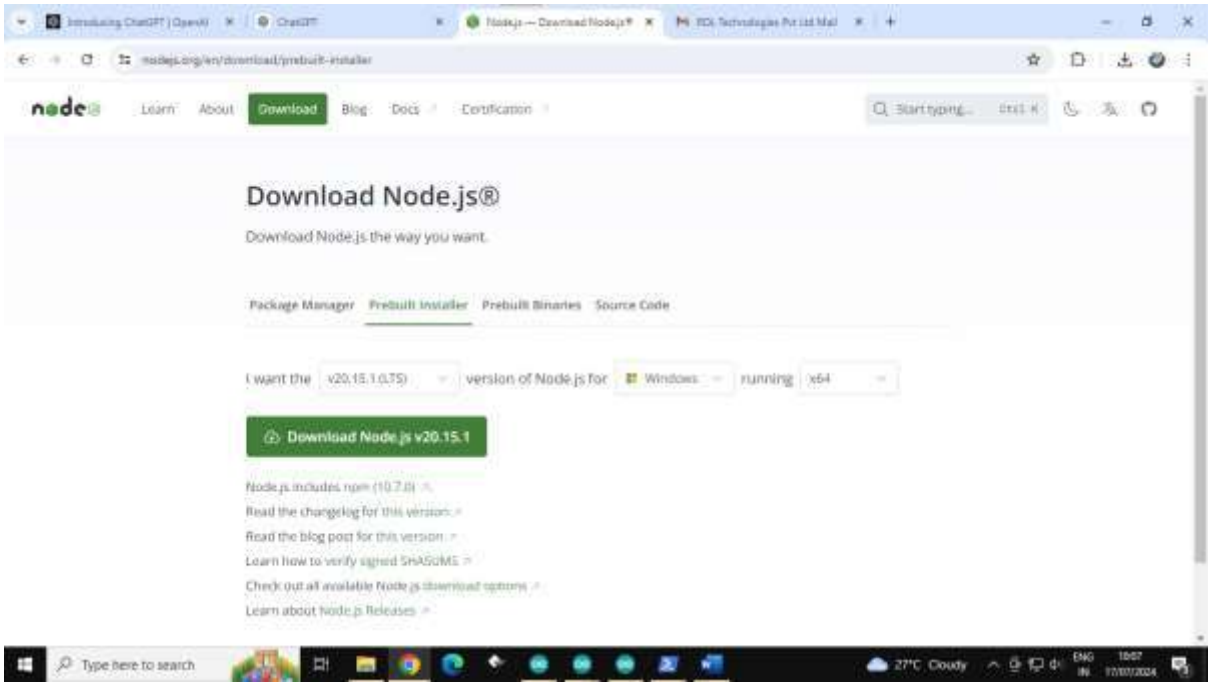
Node-Red Implementation

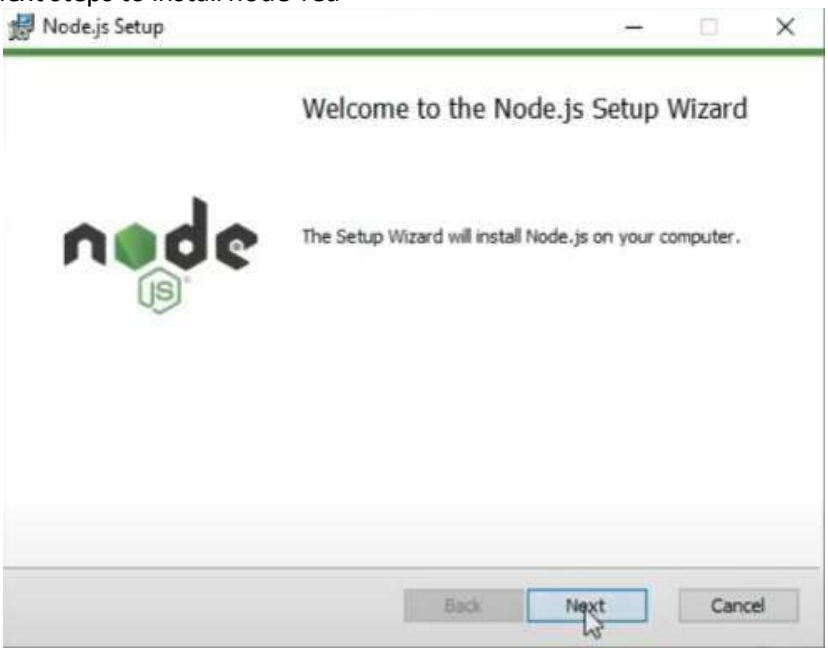
1. How to Install Node-Red in Windows (YouTube Video Reference : LINK)
   https://nodejs.org/en/download/prebuilt-installer : Click on this link. It will redirect you to this website.



Here click on 

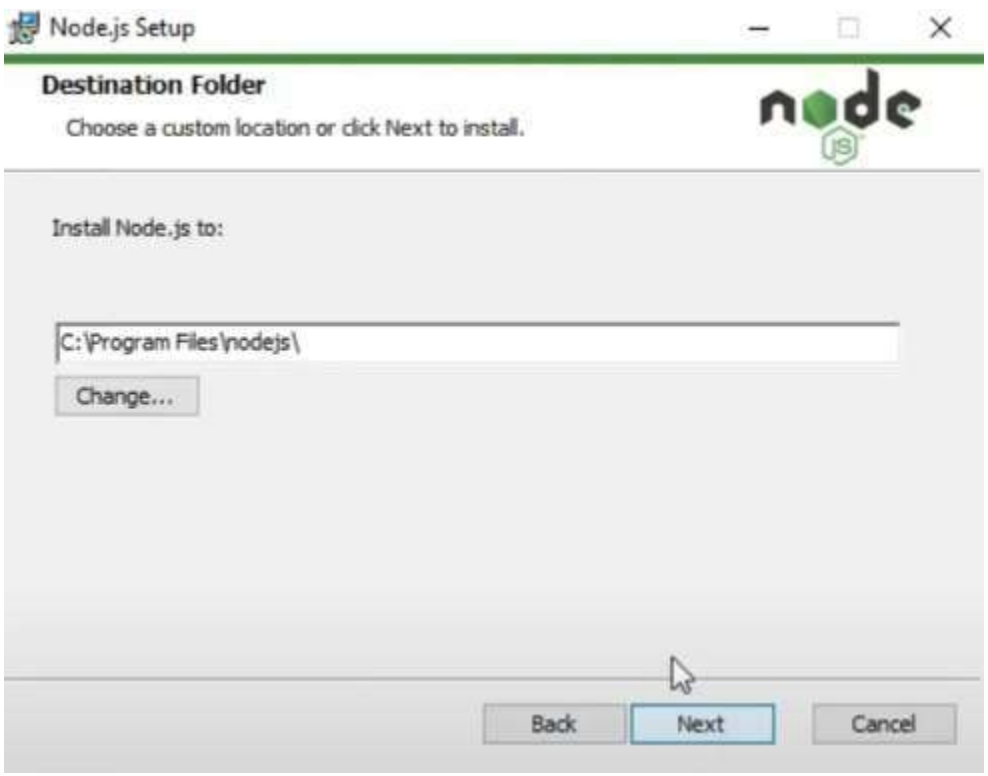
or latest version of the software or if you want the same software I used, then click on this link .

Follow the next steps to install node-red

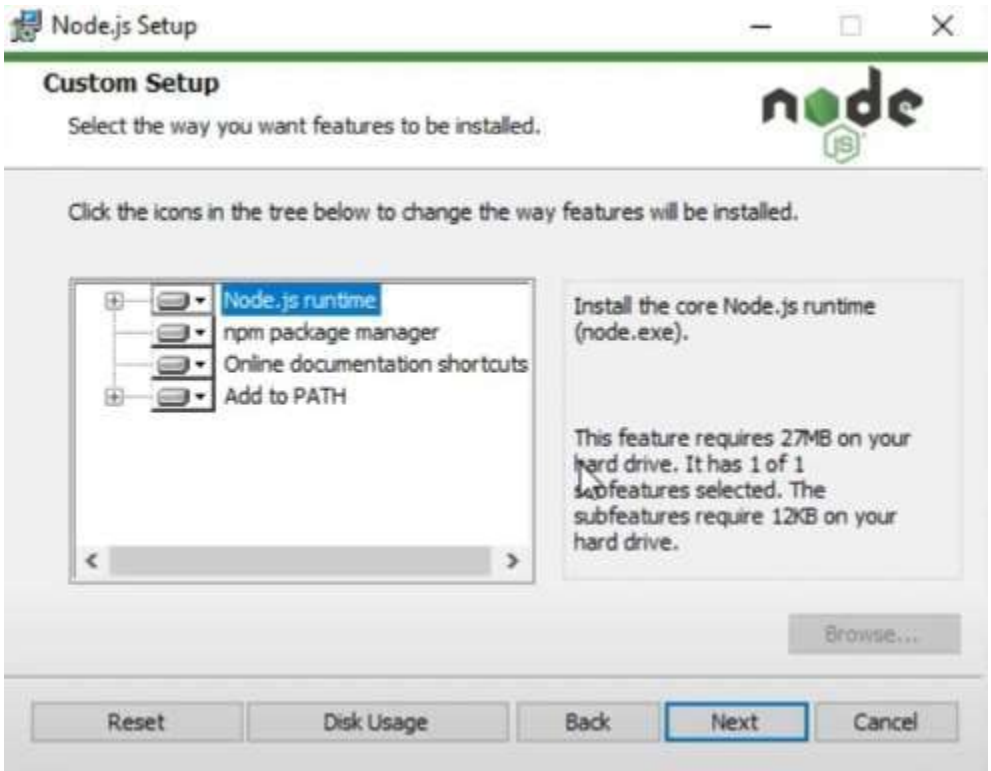


Click on Next.

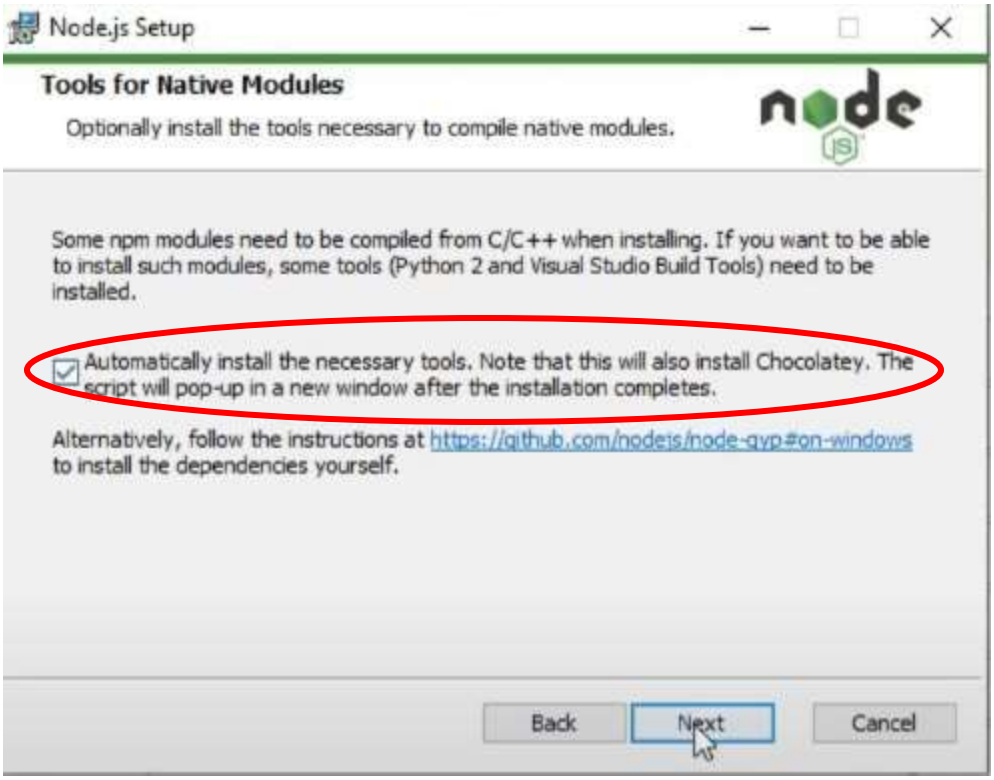
https://highvoltages.co/iot-internet-of-things/mqtt/mqtt-in-nodered-and-mqtt-dashboard/



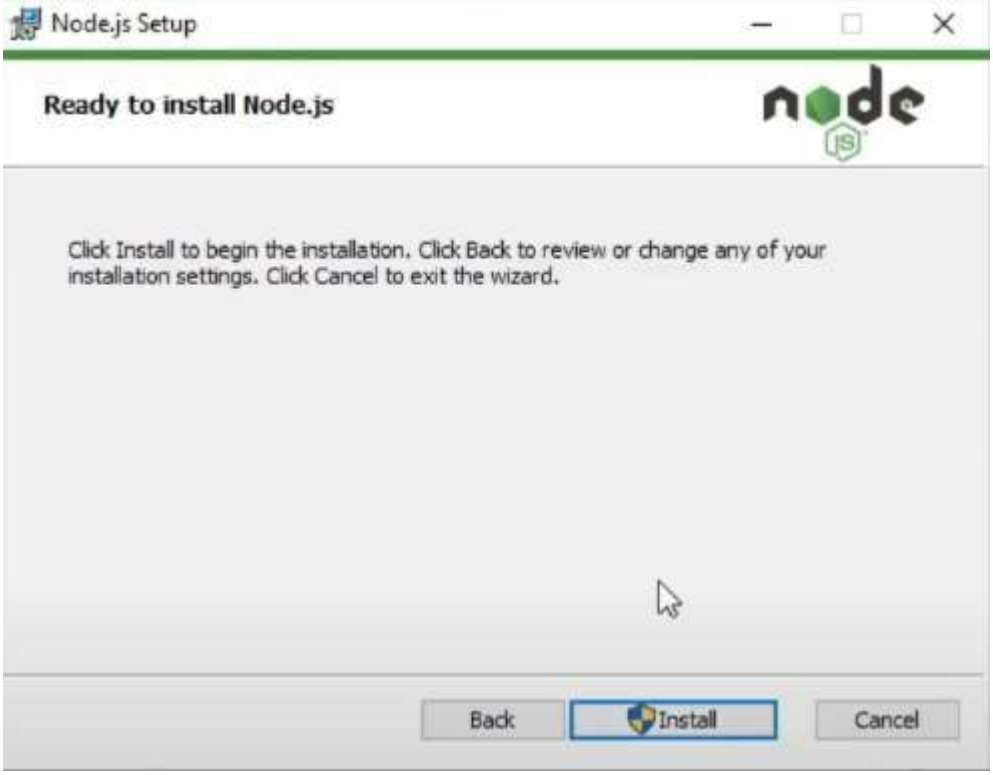
Set destination folder and click on Next



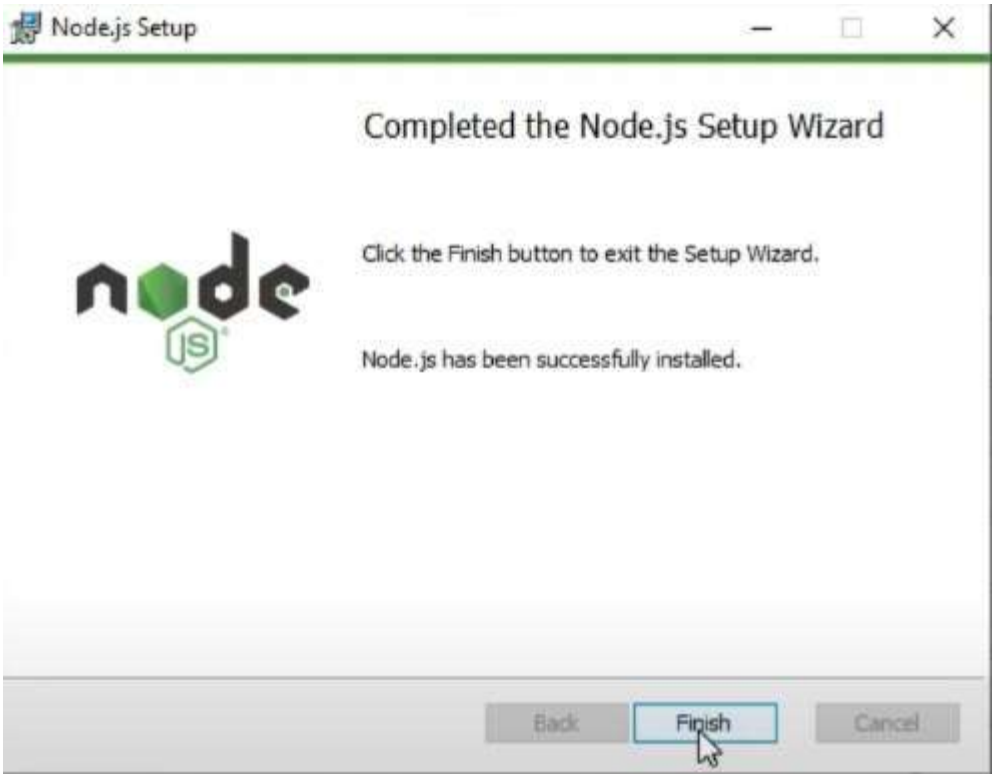
Click on Next



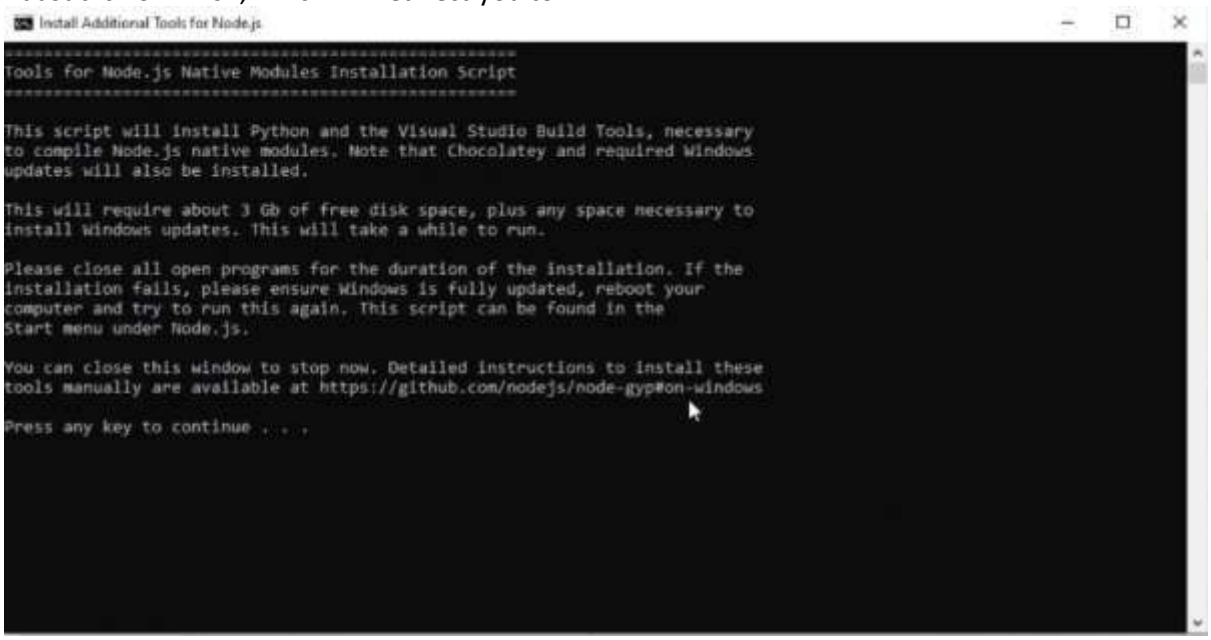
Enable the checkbox and click on Next



Next click on Install

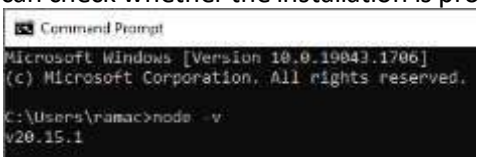


Atlast click on finish, which will redirect you to



This command promt, Press any key to continue. This will redirect to windows powershell. From there it will automatically install the necessary files. It will take 15 to 30 minutes to install depending upon the internet speed.

After the software installation, we can check whether the installation is proper or not by also version



of node js

If you get the response like this, then your software is successfully installed.

Then we need to update npm by the below command

```
C:\Users\ramac>npm install npm
```

```
C:\Users\ramac>npm install npm

added 1 package in 26s

22 packages are looking for funding
  run `npm fund` for details
npm notice
npm notice New minor version of npm available! 10.7.0 -> 10.8.2
npm notice Changelog: https://github.com/npm/cli/releases/tag/v10.8.2
npm notice To update run: npm install -g npm@10.8.2
npm notice
```

Next give the below command in command prompt

```
C:\Users\ramac>npm install npm --global
```

```
C:\Users\ramac>npm install npm --global

added 1 package in 14s

22 packages are looking for funding
  run `npm fund` for details
```

Then if you get the version response

```
C:\Users\ramac>node --version && npm --version
v20.15.1
10.8.2
```

. It should print the both the versions, then the node and npm are successfully installed.

Now we need to install node red,

For that we need to open command prompt and put the below command

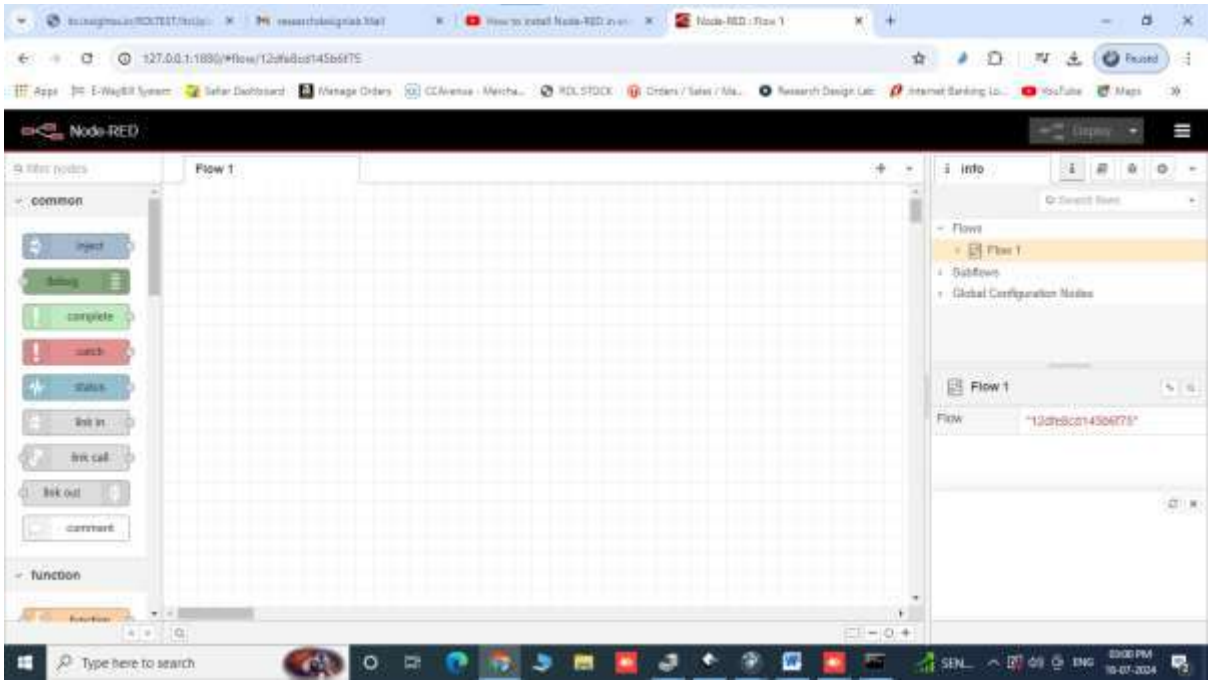
```
C:\Users\ramac>npm install -g --unsafe-perm node-red
```

```
C:\Users\ramac>npm install -g --unsafe-perm node-red

added 312 packages in 1m

60 packages are looking for funding
  run `npm fund` for details
```

Now to run node red, type node-red on command prompt and click on enter, allow the network access if PC asks for access

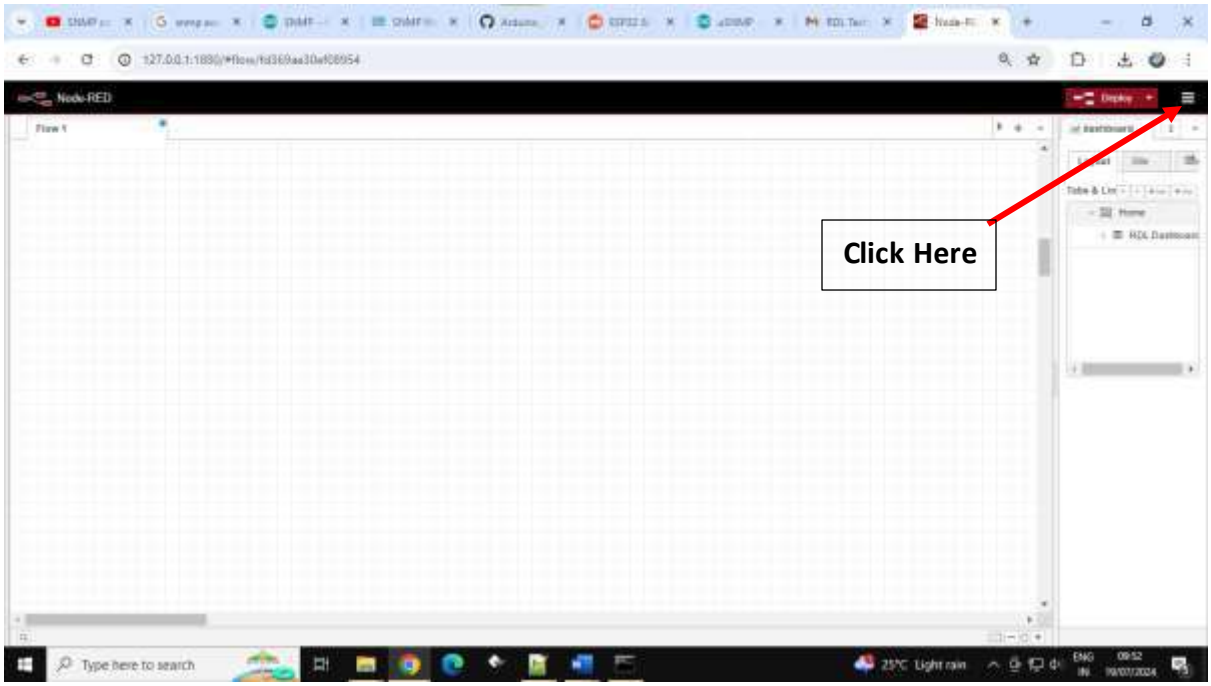
```
18 Jul 14:57:28 - [info] Server now running at http://127.0.0.1:1880/
18 Jul 14:57:28 - [warn] Encrypted credentials not found
18 Jul 14:57:28 - [info] Starting flows
18 Jul 14:57:28 - [info] Started flows
```

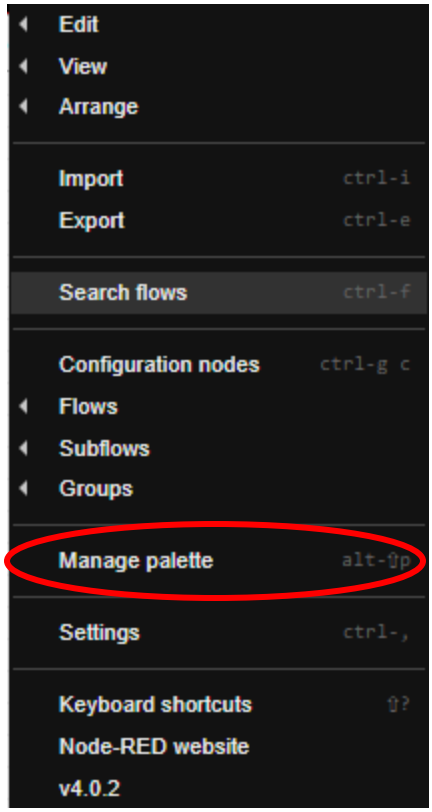
You can simply put the running server url in web browser, it will open the node-red page.



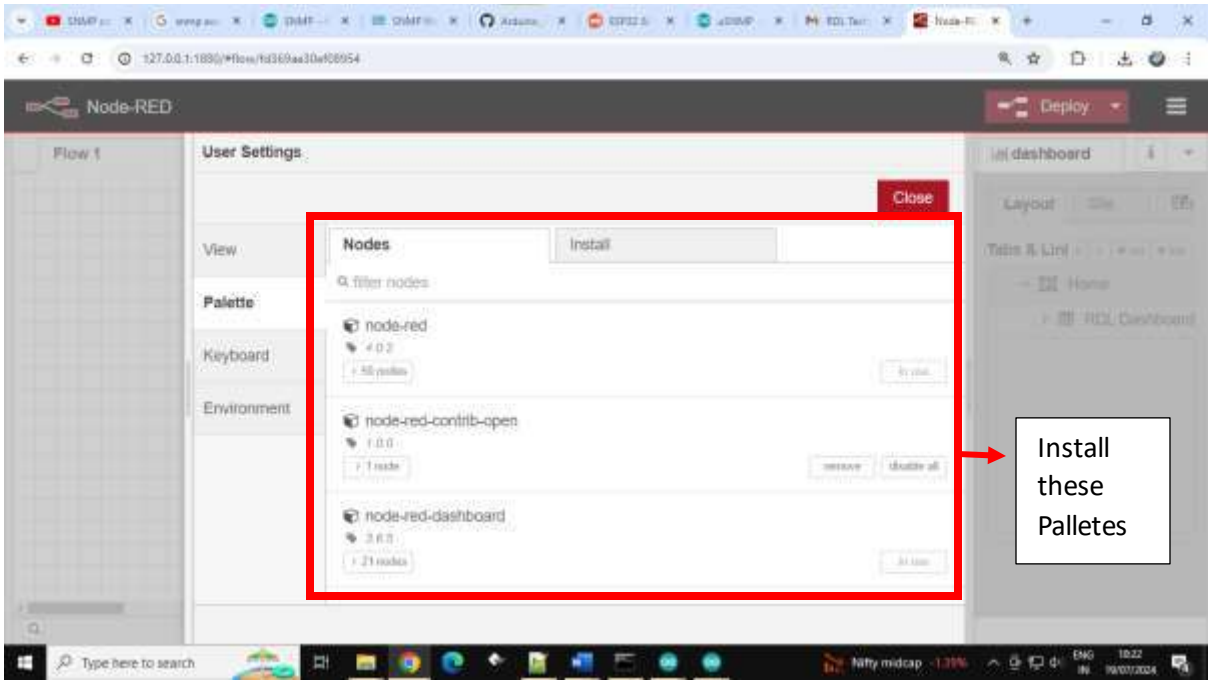
2. Sensor Values and Relay Control using MQTT Connection: (for your reference Link)

To create a dashboard, you need some palletes, to install palletes follow the below instructions,





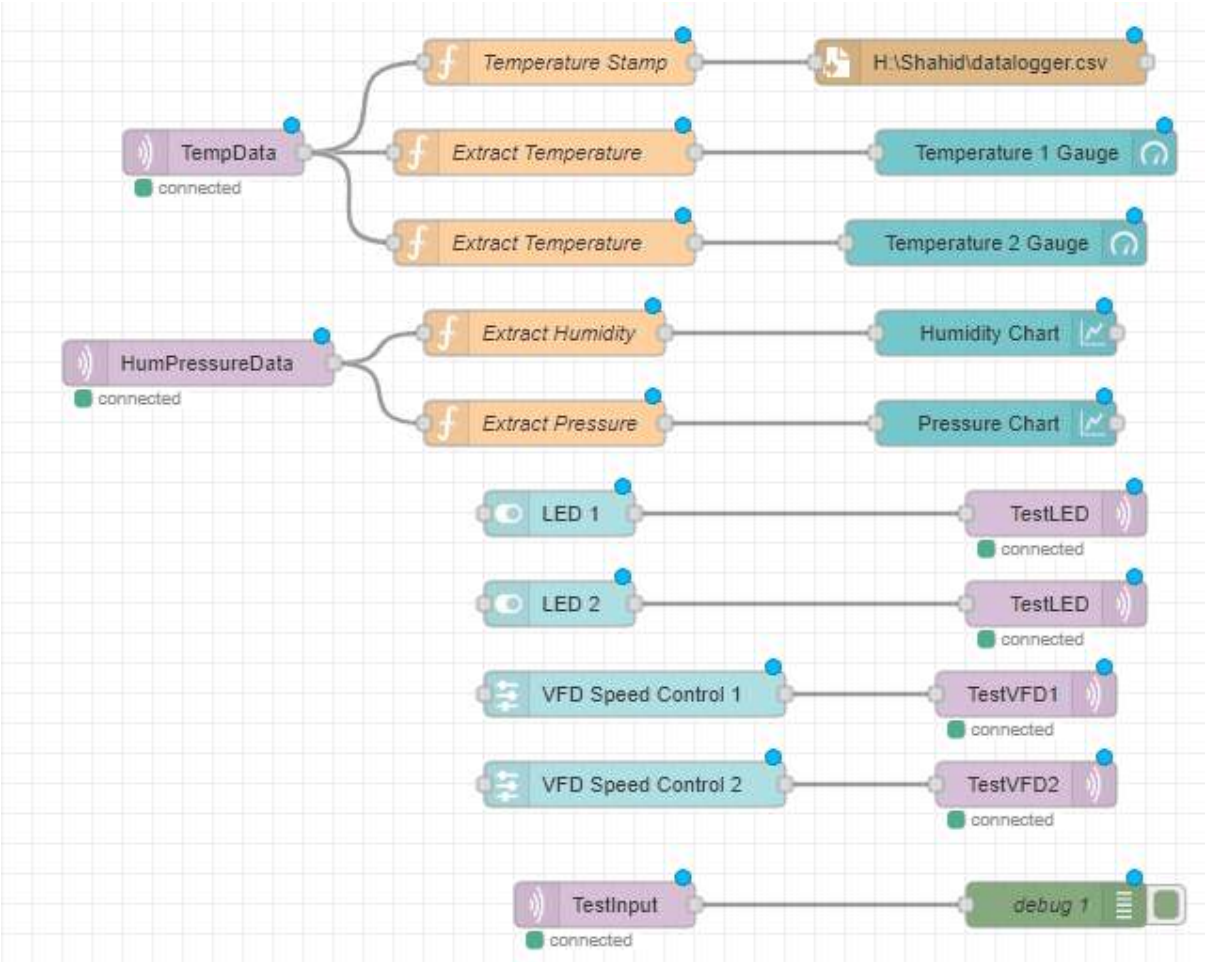
Click on manage pallete



node-red

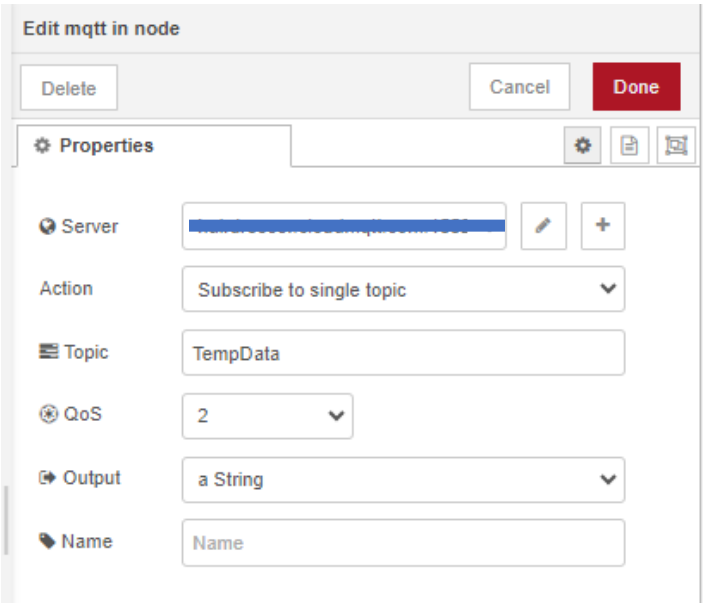
node-red-contrib-open

node-red-dashboard

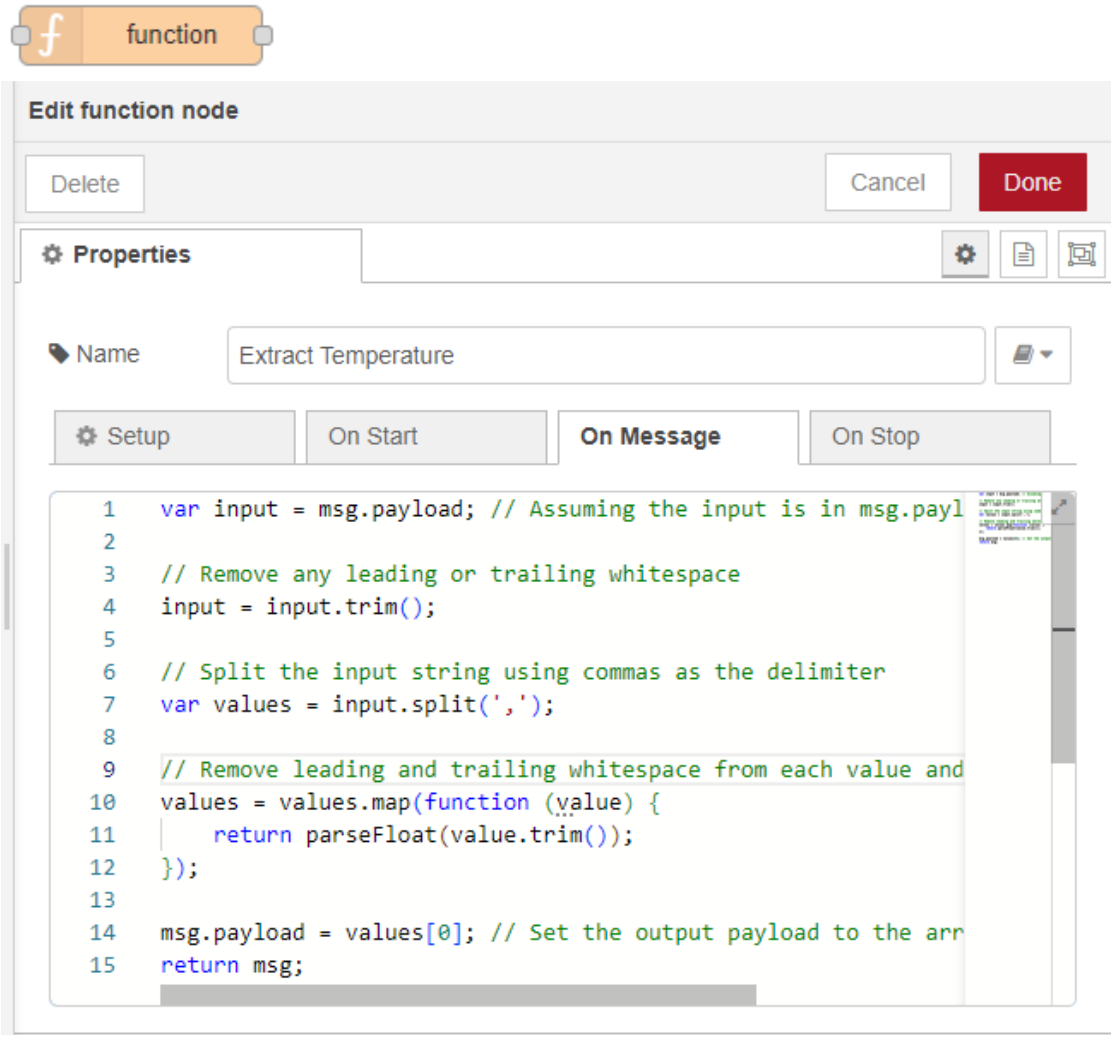


TempData:

It is a MQTT IN node and the settings I used here is given below, Server is the MQTT server, you have to add your MQTT server credentials.

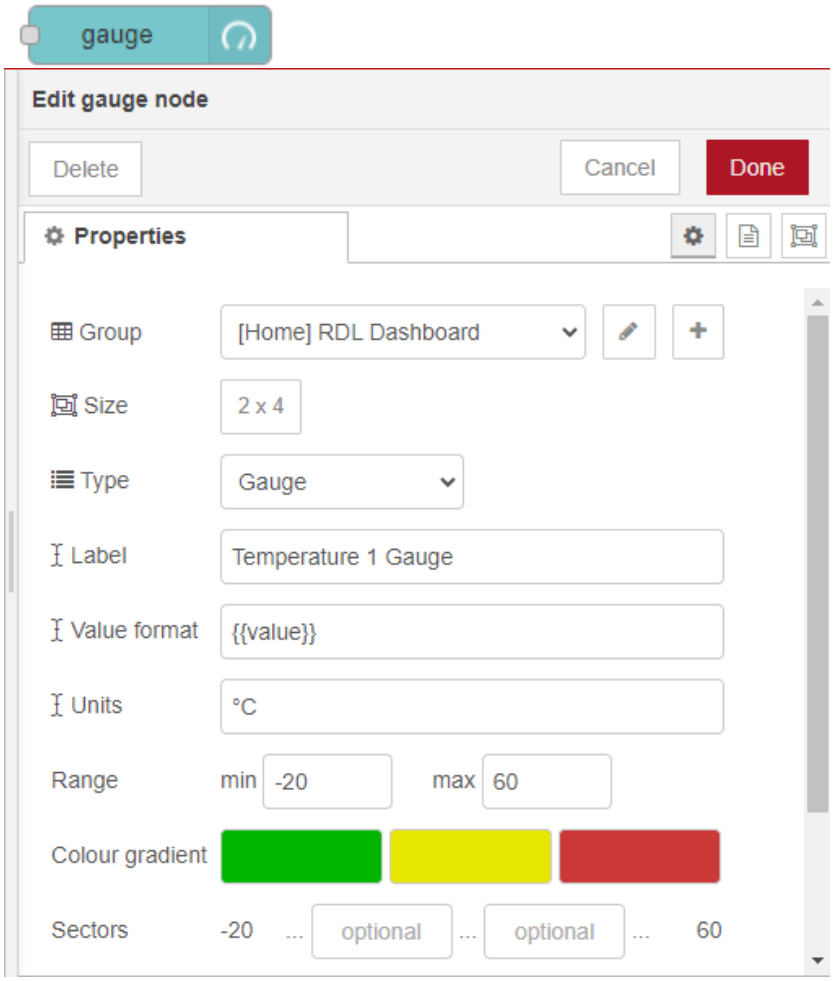


Extract Temperature 1: It is a function, where you will parse the incoming string data to separate the temperature 1 value

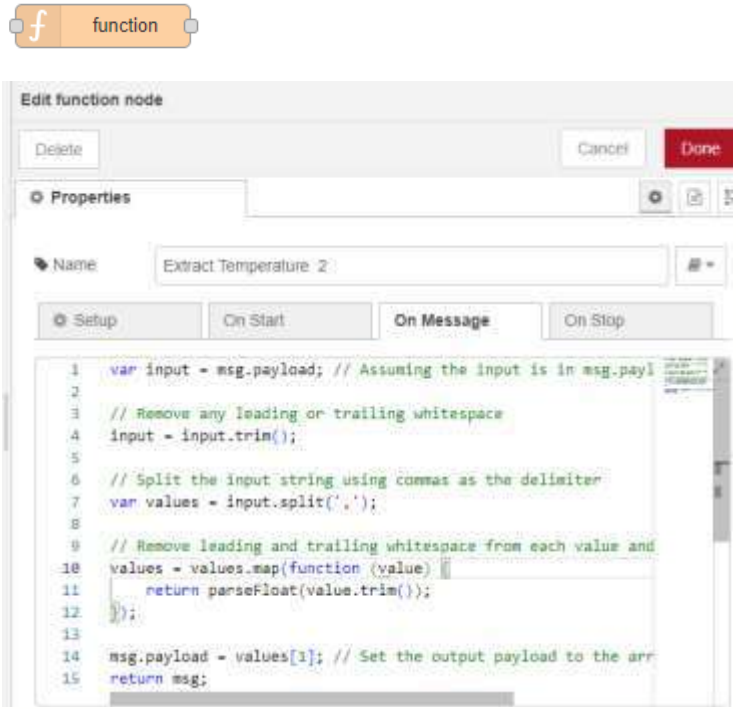



```
var input = msg.payload; // Assuming the input is in msg.payload

// Remove any leading or trailing whitespace
input = input.trim();

// Split the input string using commas as the delimiter
var values = input.split(',');

// Remove leading and trailing whitespace from each value and convert to float
values = values.map(function (value) {
    return parseFloat(value.trim());
});

msg.payload = values[0]; // Set the output payload to the array of float
values
return msg;
```

Temperature Gauge 1 : Dashboard Gauge to show the temperature 1 values

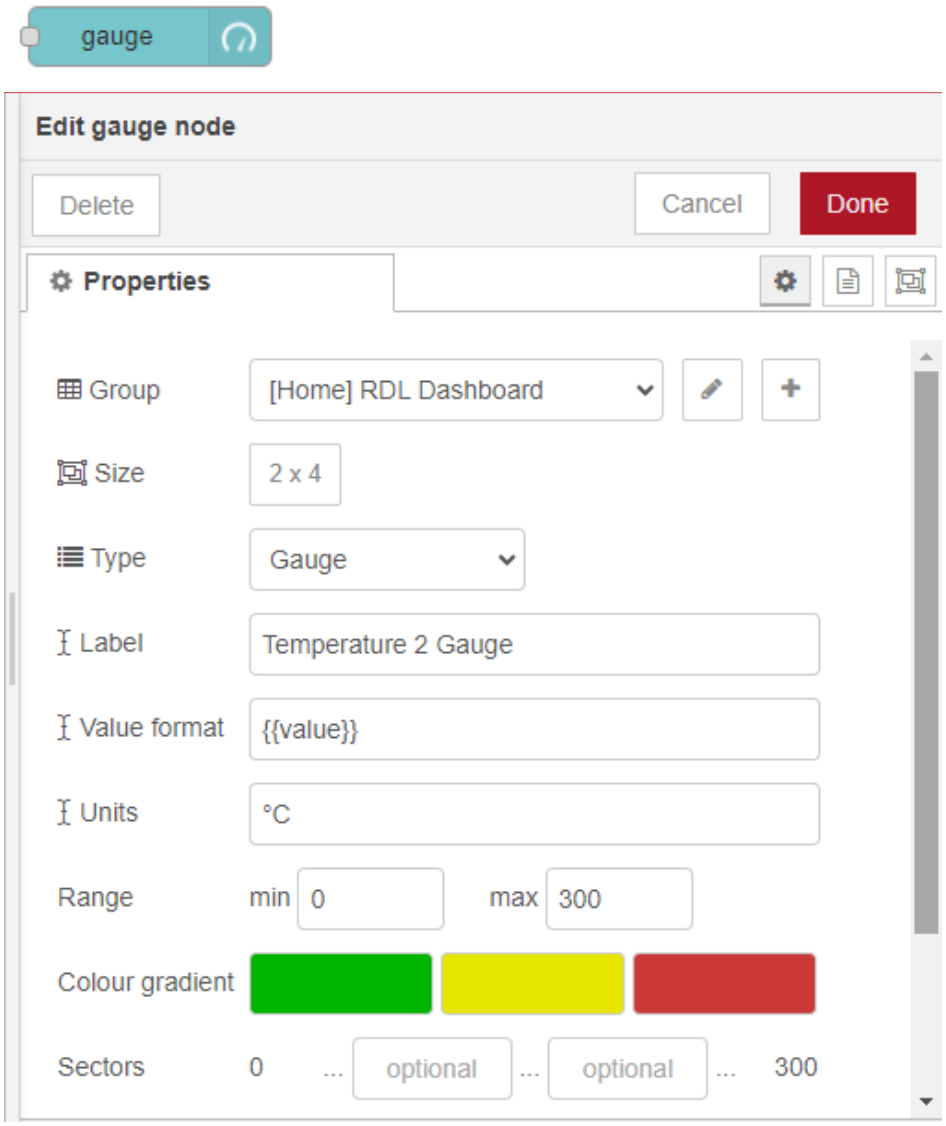


Extract temperature 2 : The incoming temperature values will be like XX,YY where XX is 1st temperature value and YY is the second temperature value.

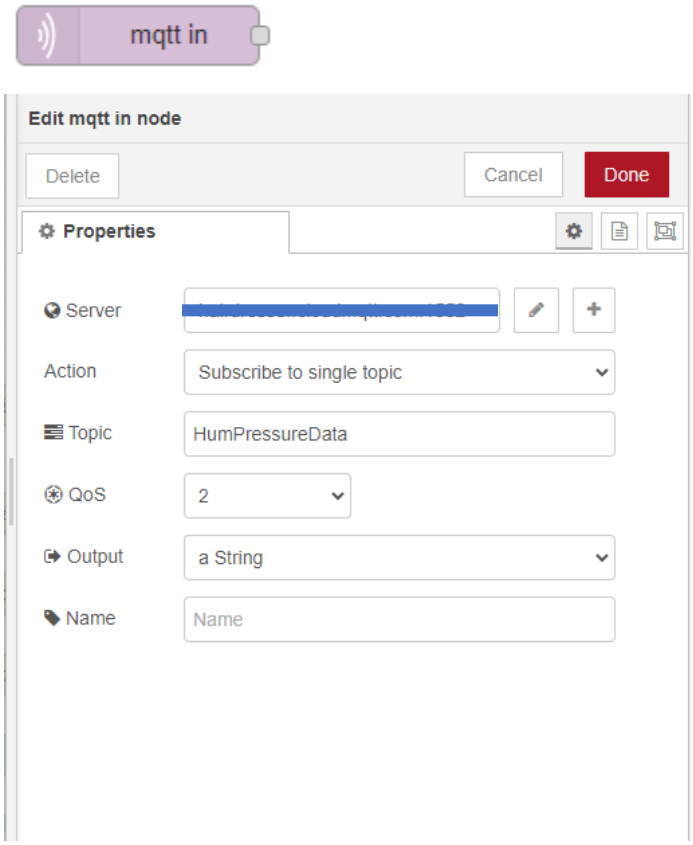


```
var input = msg.payload; // Assuming the input is in msg.payload

// Remove any leading or trailing whitespace
input = input.trim();

// Split the input string using commas as the delimiter
var values = input.split(',');

// Remove leading and trailing whitespace from each value and convert to float
values = values.map(function (value) {
    return parseFloat(value.trim());
});

msg.payload = values[1]; // Set the output payload to the array of float
values
return msg;
```

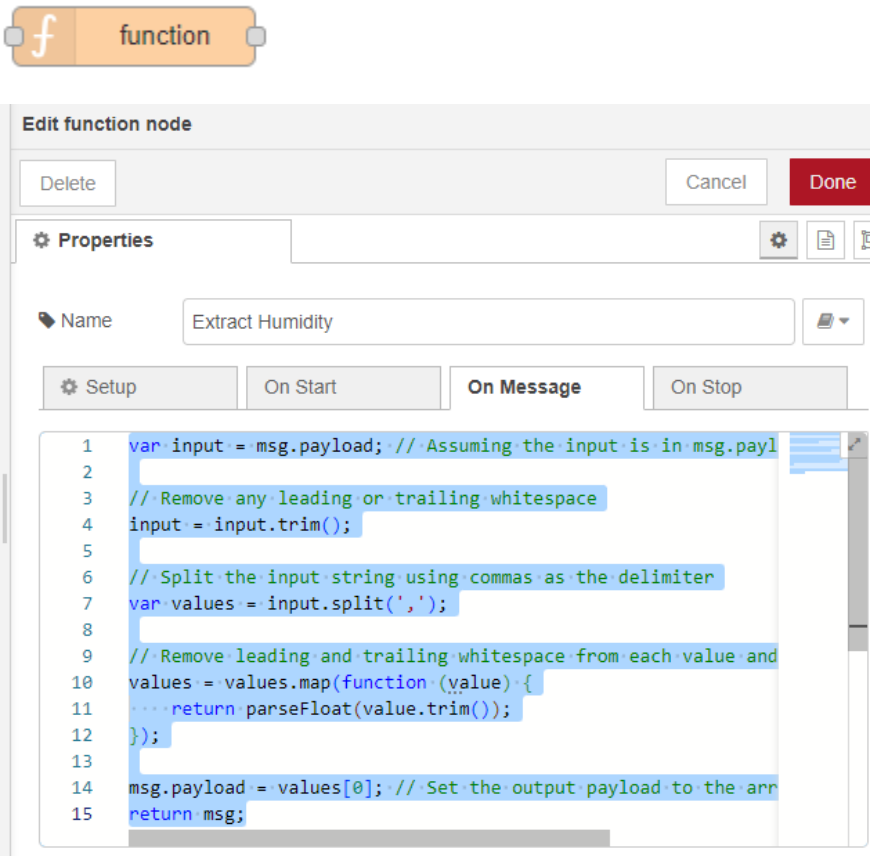
Temperature Gauge 2 : Dashboard Gauge to show the second temperature values



HumPressureData : As like temperature data, you can see the Humidity data coming through MQTT

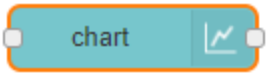


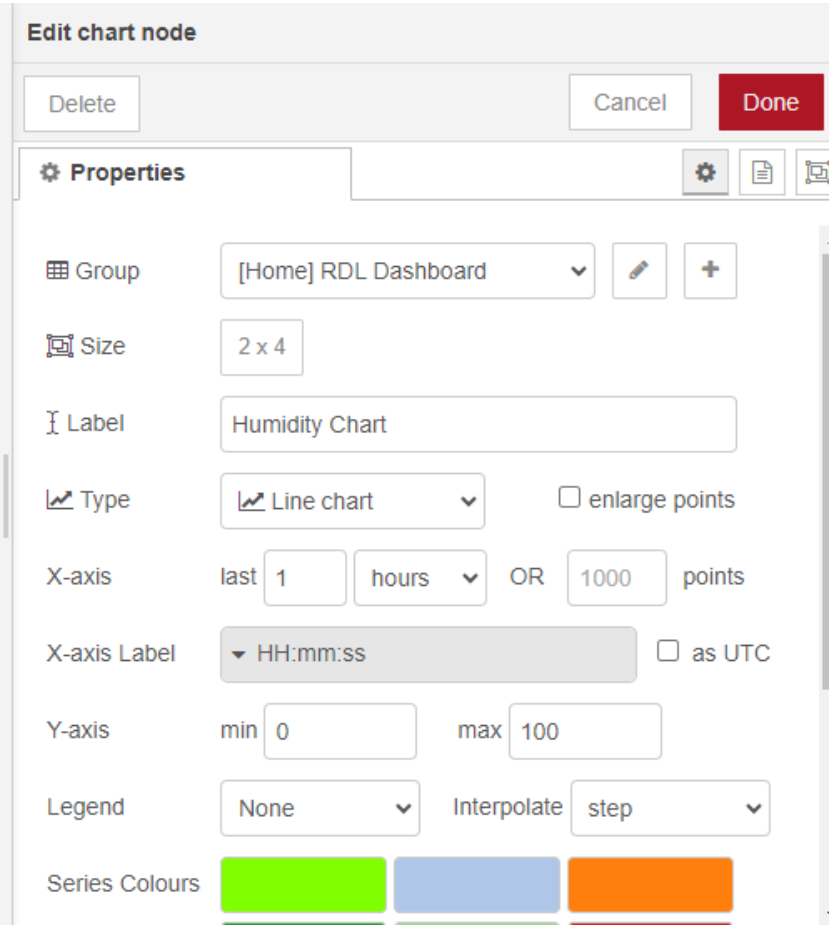
Extract Humidity : Extract the humidity values from the incoming string from MQTT



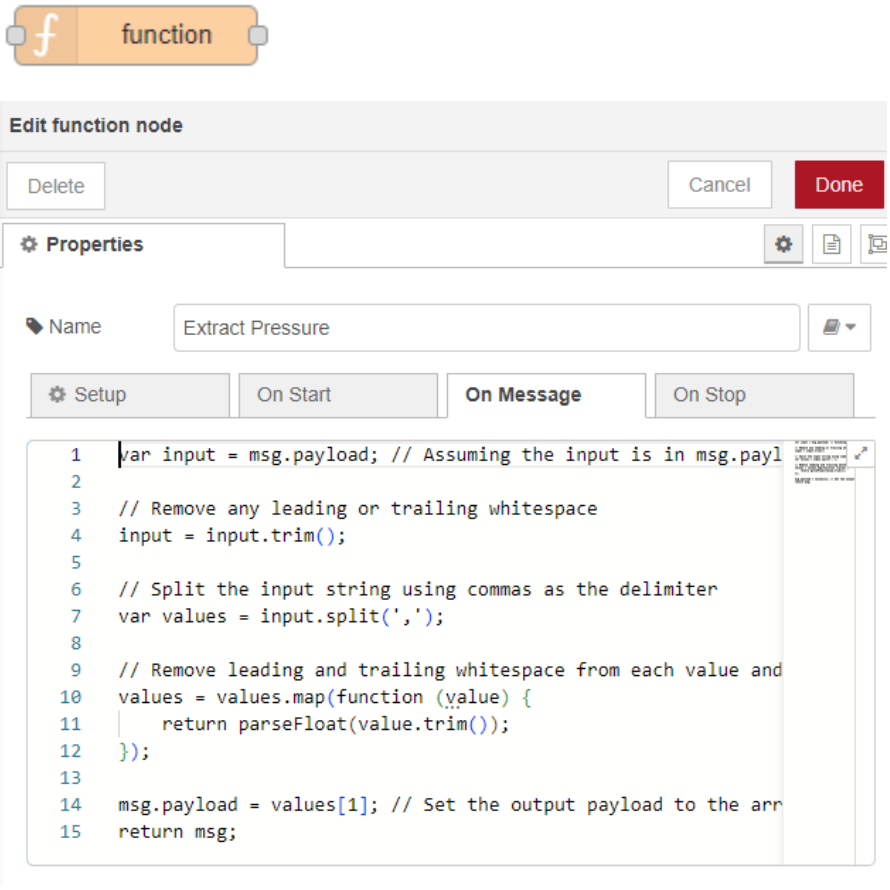
```
var input = msg.payload; // Assuming the input is in msg.payload

// Remove any leading or trailing whitespace
input = input.trim();

// Split the input string using commas as the delimiter
var values = input.split(',');

// Remove leading and trailing whitespace from each value and convert to float
values = values.map(function (value) {
    return parseFloat(value.trim());
});

msg.payload = values[0]; // Set the output payload to the array of float
values
return msg;
```

Humidity Chart : Dashboard Chart to show the humidity values



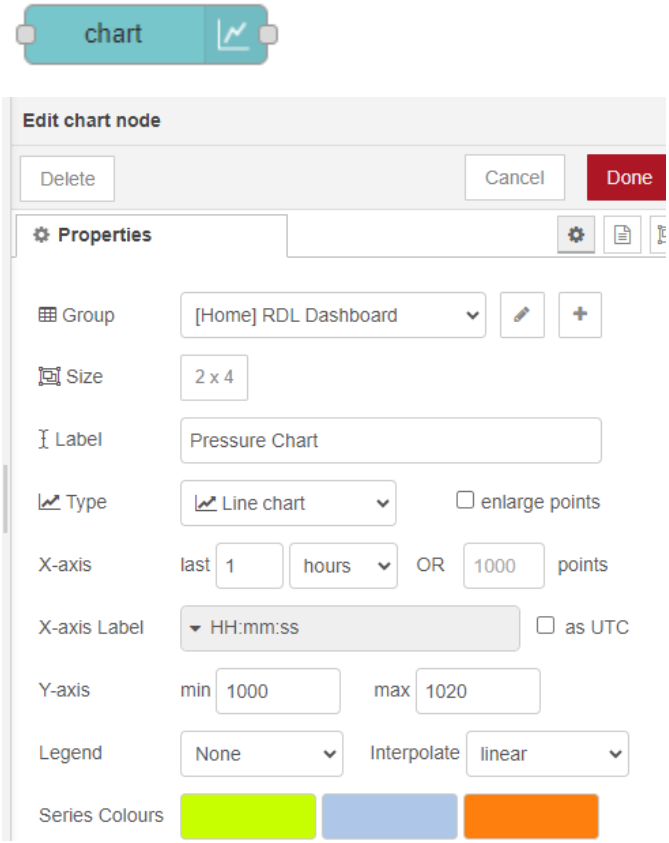


Extract Pressure : The incoming humidity and pressure values will be like XX,YY where XX is 1st humidity value and YY is the second pressure value

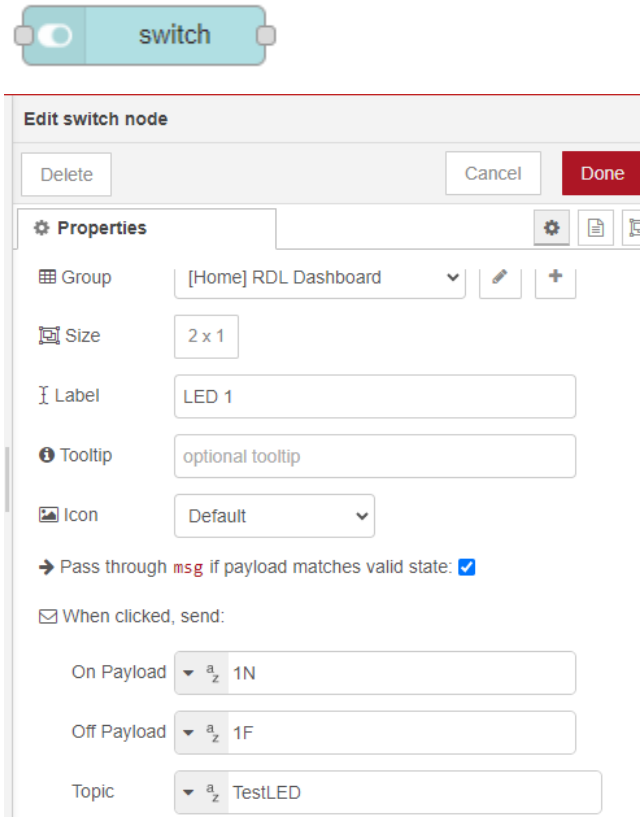



```
var input = msg.payload; // Assuming the input is in msg.payload

// Remove any leading or trailing whitespace
input = input.trim();

// Split the input string using commas as the delimiter
var values = input.split(',');

// Remove leading and trailing whitespace from each value and convert to float
values = values.map(function (value) {
    return parseFloat(value.trim());
});

msg.payload = values[1]; // Set the output payload to the array of float
values
return msg;
```

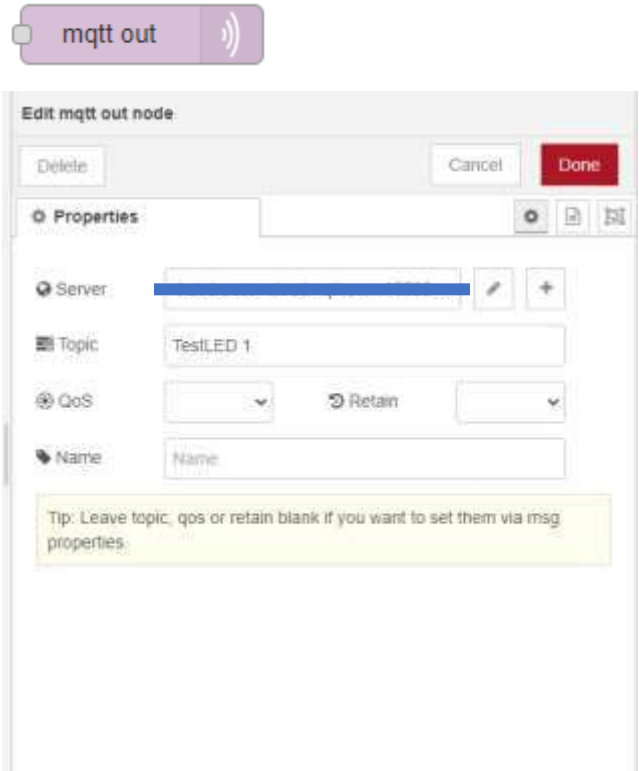
Pressure Chart : Dashboard Chart to show the pressure values



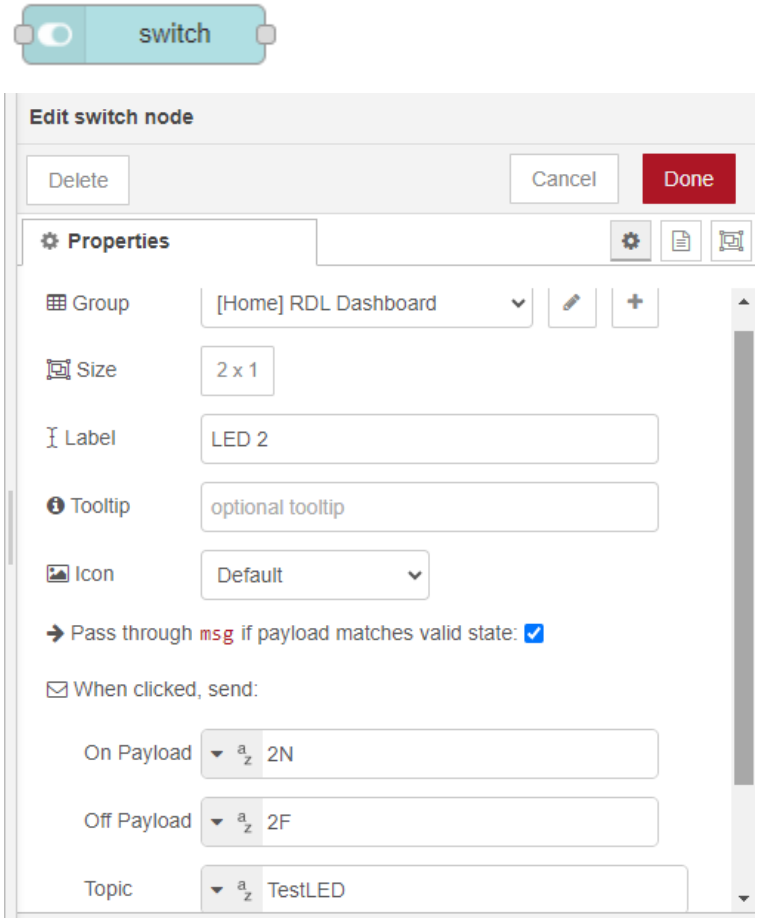
LED 1 : This is switch to turn on and off the relay 1 or led 1



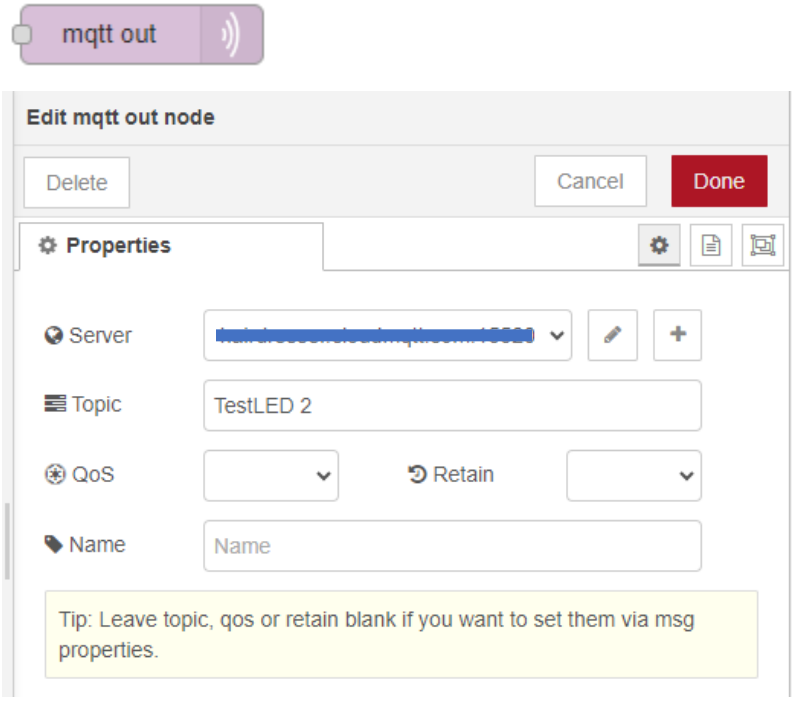
TestLed 1 : To send the switch status to MQTT Subscriber



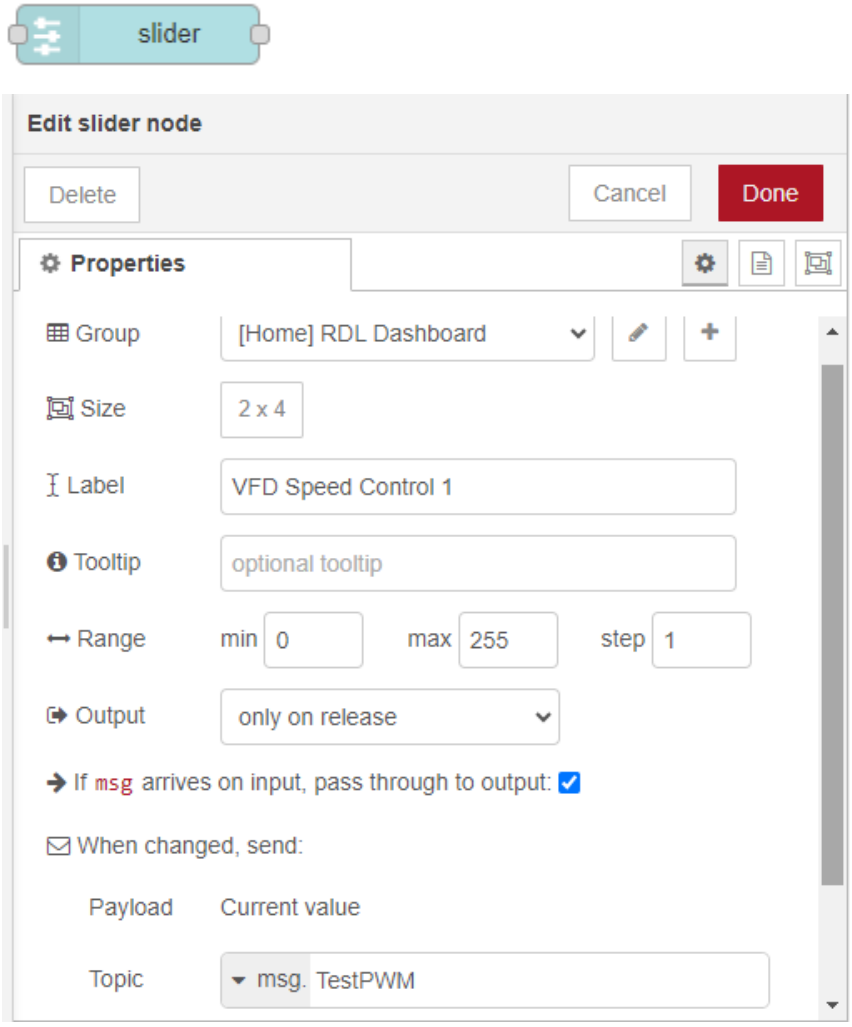
LED 2 : This is switch to turn on and off the relay 2 or led 2



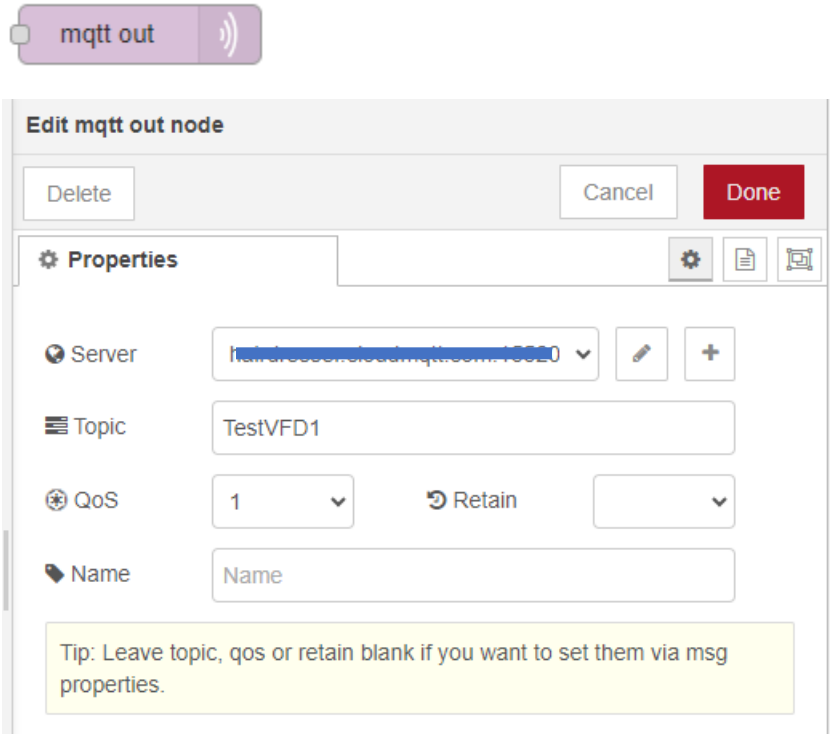
TestLed 2 : To send the switch status to MQTT Subscriber



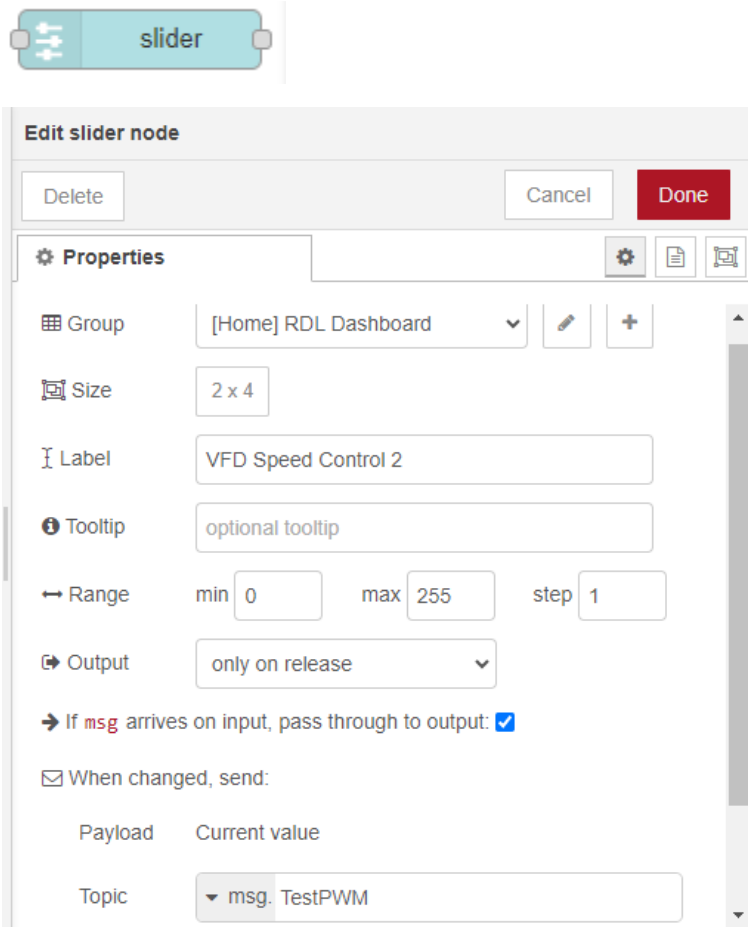
VFD Speed Control 1 : Control VFD Motor Speed



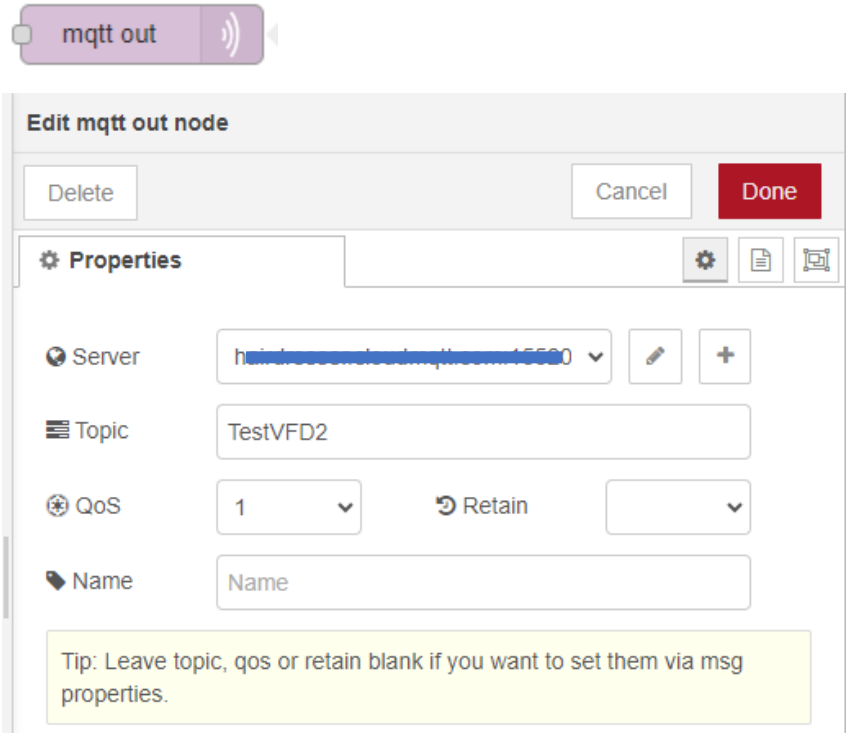
Test VFD 1 : MQTT sends data to the subscriber



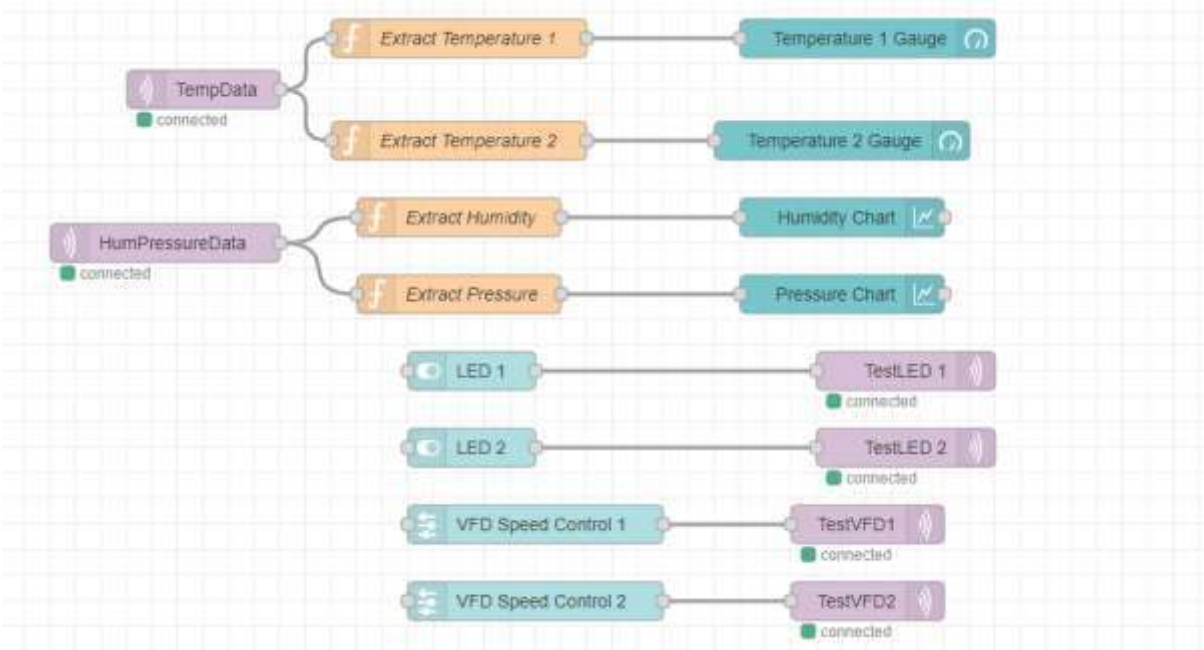
VFD Speed Control 2 : Control VFD Motor Speed



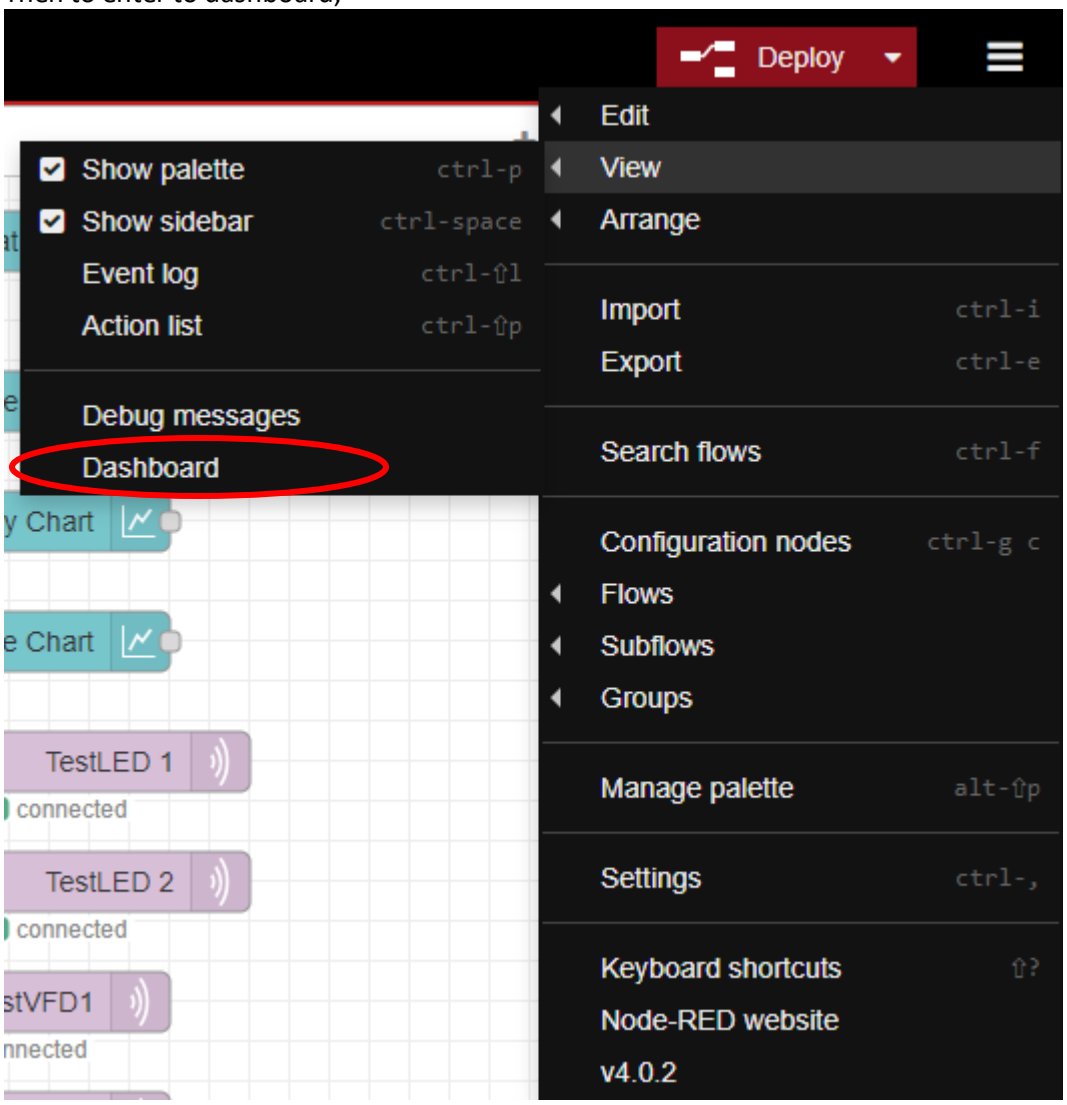
Test VFD 2 : MQTT sends data to the subscriber

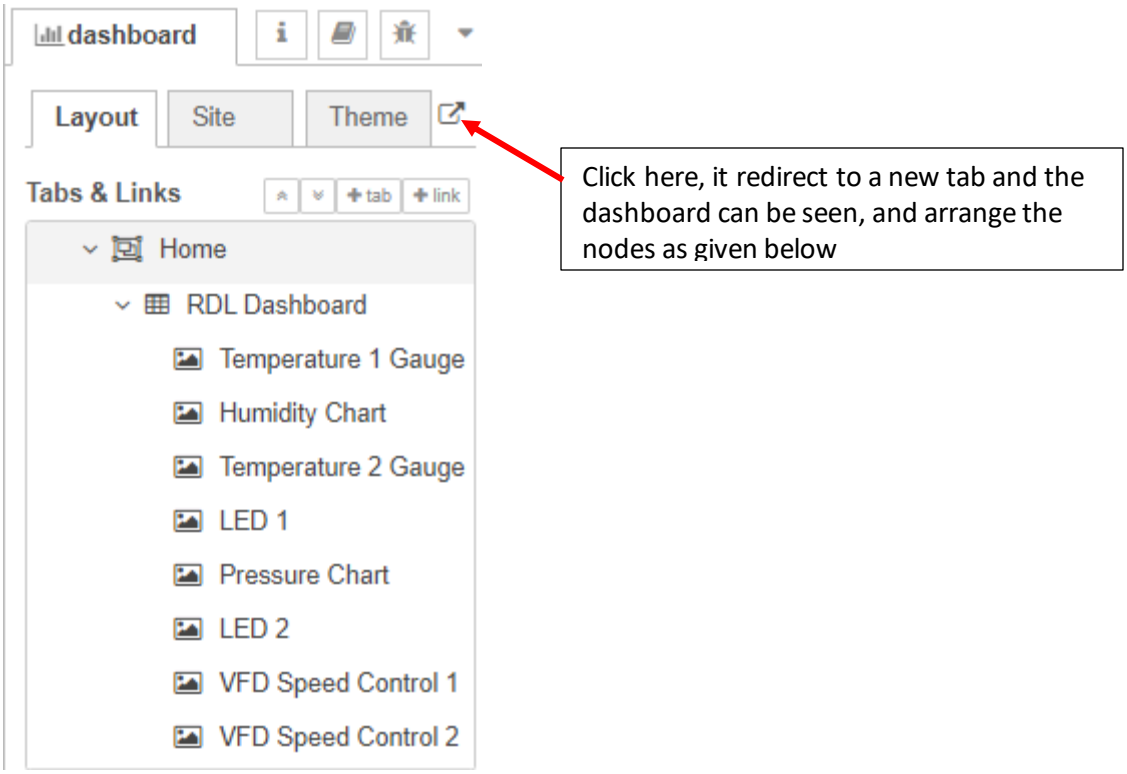


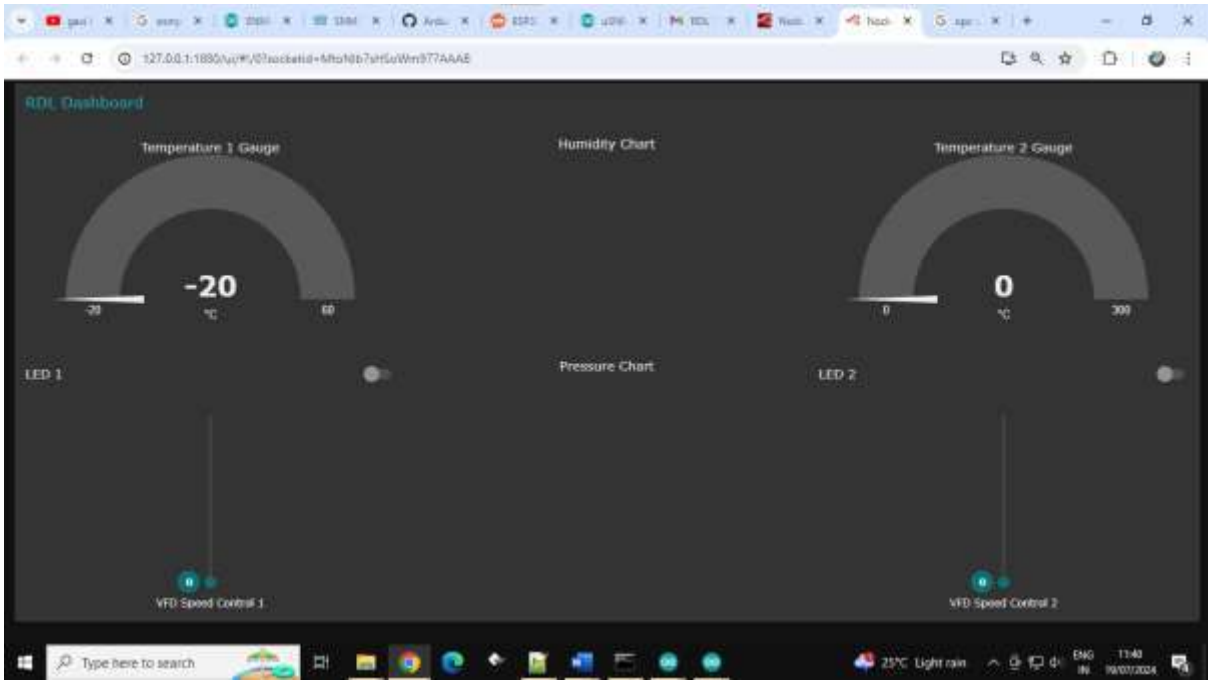
The Final connection of these created nodes will be like



Then to enter to dashboard,




dashboard
Layout
Site
Theme
Tabs & Links
+ tab
+ link
Home
RDL Dashboard
Temperature 1 Gauge
Humidity Chart
Temperature 2 Gauge
LED 1
Pressure Chart
LED 2
VFD Speed Control 1
VFD Speed Control 2
Click here, it redirect to a new tab and the dashboard can be seen, and arrange the nodes as given below


RDL Dashboard
Temperature 1 Gauge
-20
°C
Humidity Chart
Temperature 2 Gauge
0
°C
LED 1
Pressure Chart
LED 2
VFD Speed Control 1
VFD Speed Control 2

ESP32 Code :

```
#include <WiFi.h>
#include <PubSubClient.h>

// Replace these with your network credentials
const char* ssid = "yourWiFiSSID";
const char* password = "yourWiFiPASSWORD";

// MQTT Broker details
const char* mqtt_server = "yourMQTTSERVER";
const int mqtt_port = yourMQTTPORT;
const char* mqtt_user = "yourMQTTUSERNAME"; // For public brokers, you can
usually leave these empty
const char* mqtt_password = "yourMTTPASSWORD";

// Topics to subscribe and publish to
const char* subscribe_topic = "TestData";
const char* pubTopic_TM = "TempData";
const char* pubTopic_HP = "HumPressureData";

WiFiClient espClient;
PubSubClient client(espClient);

// Function to connect to WiFi
void setup_wifi() {
  delay(10);
  Serial.println();
  Serial.print("Connecting to ");
  Serial.println(ssid);

  WiFi.begin(ssid, password);

  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }

  Serial.println("");
  Serial.println("WiFi connected");
  Serial.println("IP address: ");
  Serial.println(WiFi.localIP());
}

// Callback function for when a message is received
void callback(char* topic, byte* message, unsigned int length) {
  Serial.print("Message arrived on topic: ");
  Serial.print(topic);
  Serial.print(". Message: ");

  // Convert message to String
  String messageString;
  for (int i = 0; i < length; i++) {
    messageString += (char)message[i];
  }
  Serial.println(messageString);
  if (messageString.startsWith("1N")) {
    digitalWrite(15, HIGH);
  }
  else if (messageString.startsWith("1F")) {
    digitalWrite(15, LOW);
```

```
  }
  else if (messageString.startsWith("2N")) {
    digitalWrite(13, HIGH);
  }
  else if (messageString.startsWith("2F")) {
    digitalWrite(13, LOW);
  }
}

void reconnect() {
  // Loop until we're reconnected
  while (!client.connected()) {
    Serial.print("Attempting MQTT connection...");
    // Attempt to connect
    if (client.connect("ESP32Client", mqtt_user, mqtt_password)) {
      Serial.println("connected");
      // Subscribe to topic
      client.subscribe("TestLED");
      client.subscribe("TestVFD1");
      client.subscribe("TestVFD2");
    } else {
      Serial.print("failed, rc=");
      Serial.print(client.state());
      Serial.println(" try again in 5 seconds");
      // Wait 5 seconds before retrying
      delay(5000);
    }
  }
}

void setup() {
  Serial.begin(115200);
  pinMode(13, OUTPUT);
  pinMode(15, OUTPUT);
  digitalWrite(13, LOW);
  digitalWrite(15, LOW);
  setup_wifi();
  client.setServer(mqtt_server, mqtt_port);
  client.setCallback(callback);
}

void loop() {
  if (!client.connected()) {
    reconnect();
  }
  client.loop();

  //client.publish(publish_topic, msg.c_str());

  // Publish a message every 5 seconds
  static unsigned long lastMsg = 0;
  unsigned long now = millis();
  if (now - lastMsg > 3000) {
    lastMsg = now;
    randomSeed(analogRead(0)); // Seed the random number generator
    // Generate a random float between 24.0 and 32.0
    float minTemp1 = -20.0;
    float maxTemp1 = 60.0;
    float TempValue1 = minTemp1 + (float(random(10000)) / 10000.0) *
(maxTemp1 - minTemp1);
    float minTemp2 = 0.0;
```

```
    float maxTemp2 = 300.0;
    float TempValue2 = minTemp2 + (float(random(10000)) / 10000.0) *
(maxTemp2 - minTemp2);
    float minPressure = 1010.0;
    float maxPressure = 1015.0;
    float PressureValue = minPressure + (float(random(10000)) / 10000.0) *
(maxPressure - minPressure);
    char sensorData[20];
    char HumPressureData[20];
    sprintf(HumPressureData, "%d,%0.2f", random(0, 100), PressureValue);
    sprintf(sensorData, "%0.2f,%0.2f", TempValue1, TempValue2);
    Serial.println(sensorData);
    client.publish(pubTopic_TM, sensorData);
    client.publish(pubTopic_HP, HumPressureData);
  }
}
```