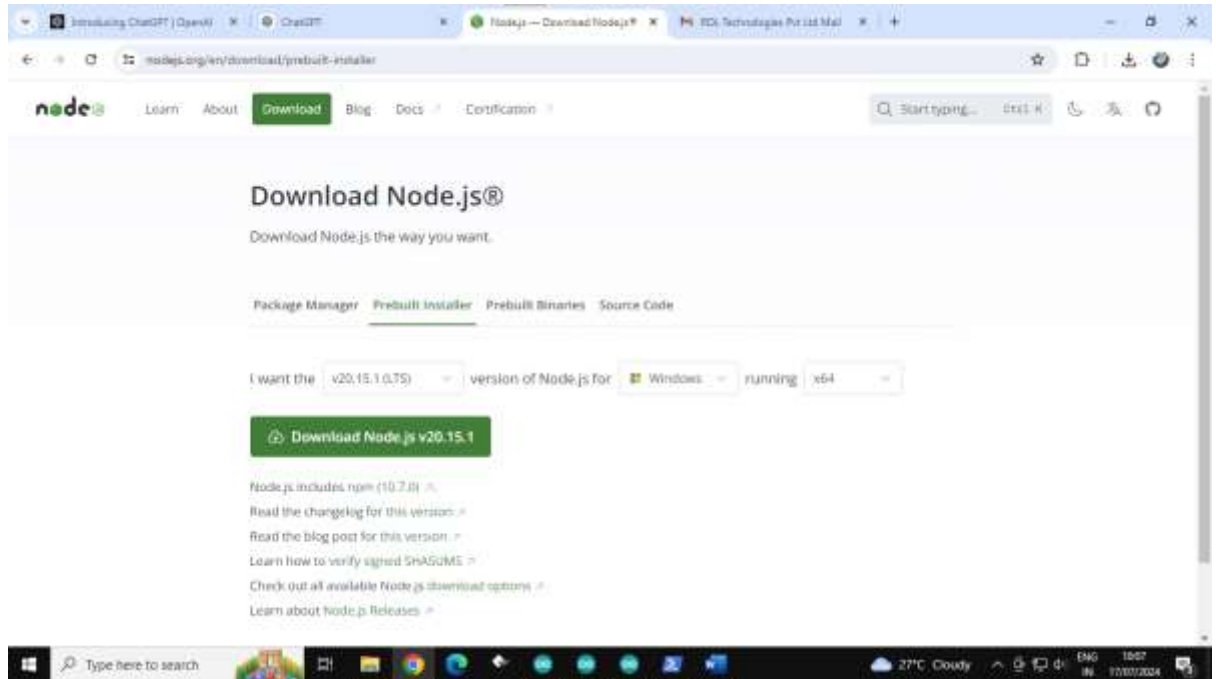



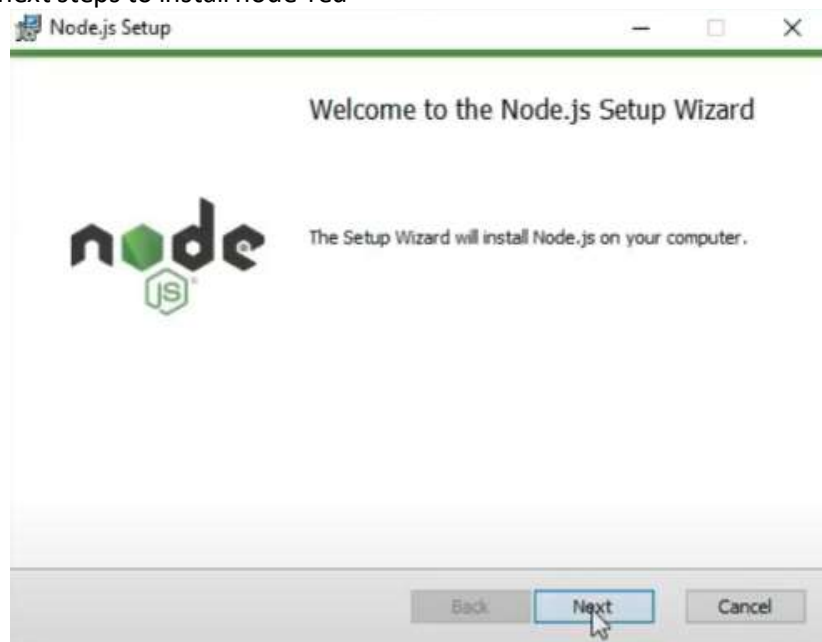
# Node-Red Implementation

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.



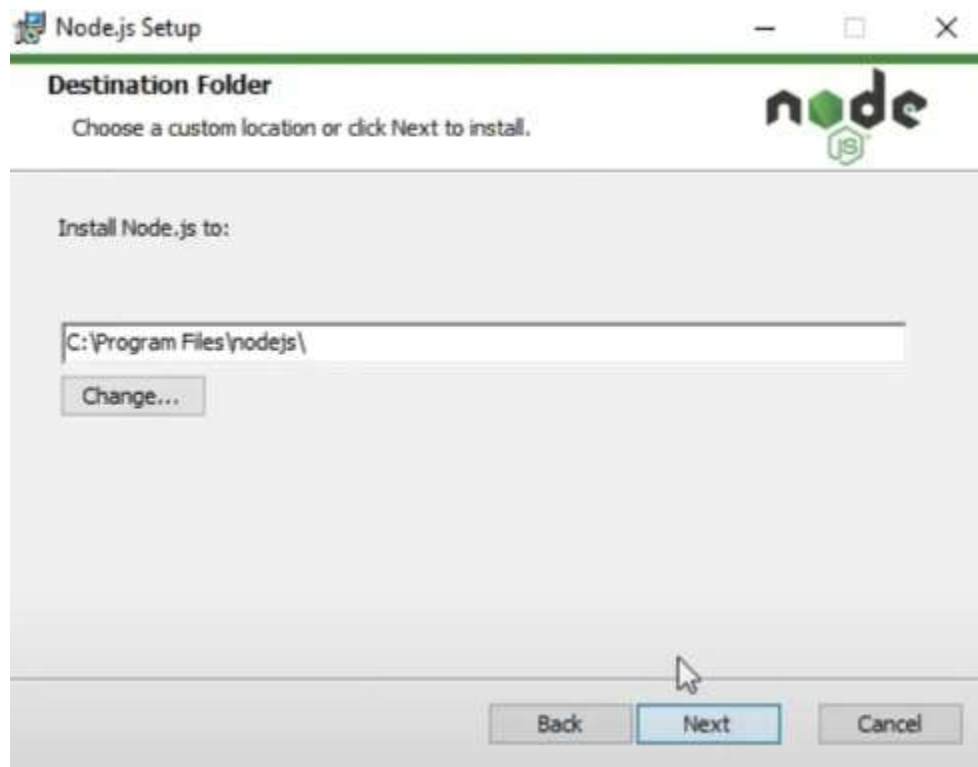
Here click on  or latest version of the software or if you want the same software I used, then click on this [link](#).

Follow the next steps to install node-red

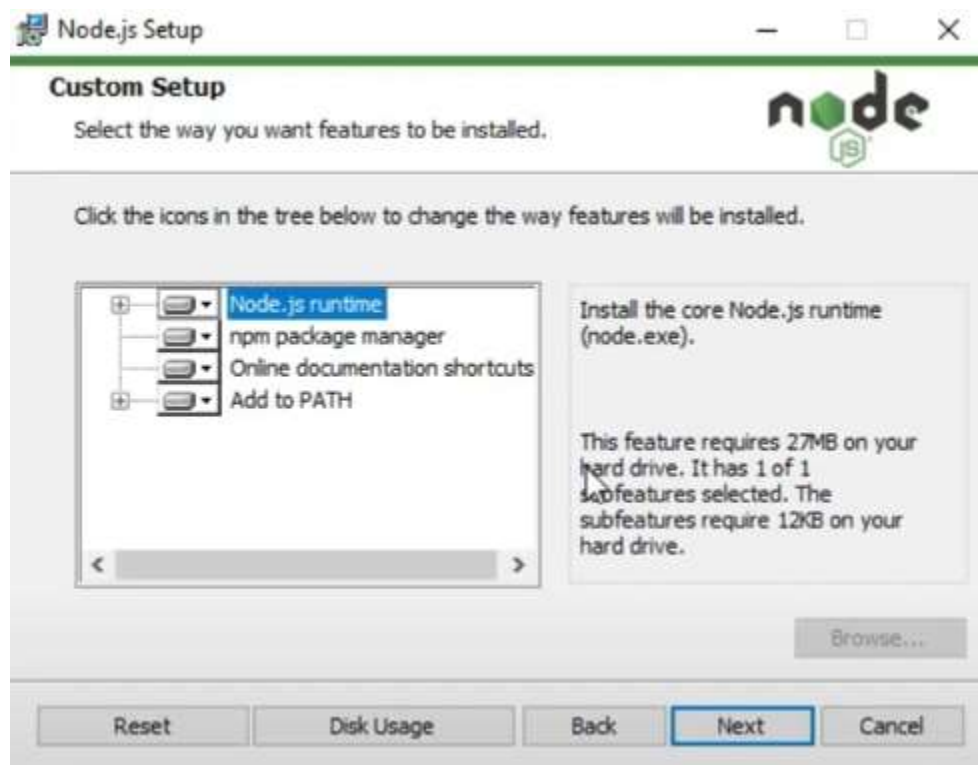


Click on Next.

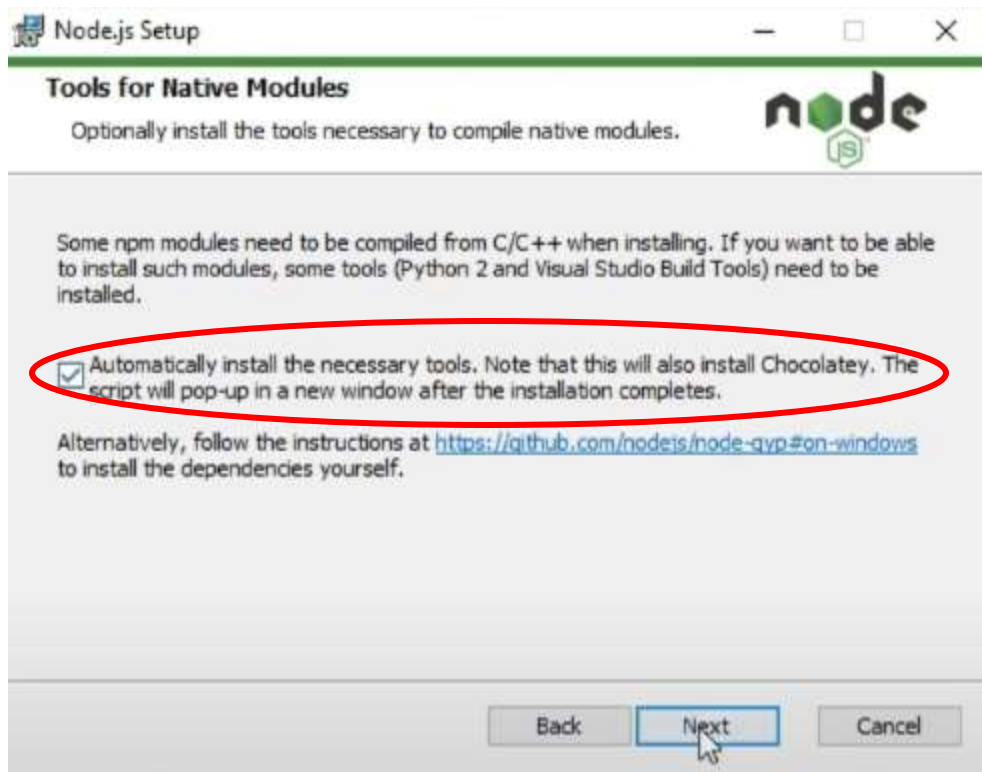
<https://highvoltages.co/iot-internet-of-things/mqtt/mqtt-in-nodered-and-mqtt-dashboard/>



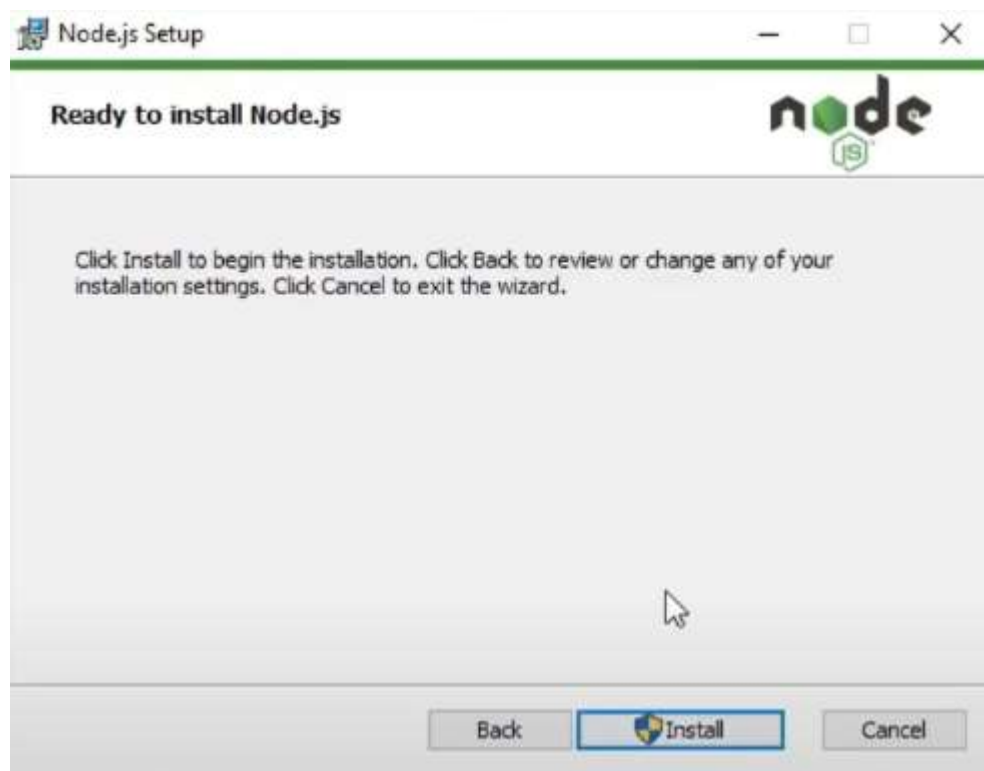
Set destination folder and click on Next



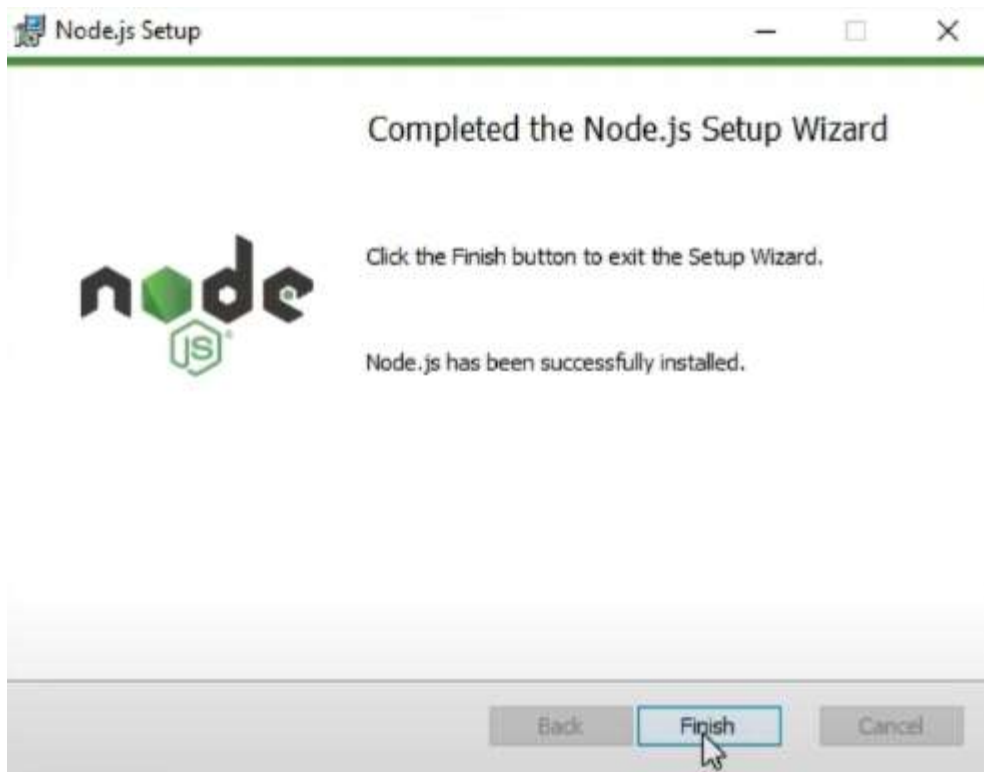
Click on Next



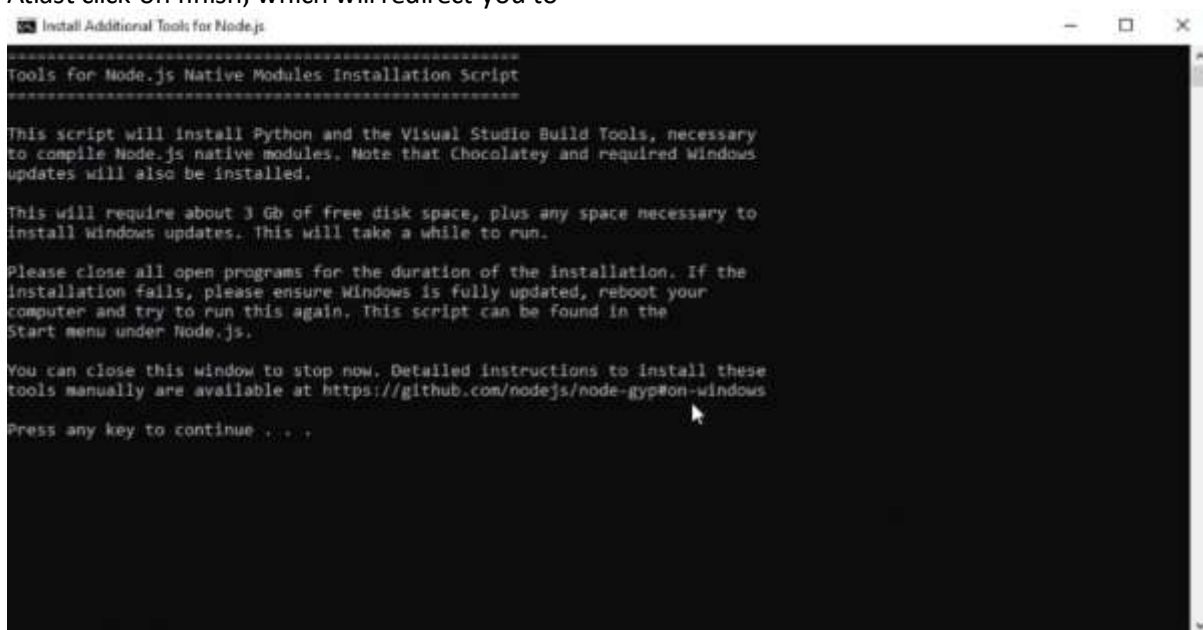
Enable the checkbox and click on Next



Next click on Install



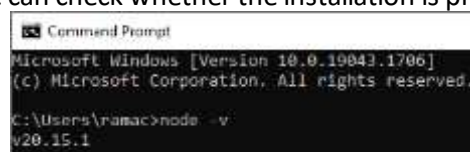
Atlast click on finish, which will redirect you to



This command prompt, 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,

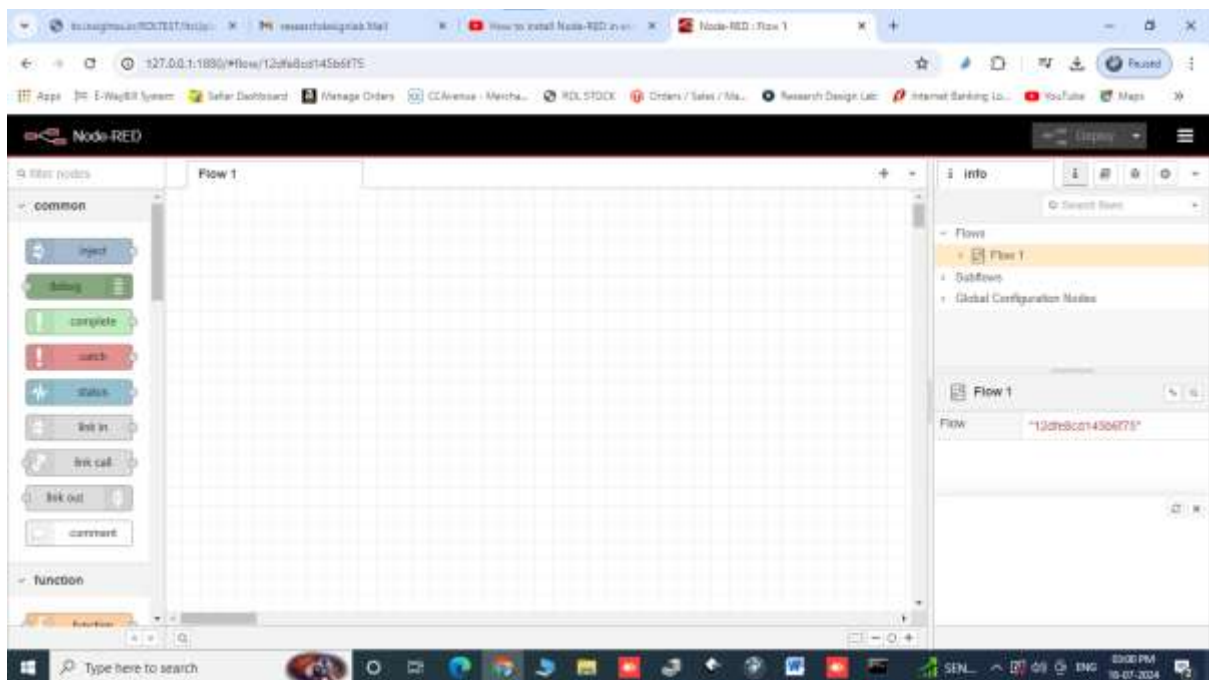
For that we need to open command prompt and put the below command

```
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

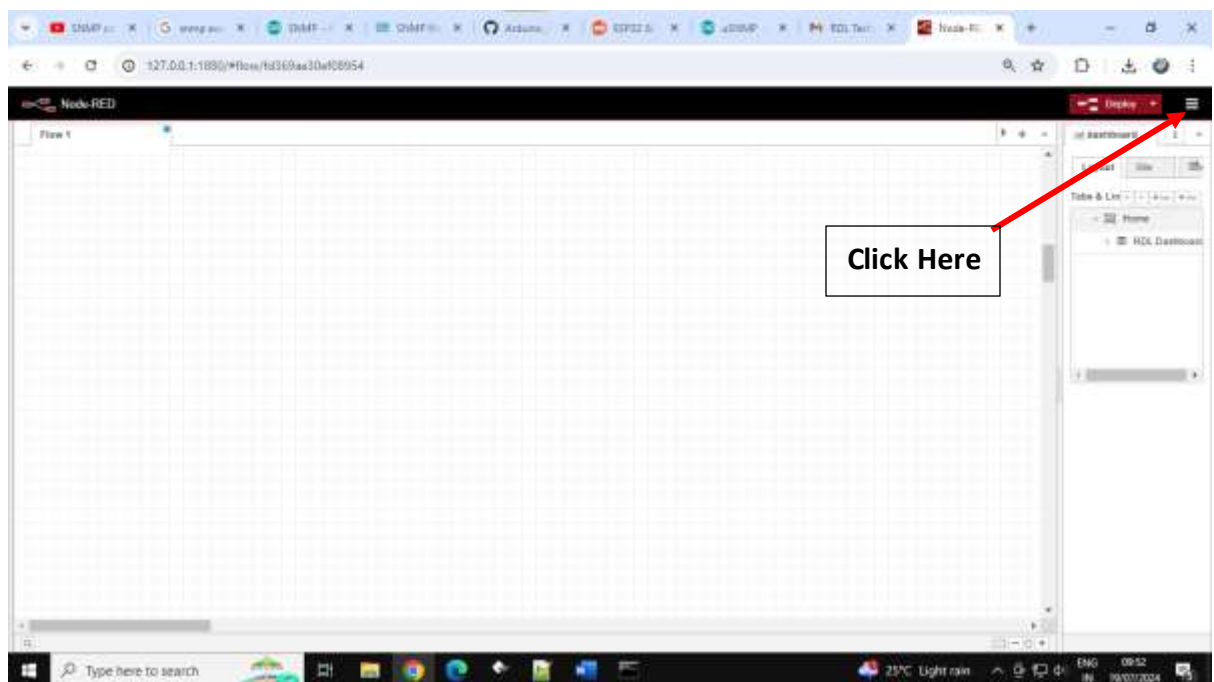
```
18 Jul 14:57:28 - [info] Server now running at http://127.0.0.1:1880/
18 Jul 14:57:28 - [warn] Encrypted credentials not found
18 Jul 14:57:28 - [info] Starting flows
18 Jul 14:57:28 - [info] Started flows
```

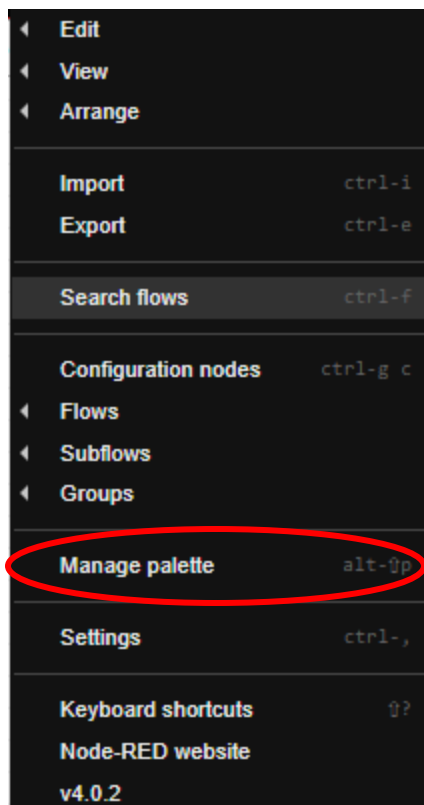
You can simply put the running server url in web browser, it will open the node-red page.



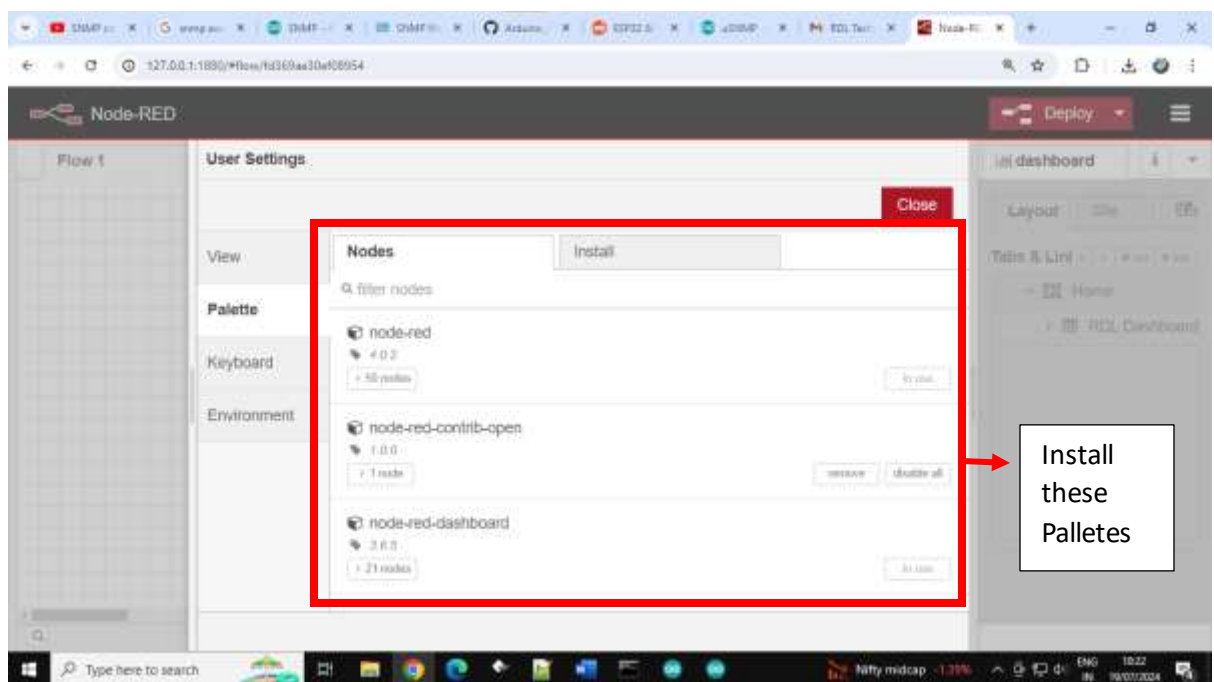
## 2. Sensor Values and Relay Control using MQTT Connection: (for your reference [Link](#))

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





Click on manage palette

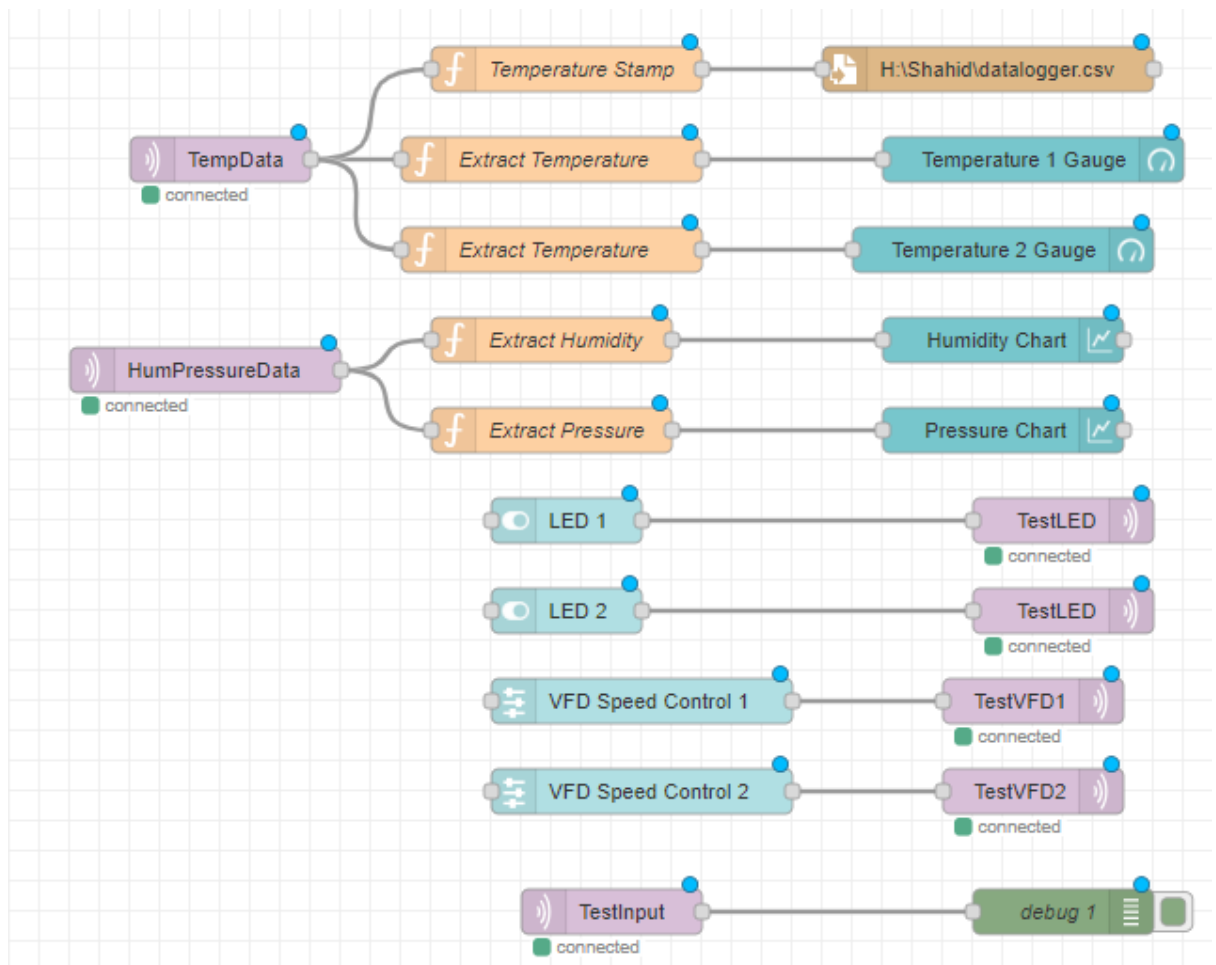


Install these Palletes

node-red

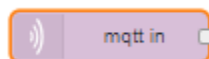
node-red-contrib-open

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.



Edit mqtt in node

Delete
Cancel
Done

⚙️

Properties

⚙️
📄
🖼️

🌐
Server

✎️
+

🔧
Action

Subscribe to single topic
▼

📋
Topic

TempData

🔊
QoS

2
▼

🔗
Output

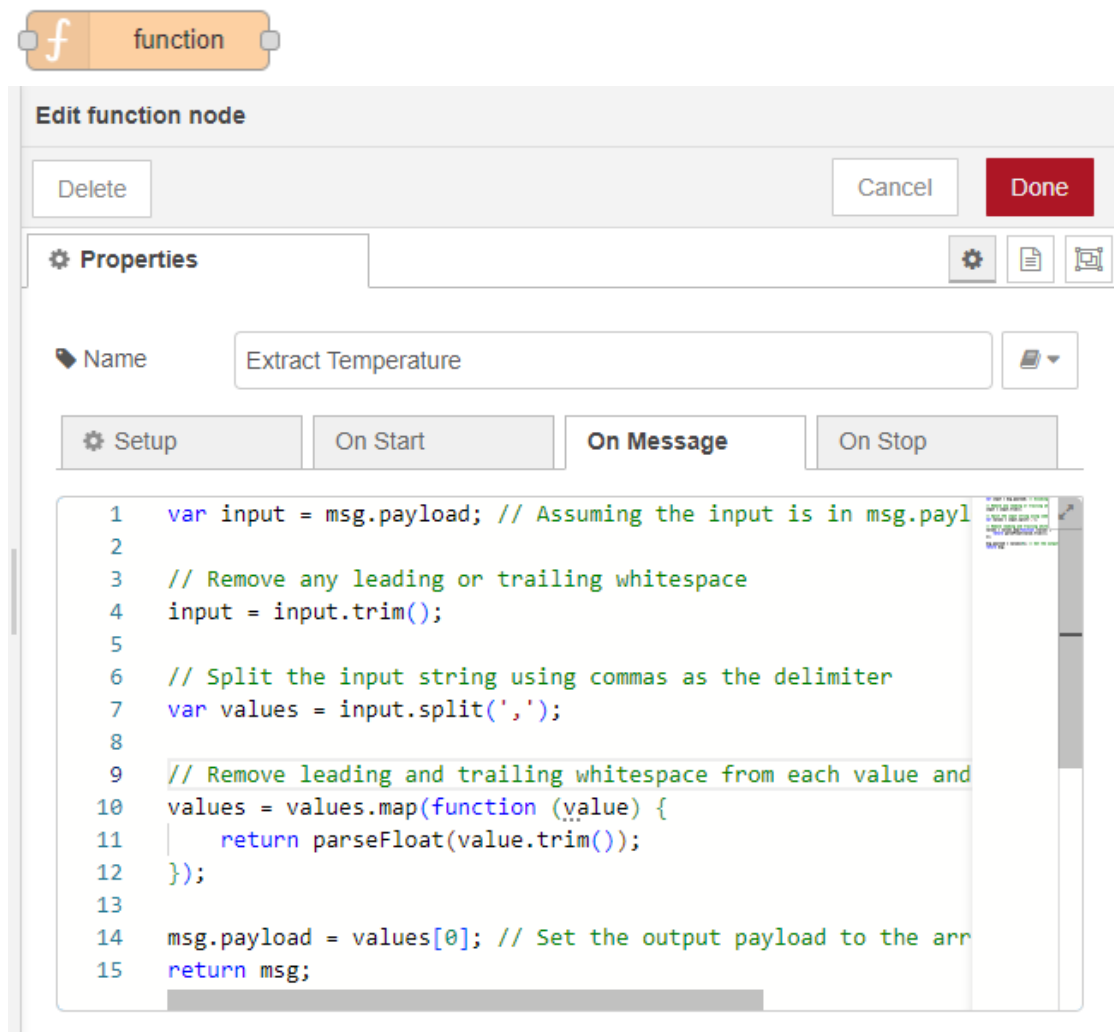
a String
▼

🏷️
Name

Name



**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

**Edit gauge node**

Delete Cancel Done

**Properties**

Group [Home] RDL Dashboard

Size 2 x 4

Type Gauge

Label Temperature 1 Gauge

Value format {{value}}

Units °C

Range min -20 max 60

Colour gradient

Sectors -20 ... optional ... optional ... 60

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

**Edit function node**

Delete Cancel Done

**Properties**

Name Extract Temperature 2

Setup On Start On Message On Stop

```
1 var input = msg.payload; // Assuming the input is in msg.payload
2
3 // Remove any leading or trailing whitespace
4 input = input.trim();
5
6 // Split the input string using commas as the delimiter
7 var values = input.split(',');
8
9 // Remove leading and trailing whitespace from each value and
10 values = values.map(function (value) {
11   return parseFloat(value.trim());
12 });
13
14 msg.payload = values[1]; // Set the output payload to the arr
15 return msg;
```

```

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



Delete
Cancel
Done

Properties

Group
[Home] RDL Dashboard

Size
2 x 4

Type
Gauge

Label
Temperature 2 Gauge

Value format
{{value}}

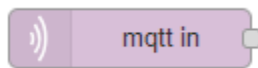
Units
°C

Range
min 0 max 300

Colour gradient

Sectors
0 ... optional ... optional ... 300

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



**Edit mqtt in node**

Delete Cancel Done

**Properties**

Server

Action

Topic

QoS

Output

Name

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



**Edit function node**

Delete Cancel Done

**Properties**

Name

Setup On Start On Message On Stop

```
1 var input = msg.payload; // Assuming the input is in msg.payload
2
3 // Remove any leading or trailing whitespace
4 input = input.trim();
5
6 // Split the input string using commas as the delimiter
7 var values = input.split(',');
8
9 // Remove leading and trailing whitespace from each value and
10 values = values.map(function(value) {
11   return parseFloat(value.trim());
12 });
13
14 msg.payload = values[0]; // Set the output payload to the array
15 return msg;
```

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

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

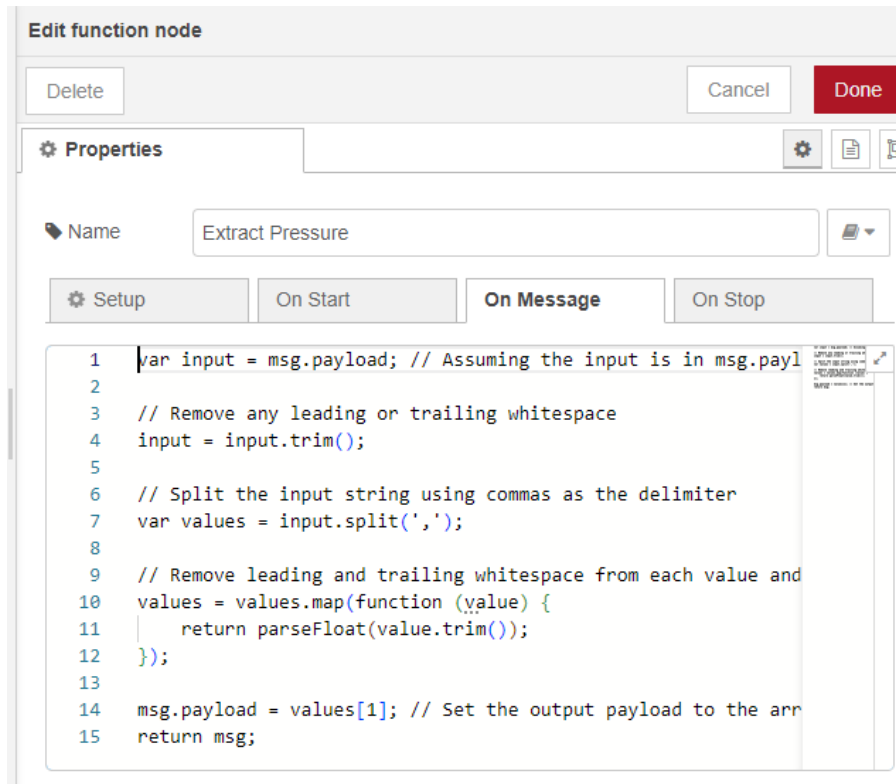
// Split the input string using commas as the delimiter
var values = input.split(',');

// 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;
```



**Extract Pressure** : The incoming humidity and pressure values will be like XX,YY where XX is 1<sup>st</sup> 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());
```

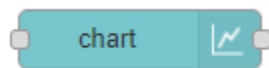
```
});
```

```
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



**Edit chart node**

Delete Cancel Done

**Properties**

Group [Home] RDL Dashboard

Size 2 x 4

Label Pressure Chart

Type Line chart ☐ enlarge points

X-axis last 1 hours OR 1000 points

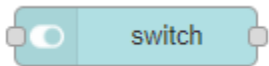
X-axis Label HH:mm:ss ☐ as UTC

Y-axis min 1000 max 1020

Legend None Interpolate linear

Series Colours

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



**Edit switch node**

Delete Cancel Done

**Properties**

Group [Home] RDL Dashboard

Size 2 x 1

Label LED 1

Tooltip optional tooltip

Icon Default

→ Pass through msg if payload matches valid state: ☒

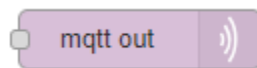
☒ When clicked, send:

On Payload a<sub>z</sub> 1N

Off Payload a<sub>z</sub> 1F

Topic a<sub>z</sub> TestLED

**TestLED 1** : To send the switch status to MQTT Subscriber



Edit mqtt out node

Delete Cancel Done

Properties

Server [redacted] [edit] [add]

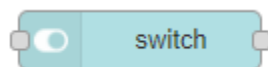
Topic TestLED 1

QoS [dropdown] Retain [dropdown]

Name Name

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

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



Edit switch node

Delete Cancel Done

Properties

Group [Home] RDL Dashboard [dropdown] [edit] [add]

Size 2 x 1

Label LED 2

Tooltip optional tooltip

Icon Default [dropdown]

→ Pass through msg if payload matches valid state: ☒

☒ When clicked, send:

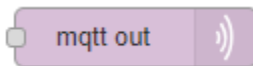
On Payload [dropdown] 2N

Off Payload [dropdown] 2F

Topic [dropdown] TestLED



**TestLed 2** : To send the switch status to MQTT Subscriber



**Edit mqtt out node**

Delete Cancel Done

**Properties**

Server [redacted] [edit] [add]

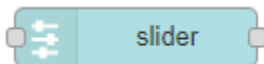
Topic TestLED 2

QoS [dropdown] Retain [dropdown]

Name Name

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

**VFD Speed Control 1** : Control VFD Motor Speed



**Edit slider node**

Delete Cancel Done

**Properties**

Group [Home] RDL Dashboard [edit] [add]

Size 2 x 4

Label VFD Speed Control 1

Tooltip optional tooltip

Range min 0 max 255 step 1

Output only on release [dropdown]

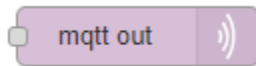
➔ If msg arrives on input, pass through to output: ☒

☑ When changed, send:

Payload Current value

Topic msg. TestPWM

## Test VFD 1 : MQTT sends data to the subscriber



**Edit mqtt out node**

Delete Cancel Done

**Properties**

Server

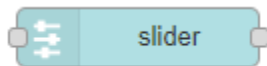
Topic

QoS  Retain

Name

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

## VFD Speed Control 2 : Control VFD Motor Speed



**Edit slider node**

Delete Cancel Done

**Properties**

Group

Size

Label

Tooltip

Range min  max  step

Output

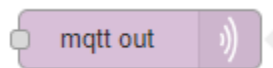
☒ If msg arrives on input, pass through to output: ☒

☒ When changed, send:

Payload

Topic

## Test VFD 2 : MQTT sends data to the subscriber



**Edit mqtt out node**

Delete Cancel Done

**Properties**

Server

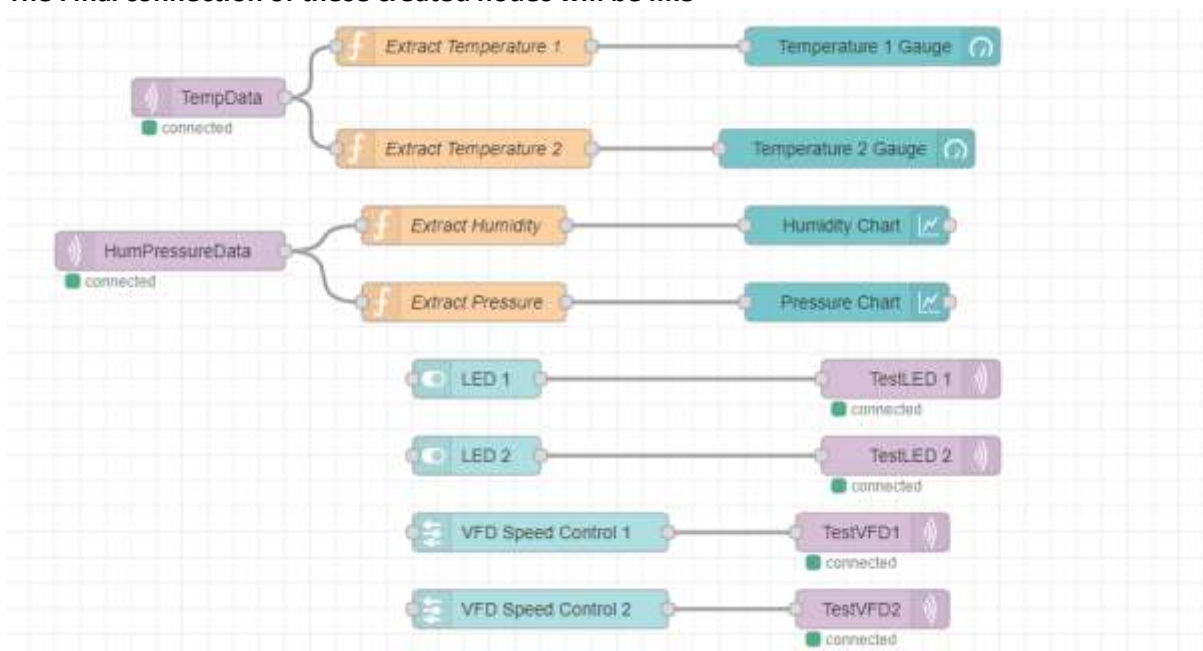
Topic

QoS  Retain

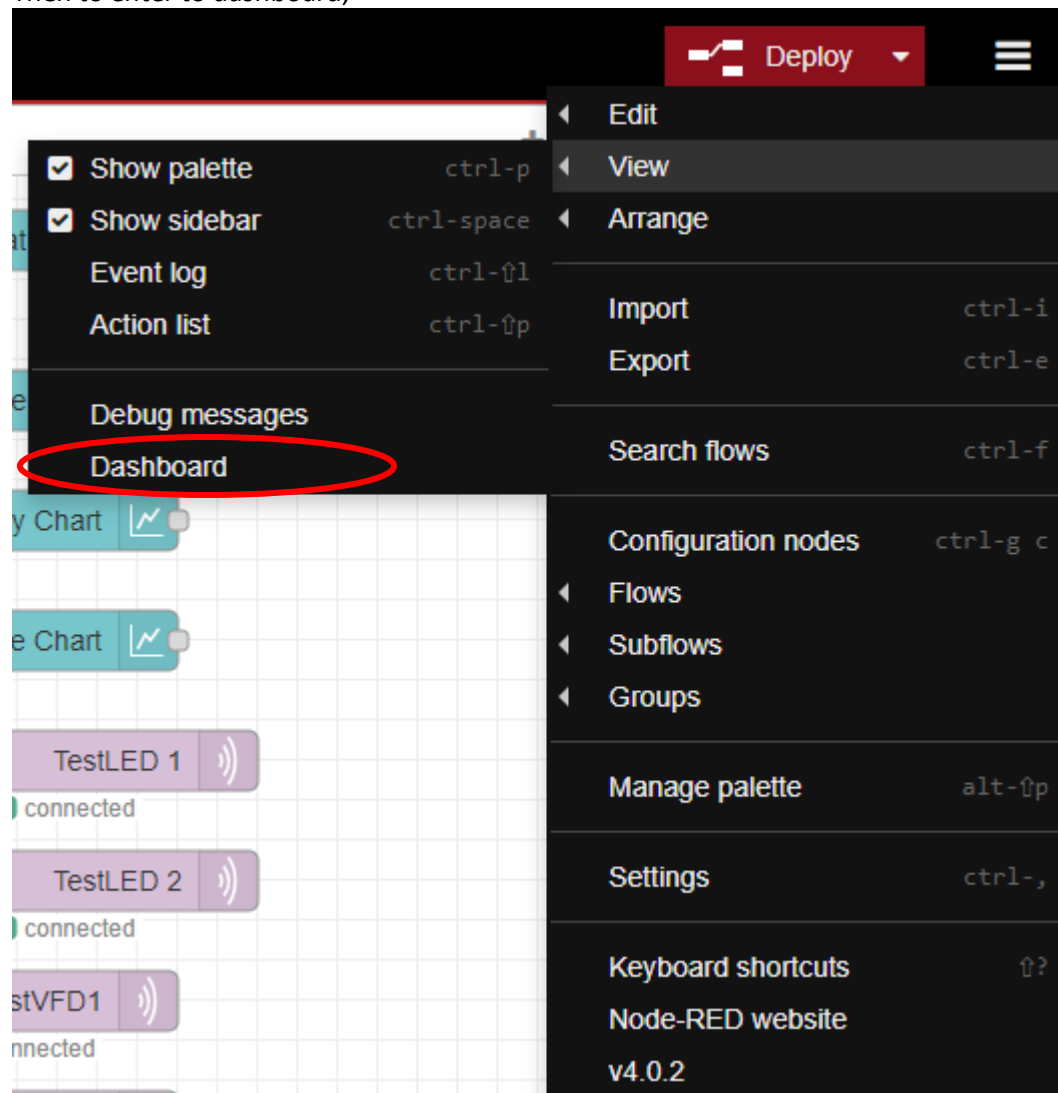
Name

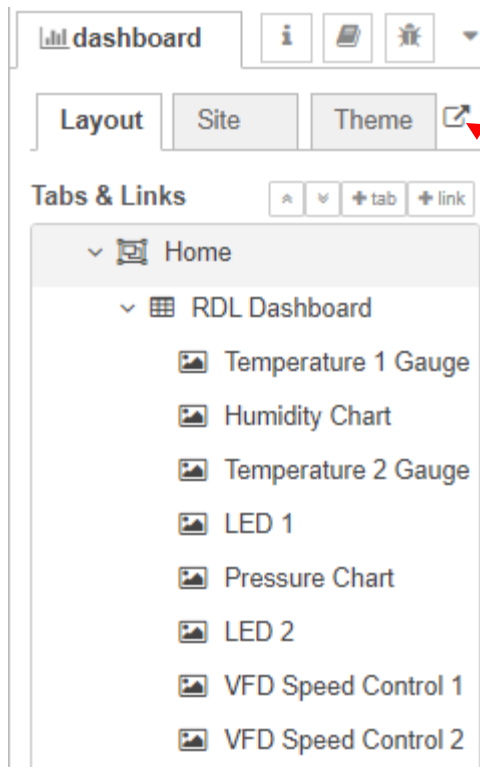
Tip: Leave topic, qos or retain blank if you want to set them via msg properties.

## The Final connection of these created nodes will be like

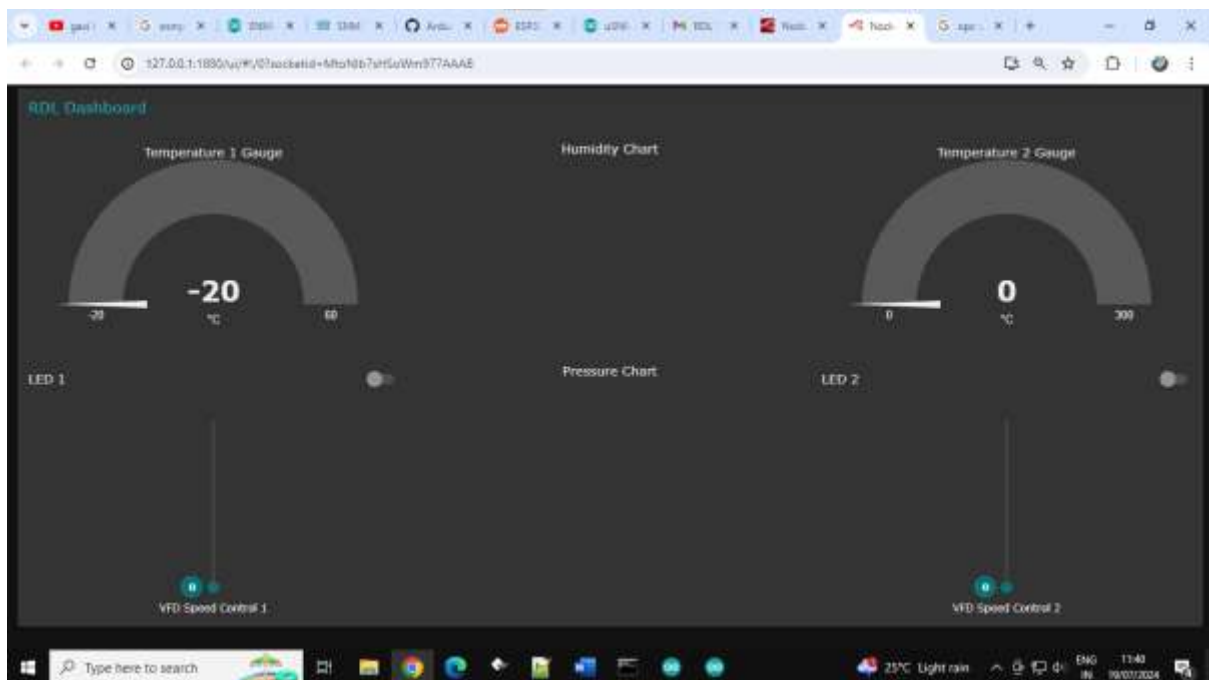


Then to enter to dashboard,





Click here, it redirect to a new tab and the dashboard can be seen, and arrange the nodes as given below



## 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 = "yourMQTTPASSWORD";

// 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);
}
}

```