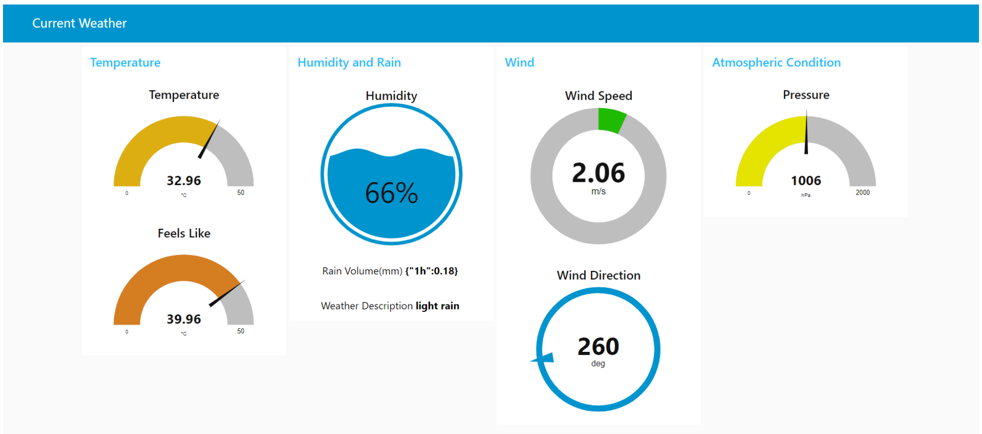


Figure 9: Node-Red dashboard

3.2 Simply Using OpenWeather

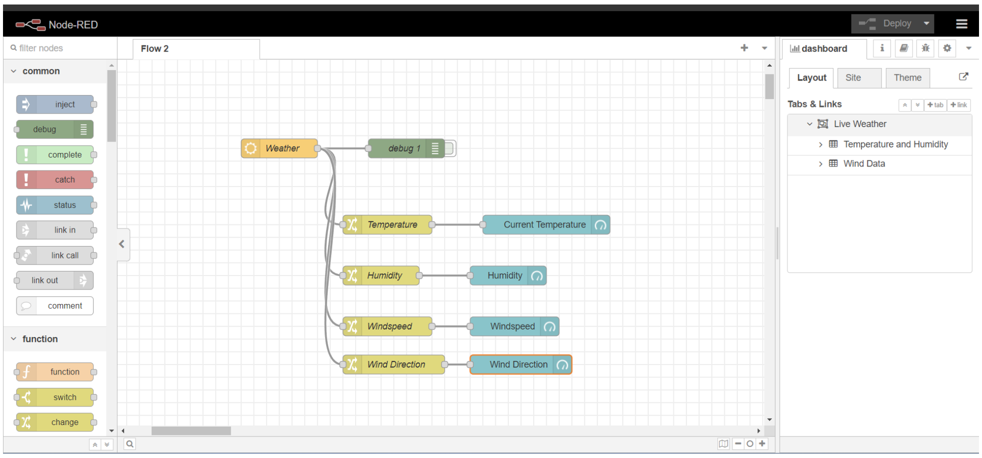


Figure 10: Node-Red flow

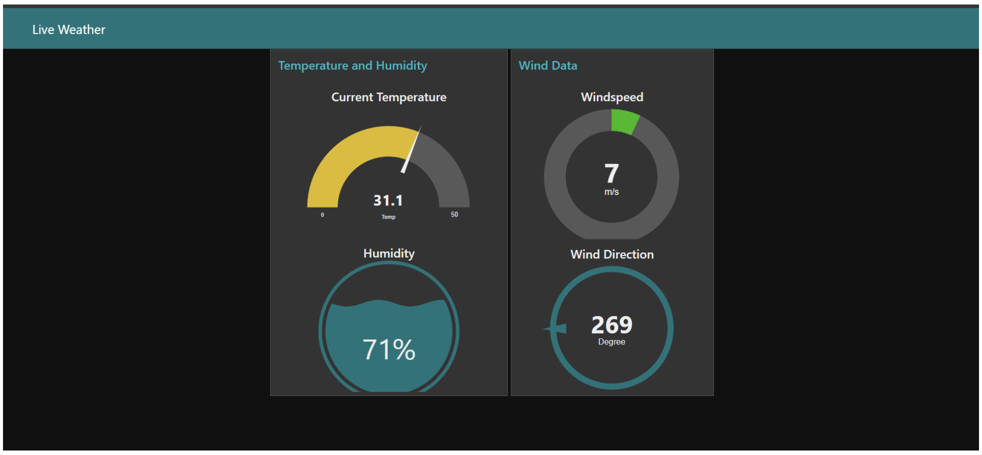


Figure 11: Node-Red dashboard