Internet Of Things

TP 1:Use of the MQTT Protocol

Group members:

Student Number	Name
a57096	Dibya Raj Khatri
a43323	Bikal Bista

Worksheet 1

Objectives : Sending Messages | Eclipse Paho | Node-RED

Introduction:

Message Queuing Telemetry Transport (MQTT) is a lightweight IoT device message-exchange protocol. Unlike client-server protocols, which require direct communication, MQTT relies on a publish-subscribe model. The publish-subscribe model enables devices to communicate via a broker, hence enabling efficient message delivery with minimal bandwidth.

The main features of MQTT are:

- Scalability : Suitable for big IoT networks.
- Lightweight Messaging : Low overhead of header.
- Reliable Communication : Offers message delivery in unreliable networks.
- Security Features : Offers authentication and encryption support.

More details are to be found at [MQTT official website](https://mqtt.org/).

MQTT Protocol in Practice

1. Use of Python Programming and Google Colab

2. What is Eclipse Paho?

Eclipse Paho is an open-source implementation of the MQTT protocol, providing client libraries for several programming languages such as Python, Java, C, and JavaScript. These libraries help developers implement MQTT communication in IoT applications.

3. Installing Eclipse Paho

To install the Eclipse Paho library in Python, run the following command:
!pip install paho-mqtt

4. Connecting to an MQTT Broker and Subscribing to a Topic

The following code subscribes to a topic and connects to a broker:
We used this code to subscribe to the Professor's laptop .

```
import paho.mqtt.client as mqtt
import time
def on_connect(client, userdata, flags, rc):
   print("Connected with result code "+str(rc))
```

```
    client.subscribe("IPB/IoT/Class01/ShiftA")

def on_message(client, userdata, msg):
    msg.payload=msg.payload.decode("utf-8")
    print(msg.topic+" MSG: "+str(msg.payload))
client = mqtt.Client()
client.on_connect = on_connect
client.on_message = on_message
client.connect('broker.mqtt-dashboard.com', 1883, 60)
```

5. Publishing Messages to a Topic

The following code can be used for publishing a message:

```
import paho.mqtt.client as mqtt
import time

BROKER = 'broker.mqtt-dashboard.com'
PORT = 1883
TOPIC = "IPB/IoT/Class02/ShiftA/STUDENT57096"

def on_connect(client, userdata, flags, rc):
    print("Connected with result code " + str(rc))

client = mqtt.Client()
client.on_connect = on_connect

client.connect(BROKER, PORT, 60)

client.loop_start()

try:
    while True:
        name = "Dibya raj khatri"
        num = "a57096"
        message = "Name: " + name + ", Num: " + num
        client.publish(TOPIC, message, qos=0, retain=False)
        print("Published:", message)
        time.sleep(5)
except KeyboardInterrupt:
    print("\nPublisher stopped by user.")
finally:
```

```
client.loop_stop()
```

6. Checking the Received Message

The subscriber should print out the message received when the publisher publishes information to the topic.

Exercises - Part I (Exchanging Messages)

7. Description of MQTT Components

Publisher:
An application or device that sends messages to a specific topic in the MQTT broker.

Subscriber:
An application or device that listens for messages from a specific topic in the MQTT broker.

Broker:
Server in the middle that handles the delivering of messages between publishers and subscribers. It delivers messages and serves multiple clients efficiently.

Topic:
Virtual pipe through which messages are being passed and received. Subscribers subscribe to topics so that they receive related messages.

8. Finding Faults in the Provided Code

There are some errors in the code presented below, which do not enable correct exchange of messages:

Faults in the Subscriber Code:
1. Incorrect indentation in the function definitions

- ‘def on_connect(client, userdata, flags, rc)’: is correctly defined, but ‘on_message’ contains an extra underscore (`on _message`) in the function name.
  - This will lead to an error when the line ‘client.on_message = on_message’ is run.

2. Client Initialization Issue

- Client object ‘mqtt.Client()’ is missing parentheses in the publisher code.
- Correct syntax: ‘client = mqtt.Client()’.

1. Incorrect Client Initialization Syntax

- ‘client = mqtt.Client)’ is incorrect.
- It should be ‘client = mqtt.Client()’.

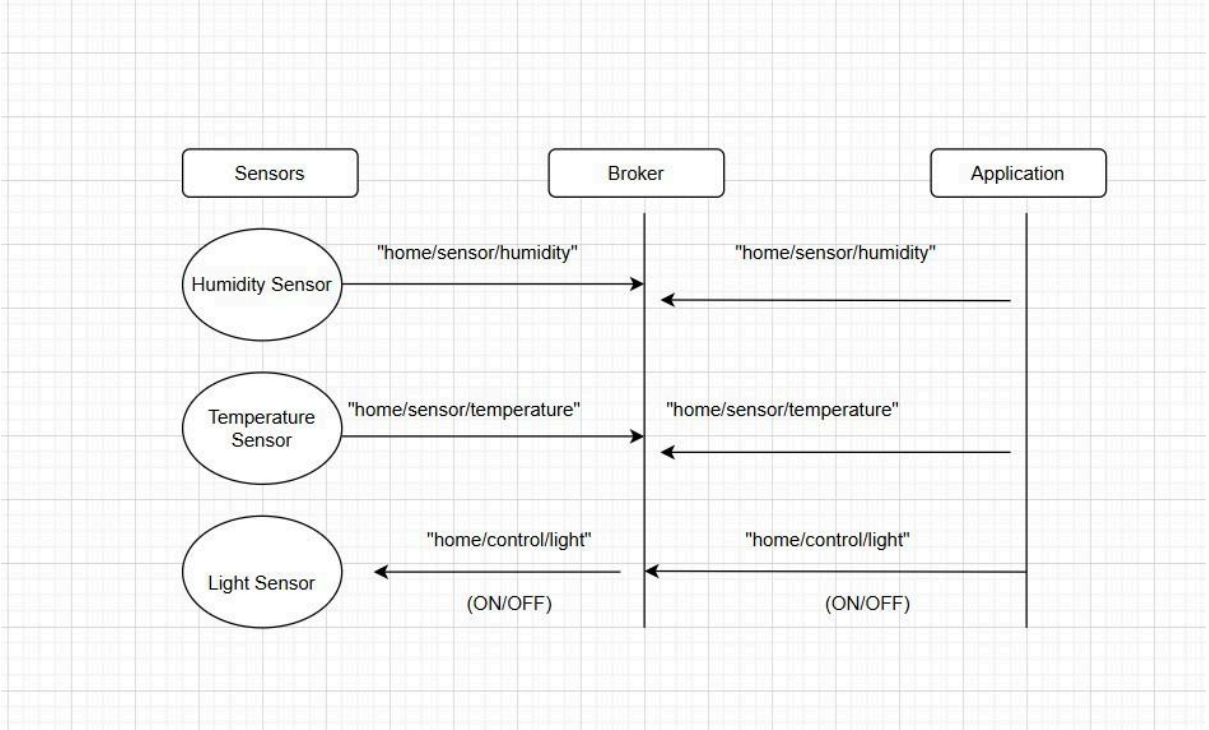
2. Different Topics Used for Publishing and Subscribing

- The subscriber listens to ‘topic_test_A’, while the publisher publishes data to ‘topic_test_B’.
- Both should utilize the same topic for successful communication.

Corrected Code for Publisher:

```
import paho.mqtt.client as mqtt

client = mqtt.Client()
client.connect("broker.mqtt-dashboard.com", 1883, 60)
client.publish("topic_test_A", "Hello World")

print("Message Published")
```

10.The system is implemented using the Python Paho MQTT library. The following steps outline the process:



10.

Publisher (Sensor Node)

```
import paho.mqtt.client as mqtt
import time

# MQTT Broker details
BROKER = 'broker.mqtt-dashboard.com'
PORT = 1883
TOPIC_TEMP = "IPB/IoT/Class02/ShiftA/Temperature"
TOPIC_HUM = "IPB/IoT/Class02/ShiftA/Humidity"
TOPIC_LIGHT = "IPB/IoT/Class02/ShiftA/LightControl"

# MQTT Client Setup
client = mqtt.Client()

def on_connect(client, userdata, flags, rc):
    print("Connected with result code " + str(rc))

client.on_connect = on_connect
client.connect(BROKER, PORT, 60)
```

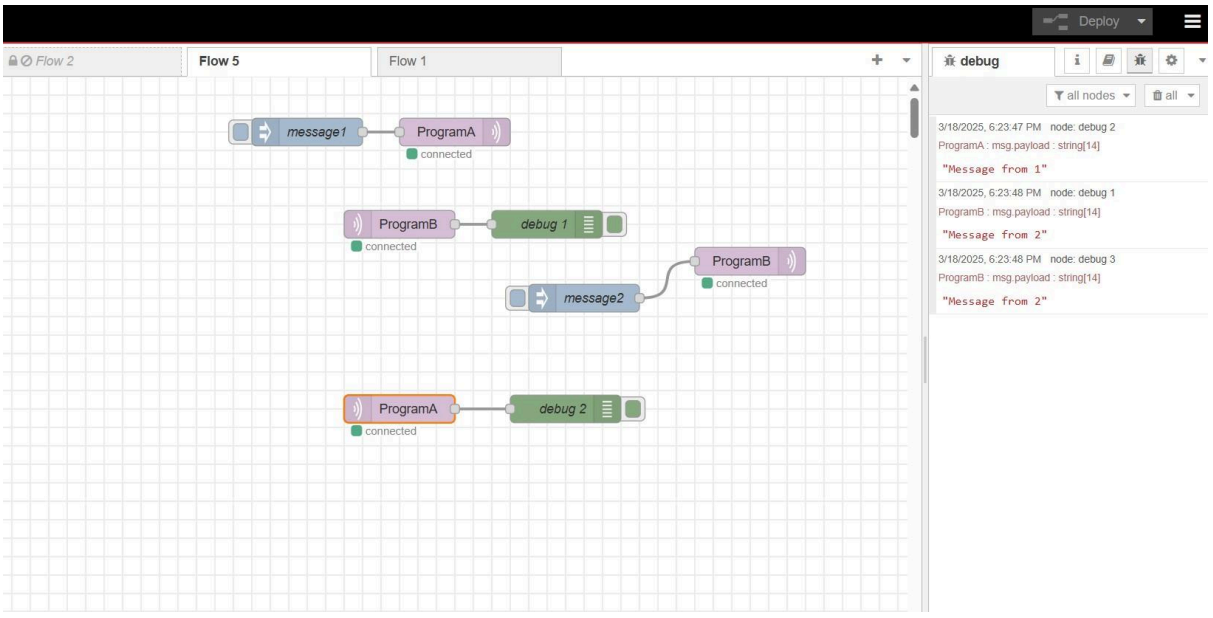
```
while True:
    temp = "22.5C"
    hum = "60%"
    light_status = "on"

    msg_temp = "{temperature: " + str(temp) + "}"
    msg_hum = "{humidity: " + str(hum) + "}"
    msg_light = "{light: " + str(light_status) + "}"

    client.publish(TOPIC_TEMP, msg_temp, qos=0, retain=False)
    client.publish(TOPIC_HUM, msg_hum, qos=0, retain=False)
    client.publish(TOPIC_LIGHT, msg_light, qos=0, retain=False)

    print("Published: ", msg_temp, msg_hum, msg_light)
    time.sleep(5)
```

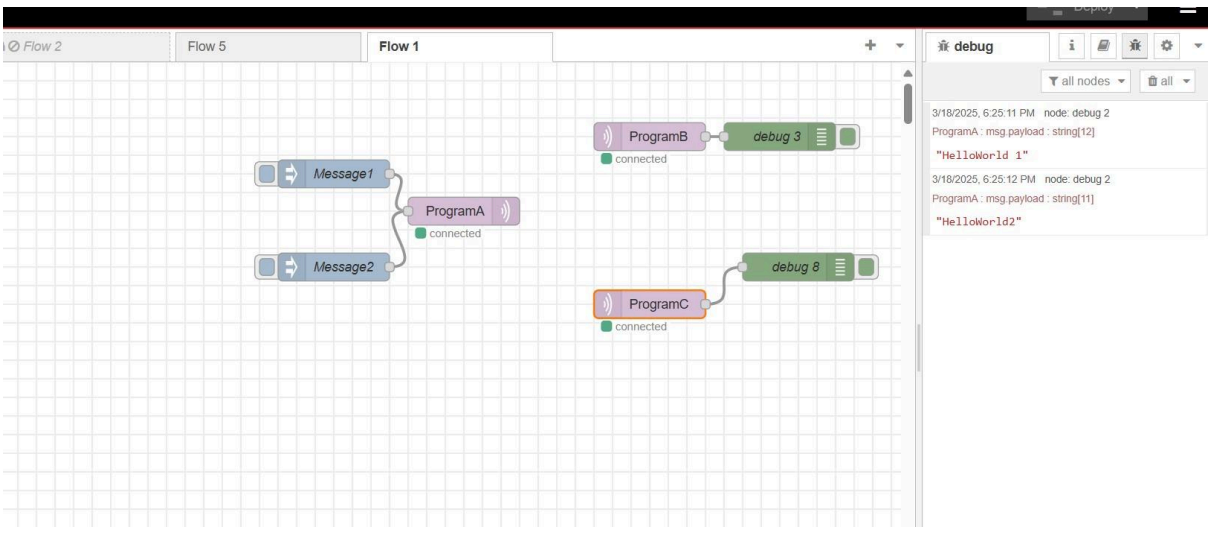
.Subscriber (Light Controller)

```
import paho.mqtt.client as mqtt

# MQTT Broker details
BROKER = 'broker.mqtt-dashboard.com'
PORT = 1883
TOPIC_LIGHT = "IPB/IoT/Class02/ShiftA/LightControl"

# Callback function when a message is received
def on_message(client, userdata, message):
    print(f"Light Control Command Received:
{message.payload.decode()}")

# MQTT Client Setup
client = mqtt.Client()

def on_connect(client, userdata, flags, rc):
    print("Connected with result code " + str(rc))
    client.subscribe(TOPIC_LIGHT)  # Subscribe after connecting
```

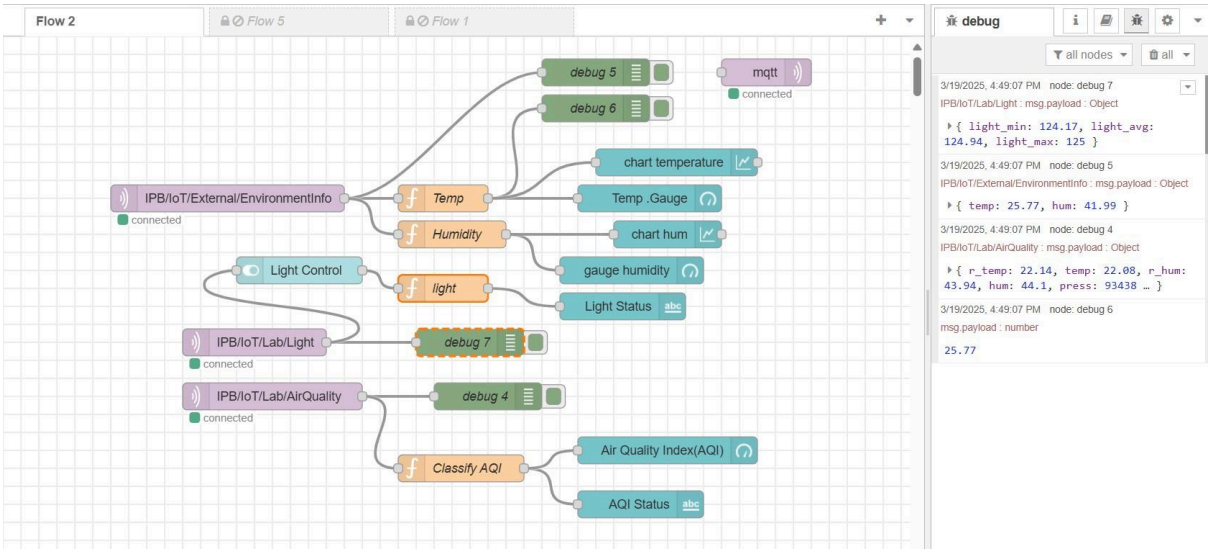
```
client.on_connect = on_connect
client.on_message = on_message
client.connect(BROKER, PORT, 60)

print("Listening for light control commands...")

# Start loop to keep listening
client.loop_forever()
```

12. Implement exercise 9 using Node-RED.

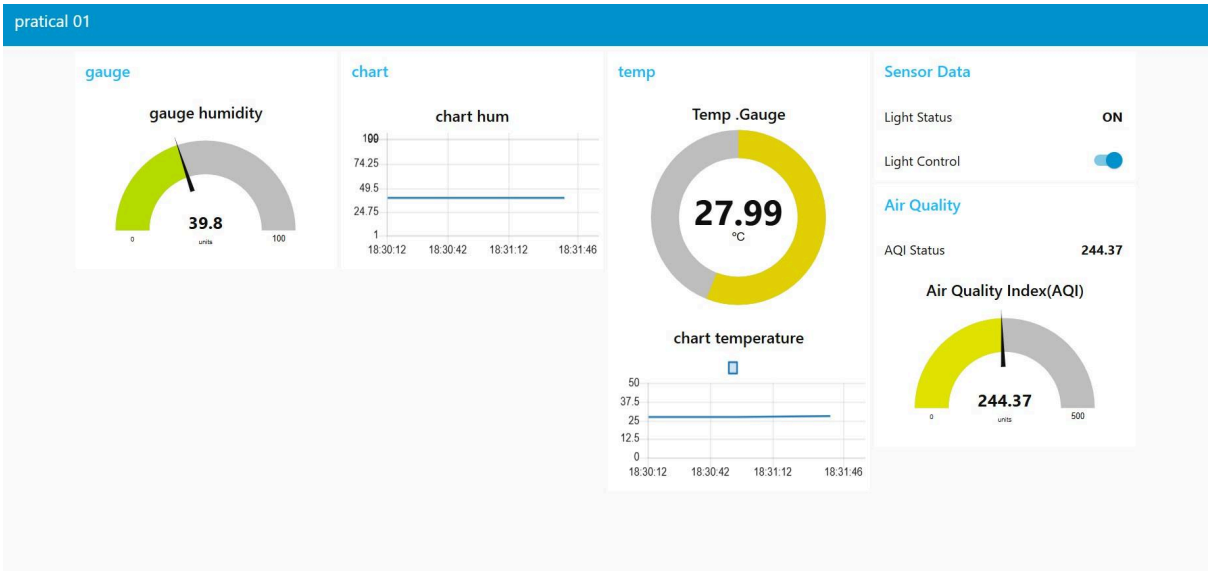




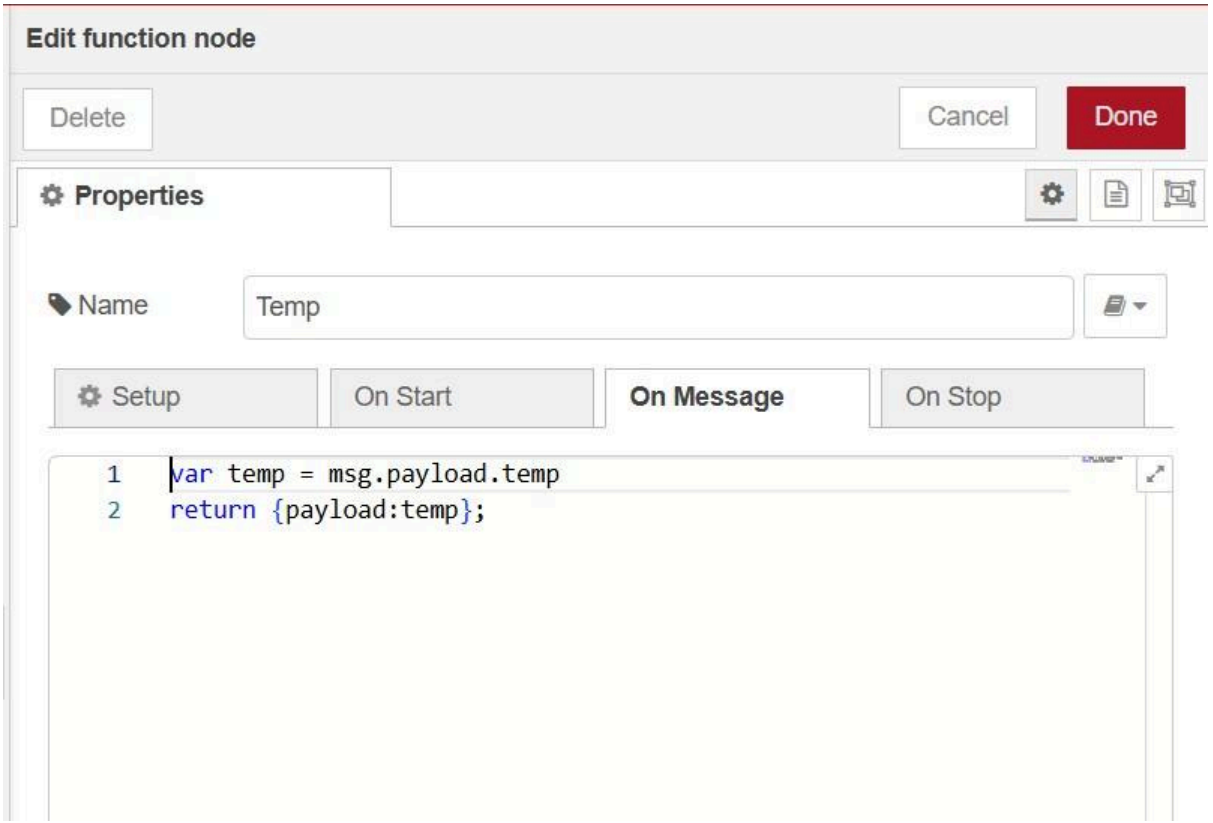
13. Implement exercise 10 using Node-RED



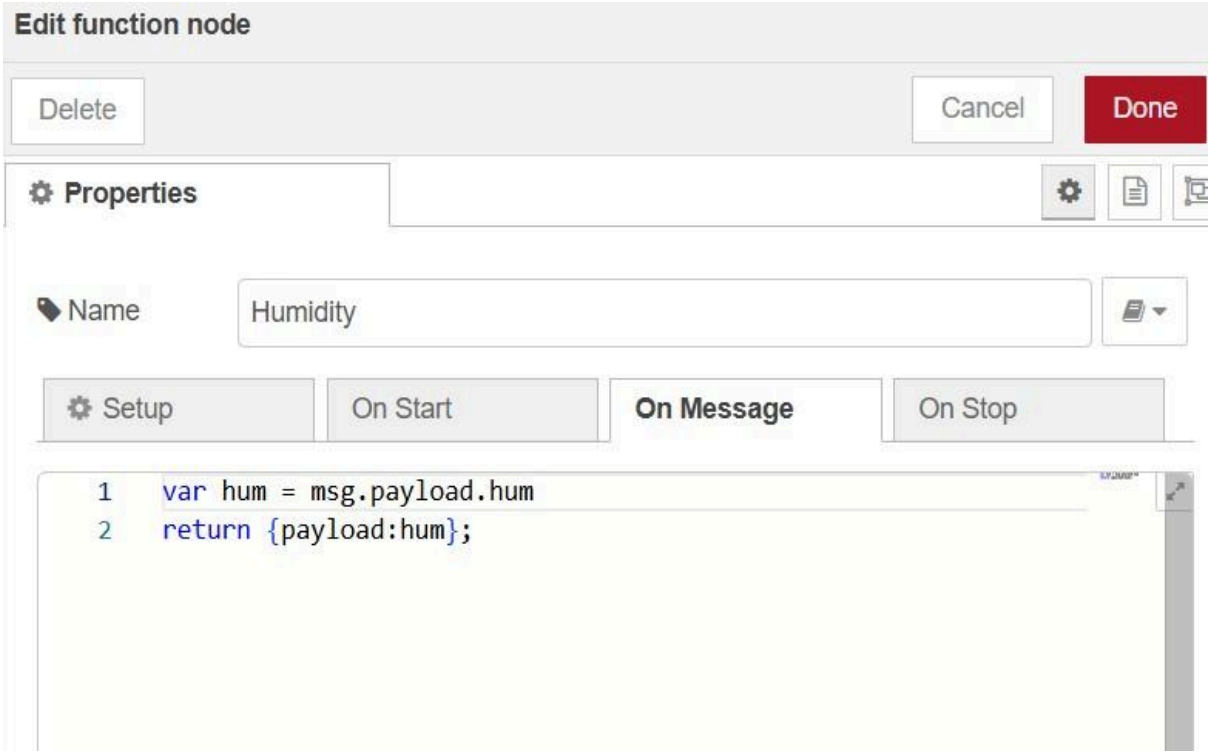
DashBoard



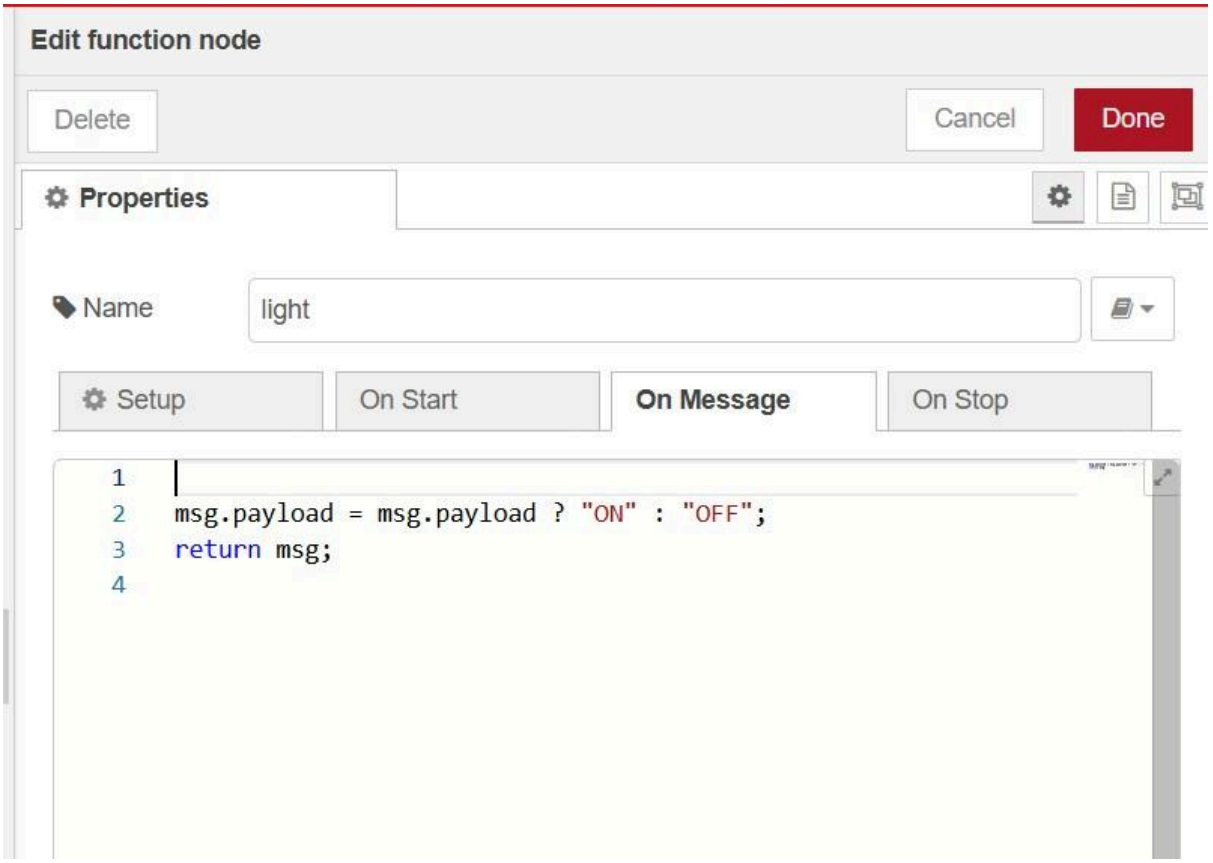
Function-Temp



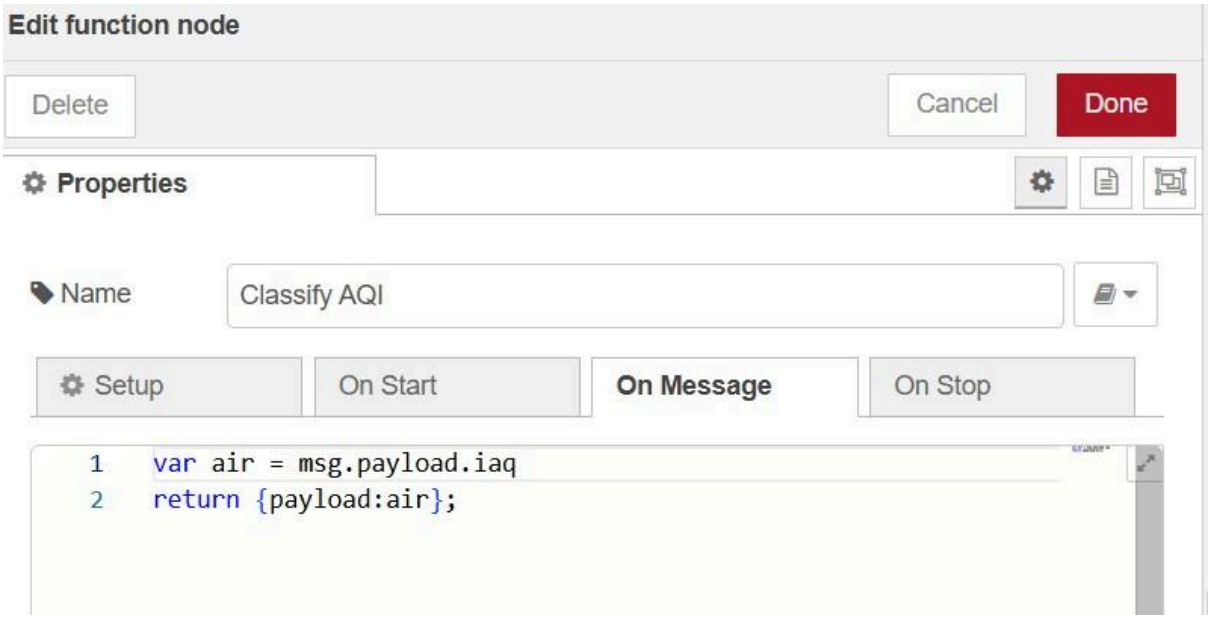
Function-Humidity



Function-light



Function-Classify AQI



Conclusion

This exercise introduced the MQTT protocol and its practical usage with Python and Eclipse Paho. By careful creation of a publisher and a subscriber, we demonstrated the fundamental characteristics of

message exchange with an MQTT broker. We also identified defects in an MQTT implementation given and corrected to offer proper implementation.