Tutorial for Node-RED and MQTT Integration with Raspberry Pi and DHT22 Sensors

In this tutorial, we will set up a monitoring system on a Raspberry Pi using two DHT22 sensors to measure temperature and humidity. The data will be sent to Node-RED via MQTT and displayed on a dashboard and all data are sent to InfluxDB database. Additionally, an alert system will be configured to send an email when certain conditions are met. We will cover the setup in detail, including node configurations and dashboard integration.

1. Requirements

- Hardware:
- Raspberry Pi 4 Model B
- 2 DHT22 temperature and humidity sensors
- 1 buzzer
- Connection wires (jumper wires)
- Breadboard (optional)

- Software:
- Raspberry Pi OS (updated)
- Python 3
- InfluxDB 1.x
- paho-mqtt Python library
- Adafruit_DHT Python library
- RPi.GPIO Python library
- Node-RED
- Gmail App Password (for email node)

2. Hardware Setup

1. Connect the first DHT22 sensor: VCC -> Raspberry Pi 5V, GND -> Raspberry Pi GND, DATA -> GPIO4 (Pin 7).
2. Connect the second DHT22 sensor: VCC -> Raspberry Pi 5V, GND -> Raspberry Pi GND, DATA -> GPIO17 (Pin 11).
3. Connect the buzzer: Positive -> GPIO18 (Pin 12), Negative -> GND.

3. Software Setup

3.2. Raspberry Pi Update

First, update the system packages:

```
sudo apt-get update
sudo apt-get upgrade
```

3.3. Install Python and required libraries

1. Install Python and required libraries:

```
sudo apt-get install python3 python3-pip
pip3 install paho-mqtt Adafruit_DHT RPi.GPIO influxdb-client
```

2. Install InfluxDB:

```
sudo apt-get install influxdb
```

3. Start and enable InfluxDB:

```
sudo systemctl start influxdb && sudo systemctl enable influxdb
```

4. Python Code for Data Collection

The following Python code reads data from the DHT22 sensors and sends it to InfluxDB. Additionally, it activates a buzzer if certain conditions are met.

```
# InfluxDB configuration for 1.x
influx_host = 'localhost'
influx_port = 8086
influx_dbname = 'sensor_data_DB'  # Your database name
influx_user = 'user_tg'  # If authentication is enabled,
enter your username
influx_password = 'password_tg'  # If authentication is
enabled, enter your password

# Create an InfluxDB client
influxdb_client = InfluxDBClient(host=influx_host,
port=influx_port, username=influx_user,
password=influx_password, database=influx_dbname)

# GPIO Setup
GPIO.setmode(GPIO.BCM)
GPIO.setup(BUZZER_PIN, GPIO.OUT)

# MQTT Client Setup
client = mqtt.Client()
client.connect(MQTT_BROKER, MQTT_PORT, 60)

def publish_data(topic, message):
    client.publish(topic, message)

def buzz_on():
    GPIO.output(BUZZER_PIN, GPIO.HIGH)

def buzz_off():
    GPIO.output(BUZZER_PIN, GPIO.LOW)

## Write InfluxDB
def write_to_influxdb(sensor_id, temperature, humidity):
    json_body = [
        {
            "measurement": "environment",
            "tags": {
                "sensor": sensor_id,
                "location": "ISU-PHY-LAB"
            },
            "fields": {
                "temperature": temperature,
                "humidity": humidity
            }
        }
    ]
    influxdb_client.write_points(json_body)
```

```
try:
    while True:
        humidity1, temperature1 =
Adafruit_DHT.read_retry(DHT_SENSOR1, DHT_PIN1)
        humidity2, temperature2 =
Adafruit_DHT.read_retry(DHT_SENSOR2, DHT_PIN2)

        if humidity1 is not None and temperature1 is not
None:
            temperature1 = round(temperature1, 4)
            humidity1 = round(humidity1, 4)
            publish_data(MQTT_TOPIC_TEMPERATURE1,
f"{temperature1} C")
            publish_data(MQTT_TOPIC_HUMIDITY1,
f"{humidity1} %")

            # Write the values to InfluxDB
            write_to_influxdb("sensor1", temperature1,
humidity1)
        else:
            publish_data(MQTT_TOPIC_ALERT, "Sensor 1
Error")

        if humidity2 is not None and temperature2 is not
None:
            temperature2 = round(temperature2, 4)
            humidity2 = round(humidity2, 4)
            publish_data(MQTT_TOPIC_TEMPERATURE2,
f"{temperature2} C")
            publish_data(MQTT_TOPIC_HUMIDITY2,
f"{humidity2} %")

            # Write the values to InfluxDB
            write_to_influxdb("sensor2", temperature2,
humidity2)
        else:
            publish_data(MQTT_TOPIC_ALERT, "Sensor 2
Error")

        # Check thresholds
        if (humidity1 and temperature1 and (temperature1 >
30 or humidity1 > 70)) or \
           (humidity2 and temperature2 and (temperature2 >
30 or humidity2 > 70)):
            publish_data(MQTT_TOPIC_ALERT, "Threshold
Exceeded")
            buzz_on()
        else:
            buzz_off()

        time.sleep(10)

except KeyboardInterrupt:
    GPIO.cleanup()
    client.close()
```

5. Node-RED Configuration

5.1. MQTT Nodes Setup

For each topic (sensor1/temperature, sensor1/humidity, sensor2/temperature, sensor2/humidity), create separate MQTT In nodes:

1. MQTT In Node Configuration:
- Topic: `sensor1/temperature`
- Broker: `localhost` or your MQTT broker address
- Name: `Temperature Sensor 1`

2. Repeat the same for the following topics:
- `sensor1/humidity` (Name: Humidity Sensor 1)
- `sensor2/temperature` (Name: Temperature Sensor 2)
- `sensor2/humidity` (Name: Humidity Sensor 2)

3. Debug Nodes:
- Add a debug node for each MQTT In node to monitor the incoming data.
- Connect each MQTT In node to its corresponding debug node.

5.2. Dashboard Configuration

1. Gauge Nodes:
- For each sensor, add a Gauge node.
- Group: Select your dashboard group.
- Label: Temperature Sensor 1, Humidity Sensor 1, etc.
- Value Format: `{{msg.payload}}`
- Min/Max Values: Set appropriate min/max values (e.g., 0-50 for temperature, 0-100 for humidity).

2. Chart Nodes:
- For each sensor, add a Chart node.
- Group: Select your dashboard group.
- Label: Temperature Sensor 1, Humidity Sensor 1, etc.
- X-axis: Timestamp (use `msg.timestamp`).
- Y-axis: `{{msg.payload}}`.
- Min/Max Values: Set appropriate min/max values.

3. Connect Nodes:
- Connect each MQTT In node to both the corresponding Gauge and Chart nodes.

5.3. Email Alert Configuration

1. MQTT In Node:
- Topic: `sensor/alert`
- Name: `Sensor Alert`

2. Switch Node:
- Property: `msg.payload`
- Rules:
  - `== "Sensor 1 Error"`
  - `== "Sensor 2 Error"`
  - `== "Threshold Exceeded"`
- Outputs: 3 (one for each condition)

3. RBE Node:
- Function: Set to `rbe` to only pass unique messages.
- Property: `msg.payload`
- Connect each output of the Switch node to its corresponding RBE node.

4. Change Nodes:
- Property: `msg.payload`
- Value: Set the value to the corresponding alert message, e.g., "Sensor 1 Error" for the first, etc.
- Connect each RBE node to its corresponding Change node.

5. Email Node:
- To: Your email address
- Subject: `MQTT Sensor Alert`
- Body: Use `msg.payload` to include the alert message.
- SMTP Server: smtp.gmail.com
- Username: Your email address
- Password: App password (generated from Google Account). **Create an application password from your Google account settings and configure the mail node using this password.*

6. Connecting Nodes:
- Connect each Change node to the Email node.
- This way, the appropriate email will be sent based on the alert message.

6. Summary and Conclusion

In this section, we configured separate MQTT nodes for each sensor data type and displayed them on the Node-RED dashboard using Gauge and Chart nodes.
Additionally, we created an alert system that sends an email based on specific conditions. This setup ensures that we receive only one alert
for a specific condition, and subsequent identical alerts are suppressed using the RBE node. This tutorial covered all the necessary steps to
integrate MQTT, Node-RED, and email alerts efficiently.

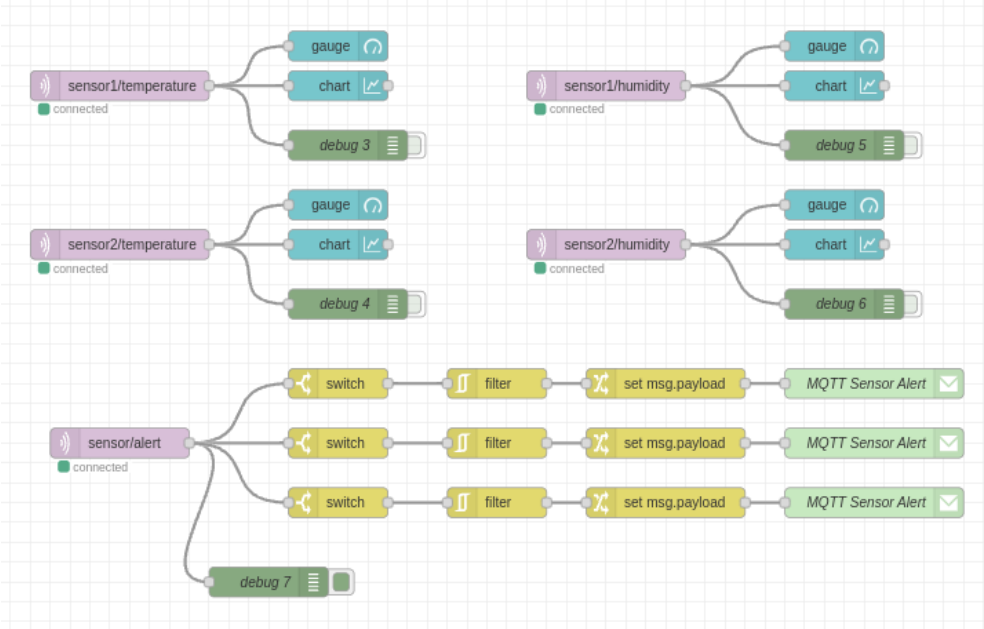

Figure 1 Node-red Configuration

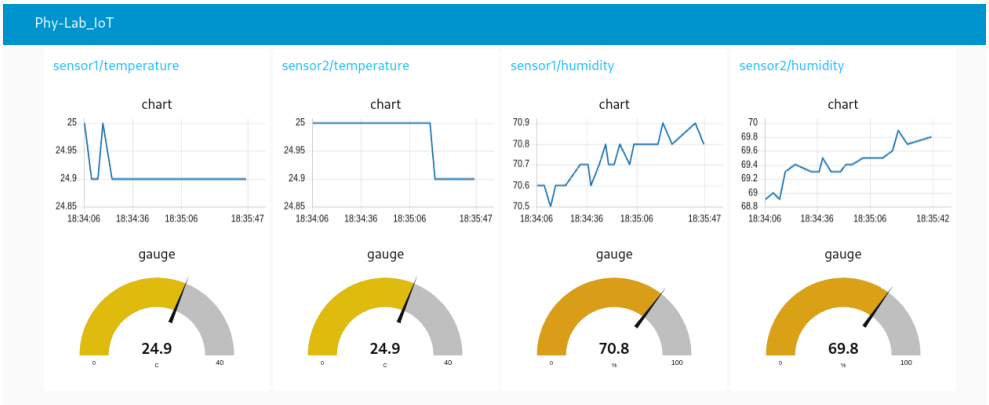

Figure 2 Node-red User-Interface Screen