Exercise 4

Learning outcome: Students will learn to configure Node-RED for data integration and utilize databases like InfluxDB and MongoDB for efficient data storage and management in IoT systems.

Tasks to do:

- Setup Node Red for connecting the data to be received from MQTT server with database
- Utilized databases to store the data from the server (Influx DB)
- Output control

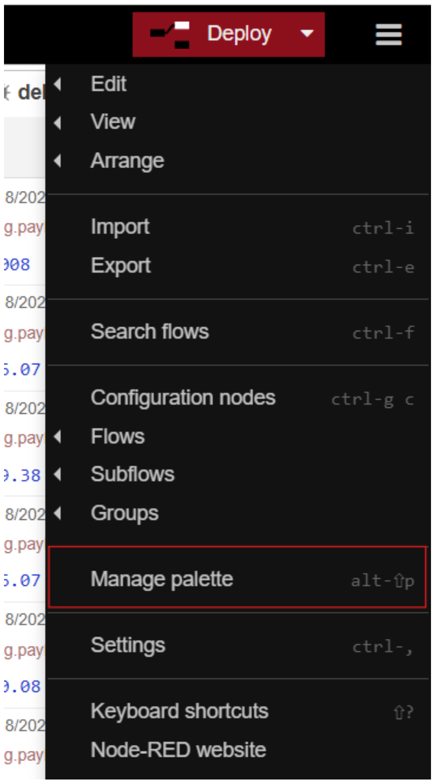
Node-Red:

Node-RED is a programming tool for wiring together hardware devices, APIs and with other online services through flow-based programming. Node-Red is very useful for developing iot based project in this exercise we will utilize node-red to acquire the data being published on the MQTT cloud server and then we will store the data into influx dB based on local machine

Implementation Steps:

Setup Node-RED for Connecting Data from MQTT Server to InfluxDB

- **Install Node-RED**:
  - For Node-red installation you need to have **nodejs** already installed if not please install that first
    Verify installation and check the version : node -v
  - The start the command prompt as administrator and run the command below in CLI to do the installation : Node-red
    ```
    npm install -g --unsafe-perm node-red
    ```
  - Start Node-RED: `node-red`
  - Once installed Open the Node-RED dashboard in your browser (usually at `http://localhost:1880`).
- **Install Required packages:**
  - Go to the menu (top right corner) > "Manage palette" > "Install" tab.



- You can use the Node-RED Manage Palette feature or run the following command in the root directory of your Node-RED in cmd.
  - `npm install node-red-contrib-influxdb`

InfluxDB installation on your local machine

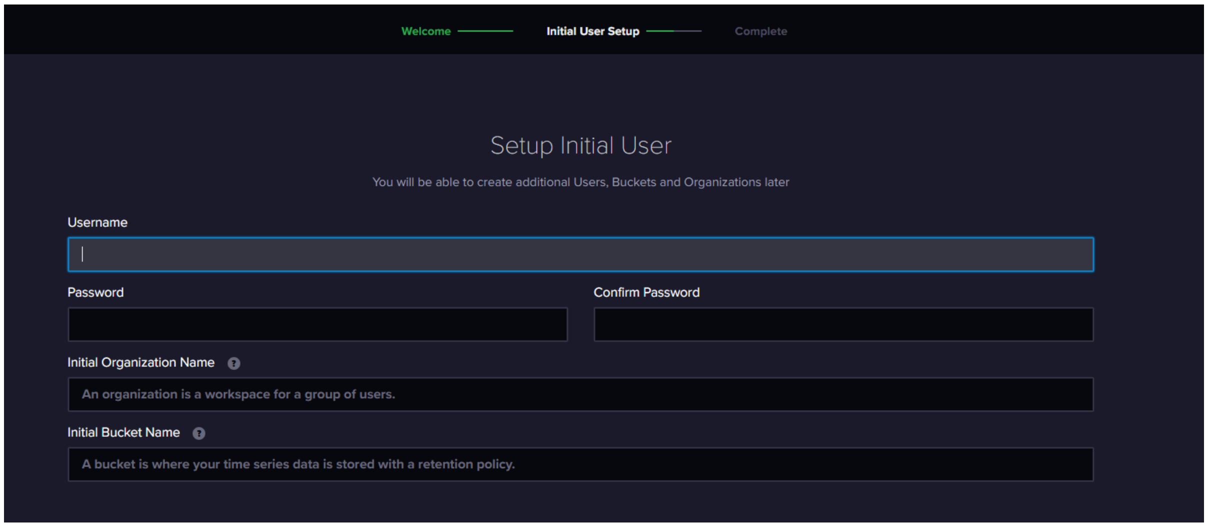
- For the influxdb installation on your local machine follow the instructions given in the following link : InfluxDB
- For installation you will first need to download the .rar file
- Now you need to unextract the files in C\programfile - Example command given below

In my case the file was located in downloads:

```
Expand-Archive -Path "E:\Downloads-Syed\influxdb2-2.7.10-windows.zip" -
DestinationPath "C:\Program Files\InfluxData\" -Force
cd "C:\Program Files\InfluxData\"
```

- Once done navigate to the directory where you have installed the files and run with the command: .\influxd.exe
- Access the influxdb on your web browser : http://127.0.0.1:8086/

- Once installed on the first access you will get this page for initial setup and creating a bucket for collecting data – On the next page you will see a **api token** save the token you will need that in nodered



Setup Initial User

You will be able to create additional Users, Buckets and Organizations later

Username

Password

Confirm Password

Initial Organization Name

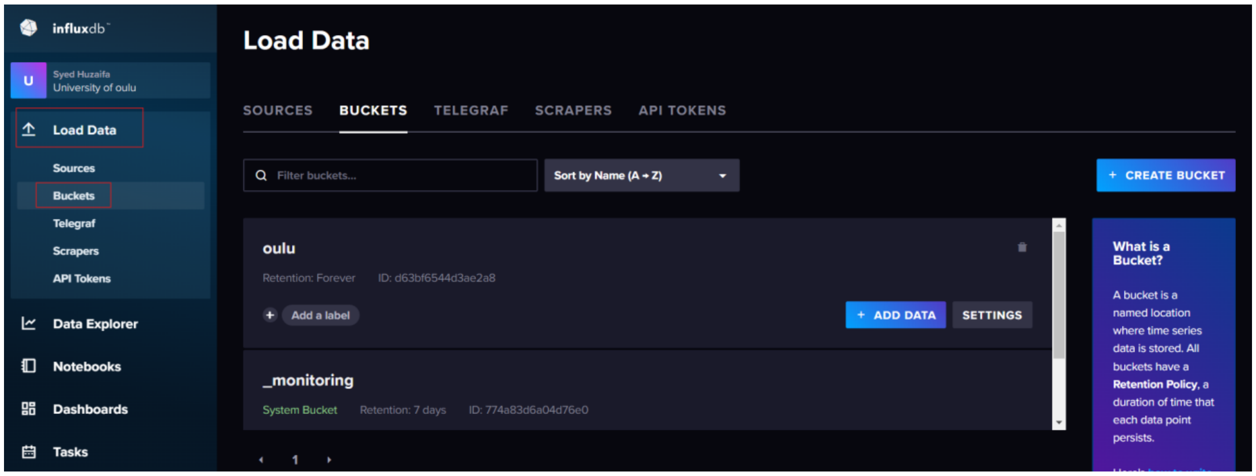
An organization is a workspace for a group of users.

Initial Bucket Name

A bucket is where your time series data is stored with a retention policy.

If in case, you don't get the setup page initially you can create the bucket with the steps below

- Setup on influx db for data collection
  - In the influxdb menu go to load data and bucket



Load Data

oulu

Retention: Forever ID: d63bf6544d3ae2a8

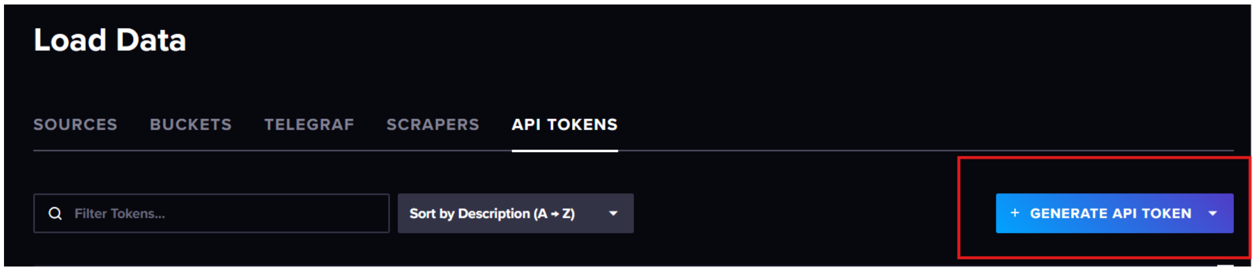
_monitoring

System Bucket Retention: 7 days ID: 774a83d6a04d76e0

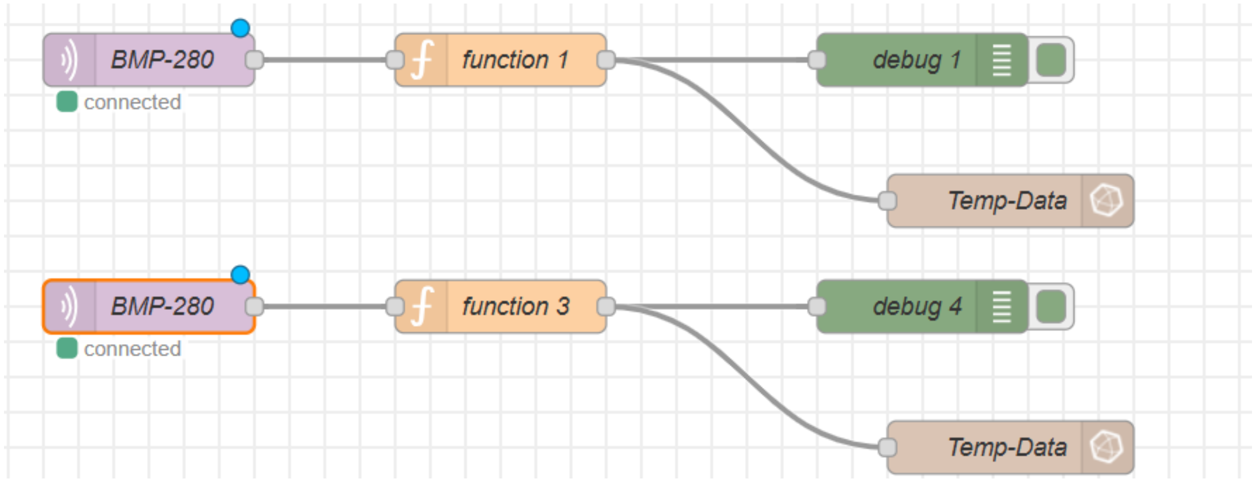
What is a Bucket?

A bucket is a named location where time series data is stored. All buckets have a **Retention Policy**, a duration of time that each data point persists.

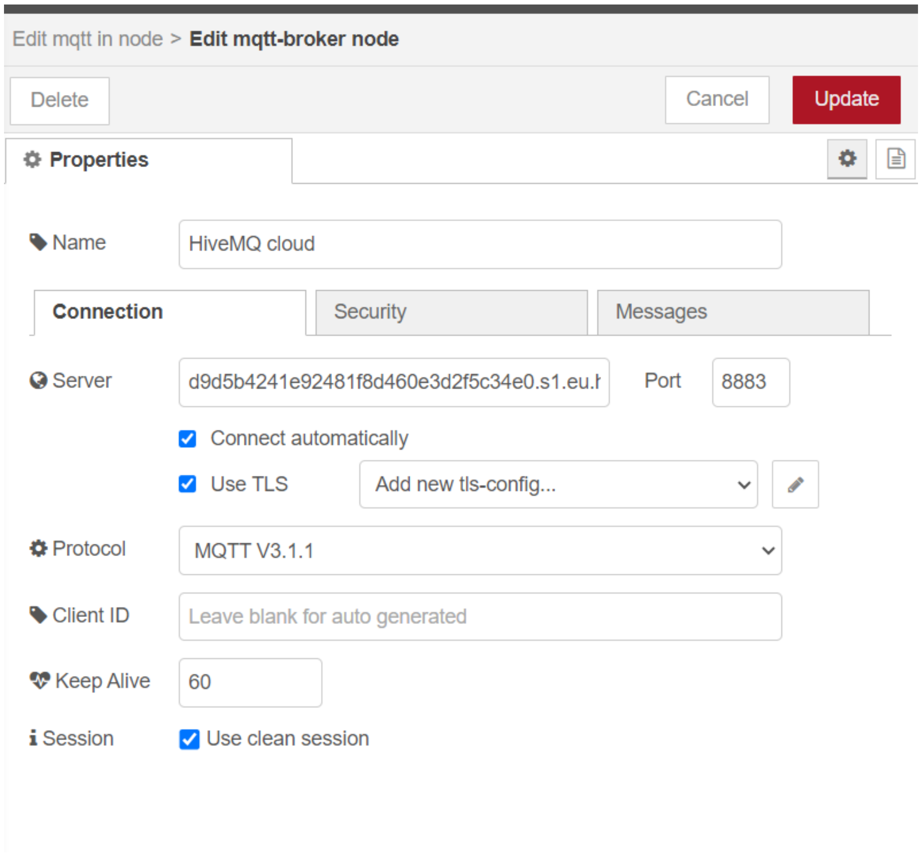
- You need to create a bucket just like in have created by the name of "Oulu"
- Then, go to the API tokens section and create an all-access API token. Save it somewhere, as it will be used later on.



Implementation in Node-RED



- **Configure MQTT Input**:
  - Drag an **`mqtt in`** node onto the workspace from the menu appearing on.
  - Double-click the **`mqtt in`** node to configure it:
    - **Server settings**: Add a new a new sever in our case we have the server on cloud fill the details of the server you created on HiveMQ Cloud



Edit mqtt in node > **Edit mqtt-broker node**

Name: HiveMQ cloud

Server: d9d5b4241e92481f8d460e3d2f5c34e0.s1.eu.h

Port: 8883

☑ Connect automatically

☑ Use TLS — Add new tls-config...

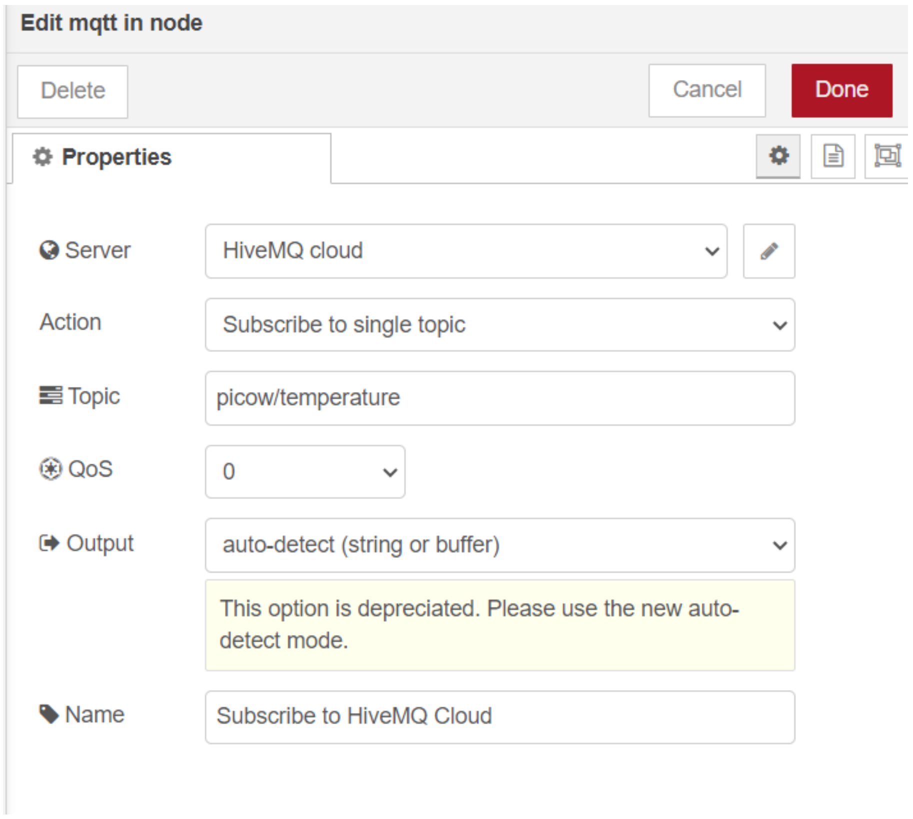
Protocol: MQTT V3.1.1

Client ID: Leave blank for auto generated

Keep Alive: 60

Session: ☑ Use clean session

- **Topic**: Set the topic to which the sensor data is published (which you have specified for publishing in the code).



Server: HiveMQ cloud
Action: Subscribe to single topic
Topic: picow/temperature
QoS: 0
Output: auto-detect (string or buffer)
This option is depreciated. Please use the new auto-detect mode.
Name: Subscribe to HiveMQ Cloud

- **Setup Function Node to Process Data**:
  - Drag a `function` node onto the workspace and connect it to the `mqtt in` node.
  - Double-click the `function` node and write JavaScript code to format the incoming data into a floating point for Influx DB.You can use the following code :

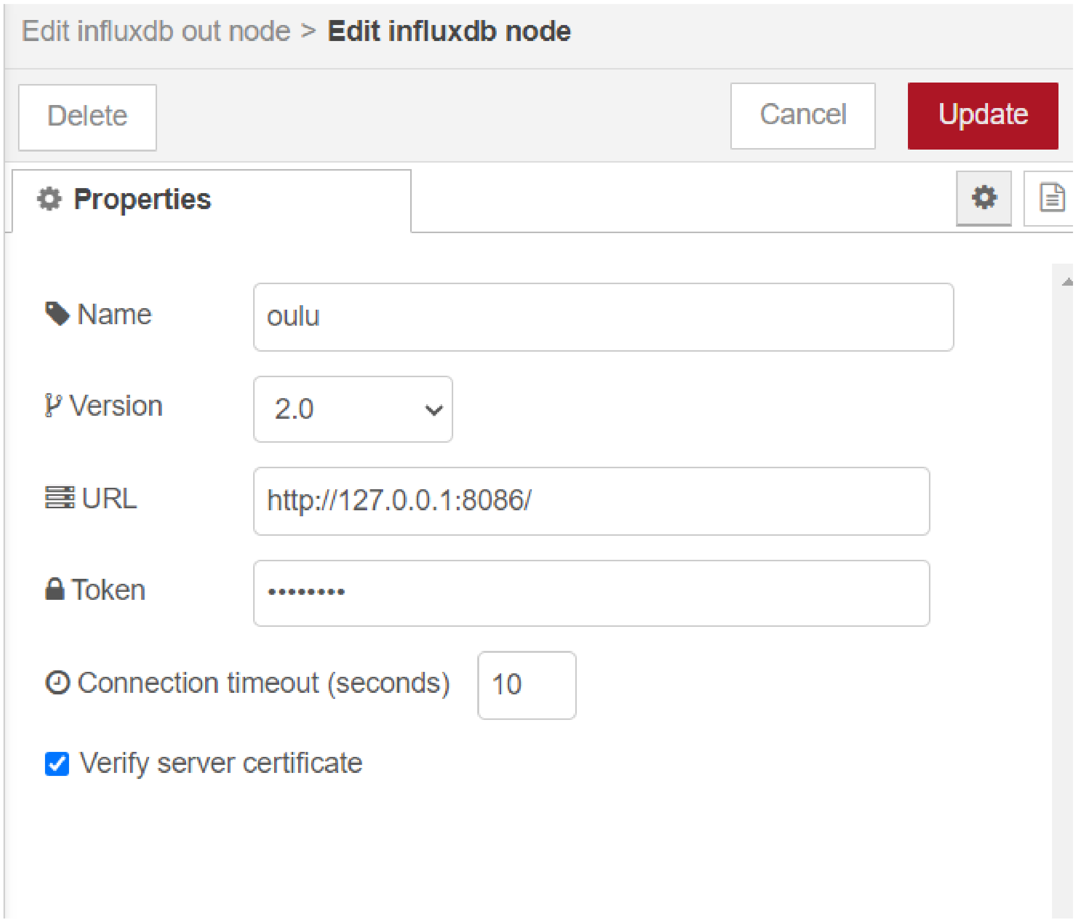
```
newMsg = {};
newMsg.payload = parseFloat(msg.payload.slice(0, 5));
return newMsg;
```

  - Debug: Once the setup is done in node red you can utilize debug block into workspace to see the data being published on mqtt server

At the end when you have setup mqttin ,function and debug for all the three topics on which the data is being published, you will be able to have a similar flow diagram given below

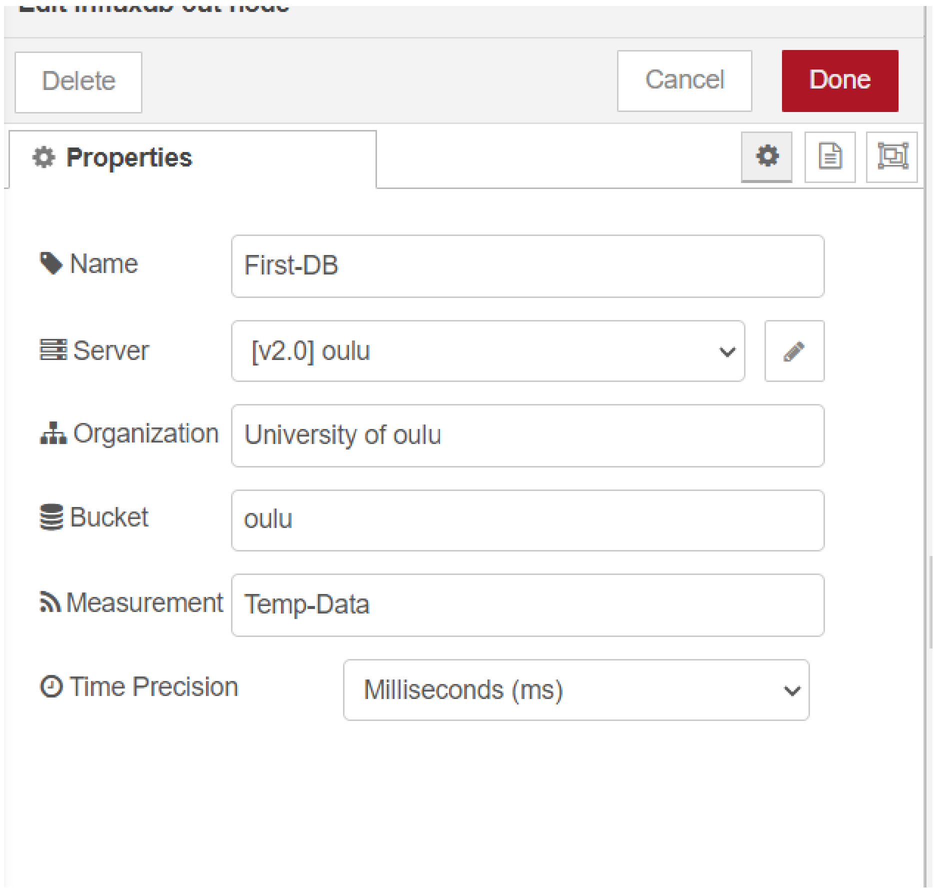
- **Configure Node-RED to Store Data in InfluxDB**:
  Now we need to come back on that node-red
  - Drag an `influx dB out` node onto the canvas and connect it to the `function` node.

  - Double-click the `influxdb out` node to configure it:
    - **Server Settings:**
      In the server settings, provide a name, the InfluxDB URL, and the token you copied while creating the bucket in InfluxDB



Name: oulu
Version: 2.0
URL: http://127.0.0.1:8086/
Token: ••••••••
Connection timeout (seconds): 10
☑ Verify server certificate

- Then, you need to add the details according to the information you provided in InfluxDB while creating the bucket and name the measurement as well



Delete | Cancel | Done

Properties

Name: First-DB

Server: [v2.0] oulu

Organization: University of oulu

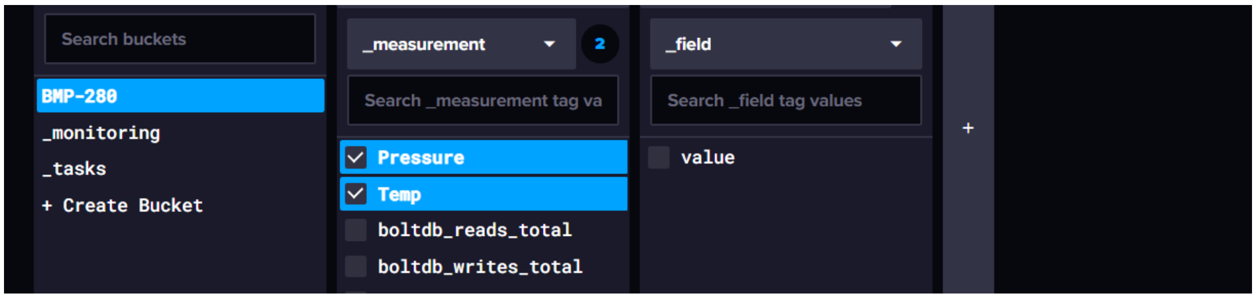
Bucket: oulu

Measurement: Temp-Data

Time Precision: Milliseconds (ms)

- **Visualize Data in InfluxDB**

Once the setup is done in Node-RED, go to InfluxDB Data Explorer. Select the bucket, then in the measurement section, choose the measurement name specified in the InfluxDB block. For example, in the image above, one measurement name was 'Temp'



Search buckets

BMP-280
_monitoring
_tasks
+ Create Bucket

_measurement 2

Search _measurement tag va

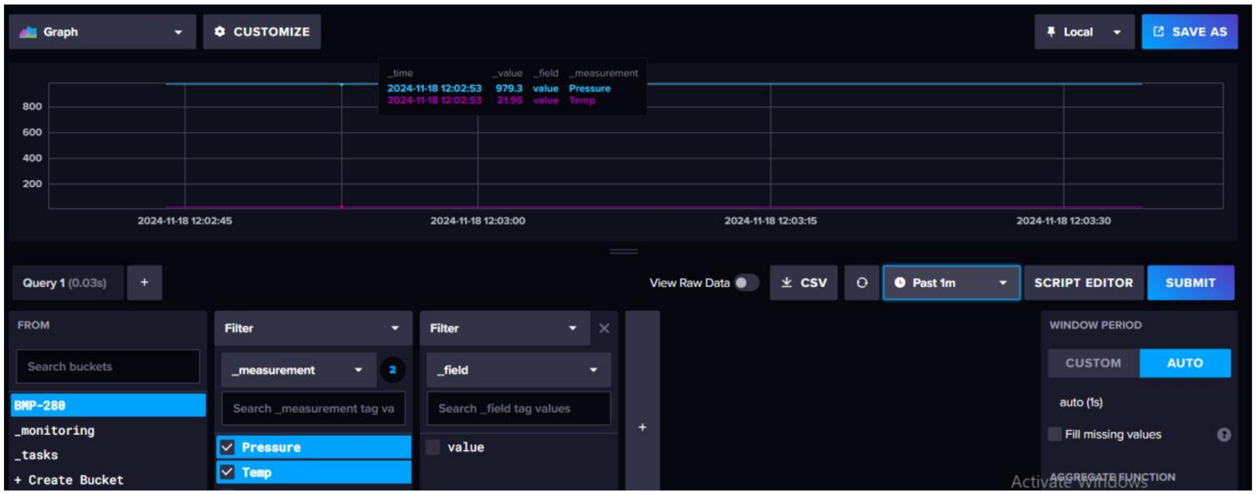
☑ Pressure
☑ Temp
☐ boltdb_reads_total
☐ boltdb_writes_total

_field

Search _field tag values

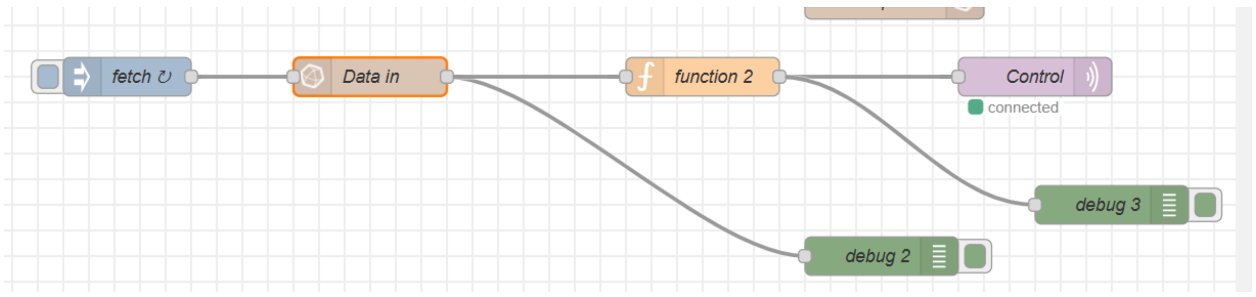
☐ value

- Once you have selected the measurement, you will be able to see the graph of the incoming measurements from the MQTT cloud server.

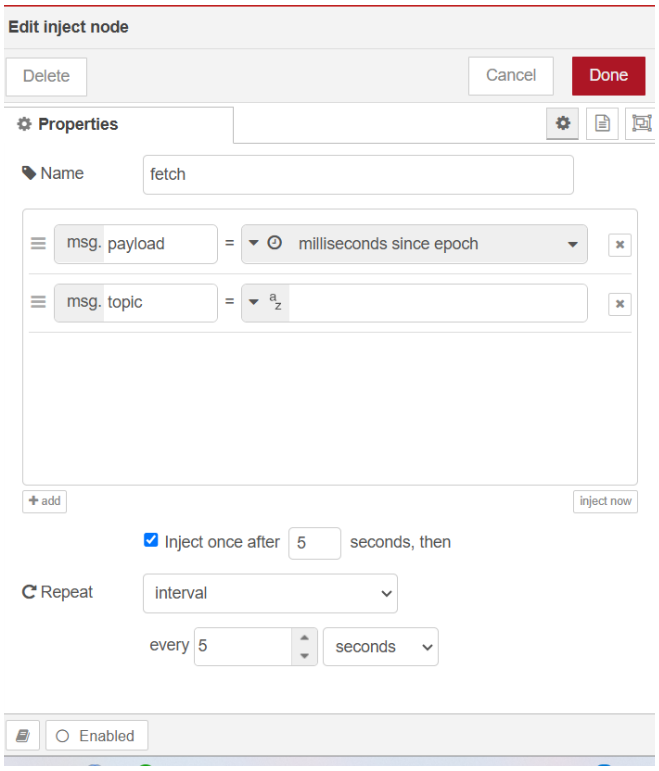


Controlling the Output on PICO W:

Implementation diagram:



1. **Inject Node**: Used to periodically trigger data retrieval.


Edit inject node

Name: fetch

msg. payload = milliseconds since epoch

msg. topic =

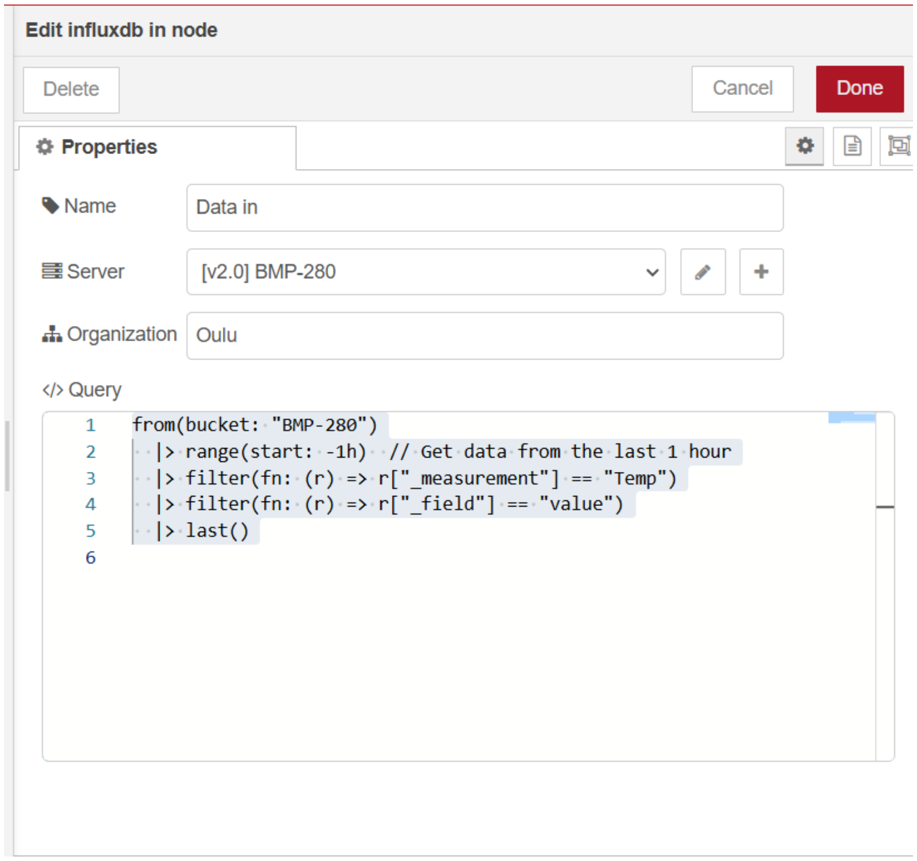
Inject once after 5 seconds, then

Repeat: interval

every 5 seconds

Enabled

2. **Data In Node**: An “ **InfluxDB In” Node**, fetching the latest temperature data from the database.


Edit influxdb in node

Name: Data in

Server: [v2.0] BMP-280

Organization: Oulu

Query

```
from(bucket: "BMP-280")
  |> range(start: -1h)  // Get data from the last 1 hour
  |> filter(fn: (r) => r["_measurement"] == "Temp")
  |> filter(fn: (r) => r["_field"] == "value")
  |> last()
```

Query for Influx-in node to get the temp data

```
from(bucket: "BMP-280")
  |> range(start: -1h)  // Get data from the last 1 hour
  |> filter(fn: (r) => r["_measurement"] == "Temp")
  |> filter(fn: (r) => r["_field"] == "value")
  |> last()
```

3. **Function 2 Node**: A **Function Node** that processes the temperature data and check the value and determines whether to send "ON" or "OFF" based on a threshold.
   Code for function block in JavaScript:

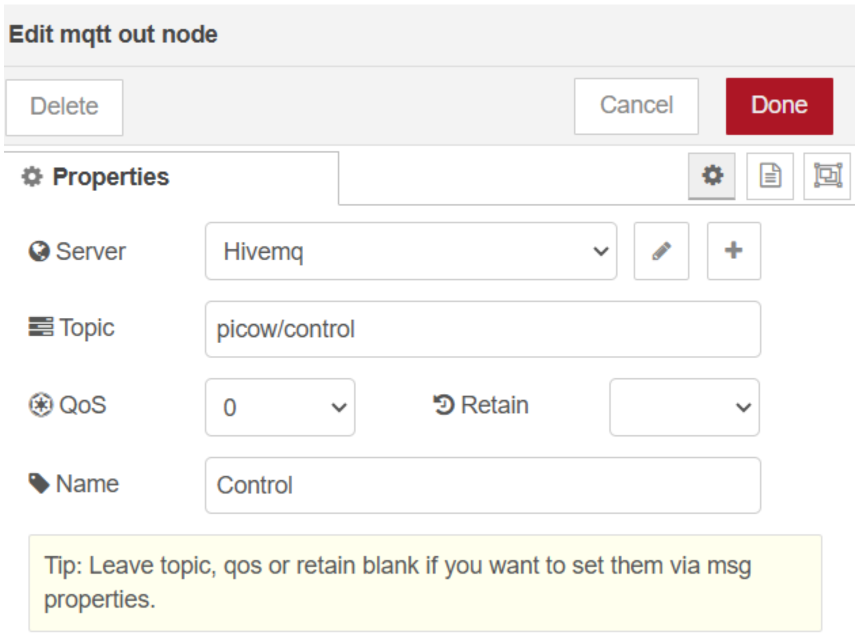
```
// Extract the first object from the payload
let temperatureData = msg.payload[0];

// Ensure the temperature value is available and is a number
if (temperatureData && typeof temperatureData._value === 'number') {
    let temperature = temperatureData._value;

    // If the temperature is above 23°C, send "ON", else send "OFF"
    msg.payload = (temperature > 23) ? "ON" : "OFF";
} else {
    msg.payload = "Error: No valid temperature data found";  // Handle missing or
invalid temperature data
}

// Return the modified message
return msg;
```

4. **Control Node**: **MQTT Output Node**, publishing the "ON"/"OFF" message to a picow/Control topic for the Raspberry Pi Pico W.


Edit mqtt out node

Server: Hivemq

Topic: picow/control

QoS: 0

Retain:

Name: Control

Tip: Leave topic, qos or retain blank if you want to set them via msg properties.

5. **Debug Nodes (2 and 3)**: These debug nodes are placed to monitor the processed data and ensure the function gives the right output values.
6. Add a message callback function in your micropython code to receive the string data from the "Control" topic and control the built-in-Led available on PICO based on that

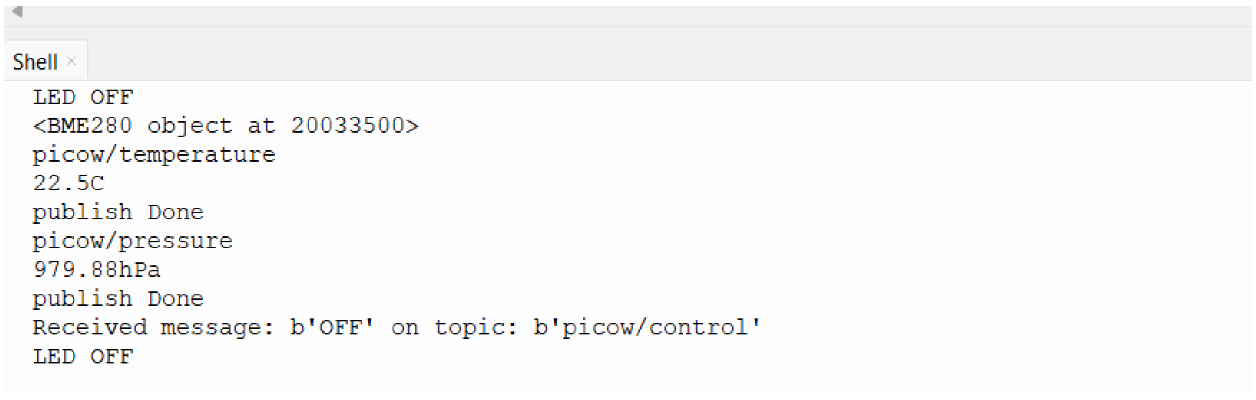
   Example code :

```
def on_message(topic, msg):
    print(f"Received message: {msg} on topic: {topic}")
    if msg == b"ON":
        led_pin.on()  # Turn on the LED
        print("LED ON")
    elif msg == b"OFF":
        led_pin.off()  # Turn off the LED
        print("LED OFF")
client.set_callback(on_message)
client.subscribe(b"picow/control")
```

Add this in loop function to check if there is any new msg :

```
    client.check_msg()
```

Once you have updated the code you will be able to receive the control signals



```
LED OFF
<BME280 object at 20033500>
picow/temperature
22.5C
publish Done
picow/pressure
979.88hPa
publish Done
Received message: b'OFF' on topic: b'picow/control'
LED OFF
```