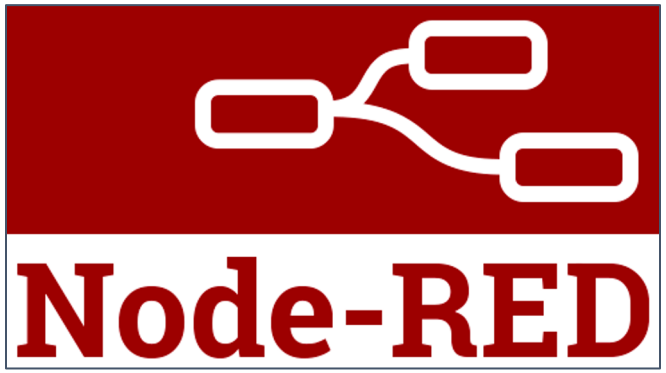
NodeRED RoboCamp:

From Hero to Superhero

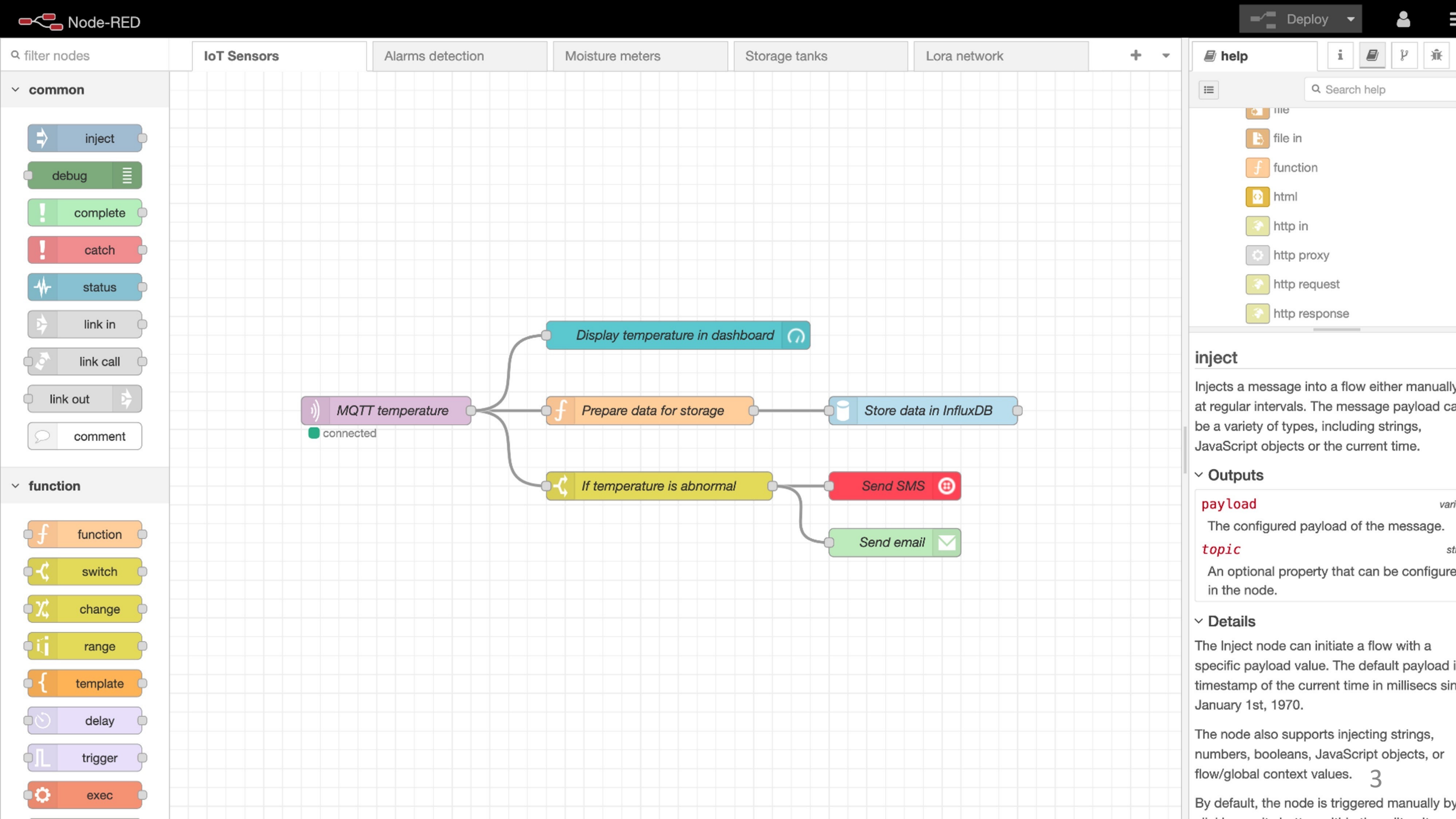
Building IoT applications with NodeMCU and NodeRED



Introduction to Node-RED

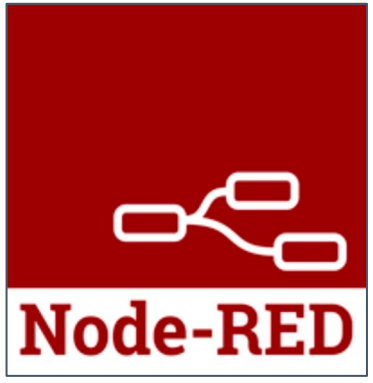
Web-based Visual Programming Tools

Wiring of the Internet of Things Programming

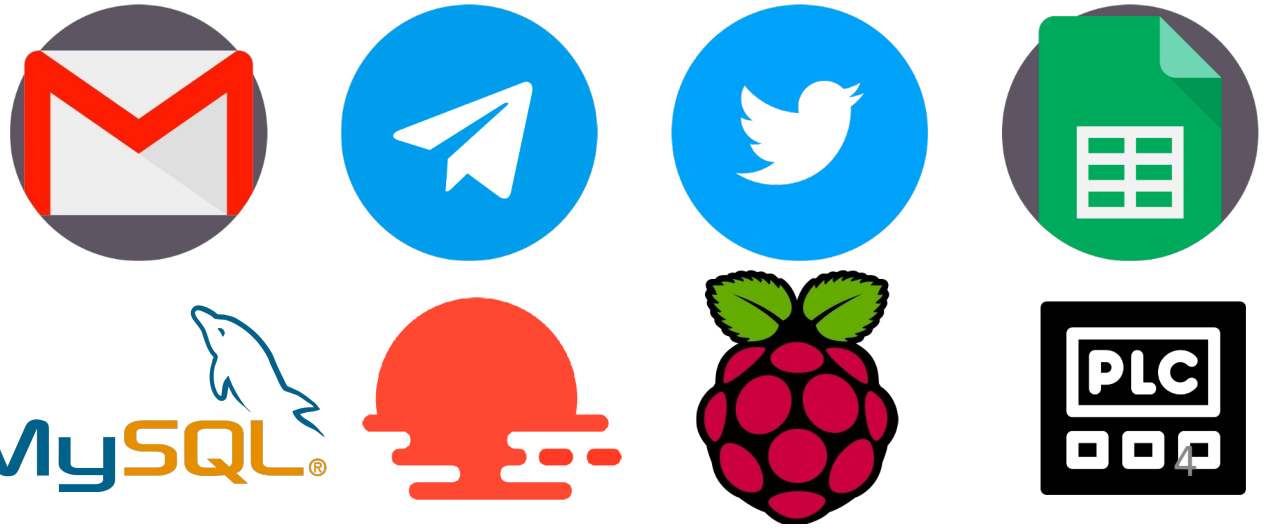

Node-RED
Deploy
filter nodes
IoT Sensors
Alarms detection
Moisture meters
Storage tanks
Lora network
common
inject
debug
complete
catch
status
link in
link call
link out
comment
function
function
switch
change
range
template
delay
trigger
exec
Display temperature in dashboard
MQTT temperature
connected
Prepare data for storage
Store data in InfluxDB
If temperature is abnormal
Send SMS
Send email
help
Search help
file
file in
function
html
http in
http proxy
http request
http response
inject
Injects a message into a flow either manually
at regular intervals. The message payload ca
be a variety of types, including strings,
JavaScript objects or the current time.
Outputs
payload
The configured payload of the message.
topic
An optional property that can be configure
in the node.
Details
The Inject node can initiate a flow with a
specific payload value. The default payload i
timestamp of the current time in millisecs sin
January 1st, 1970.
The node also supports injecting strings,
numbers, booleans, JavaScript objects, or
flow/global context values.
3
By default, the node is triggered manually by

Introduction to Node-RED

- Visual programming tool that can be accessed right from the web browser.
- Functional programming block called **nodes** are ready to use with less code or without having to program anything.
- Allow us to visually create automation system, system bot or even web scrapping.
- Able to interact with online services, such as Gmail, Telegram, Twitter, Google Sheet, OpenWeather, or even with the connected machine.



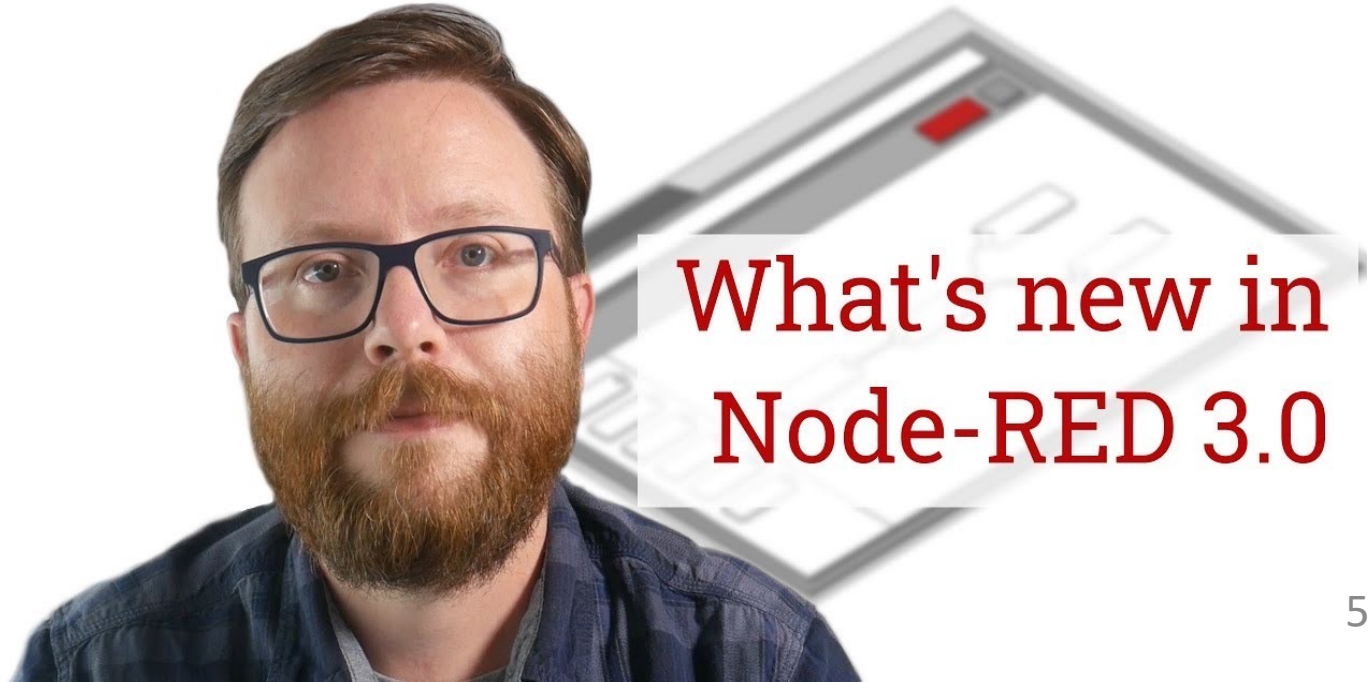
Programming & Integrations



Introduction to Node-RED

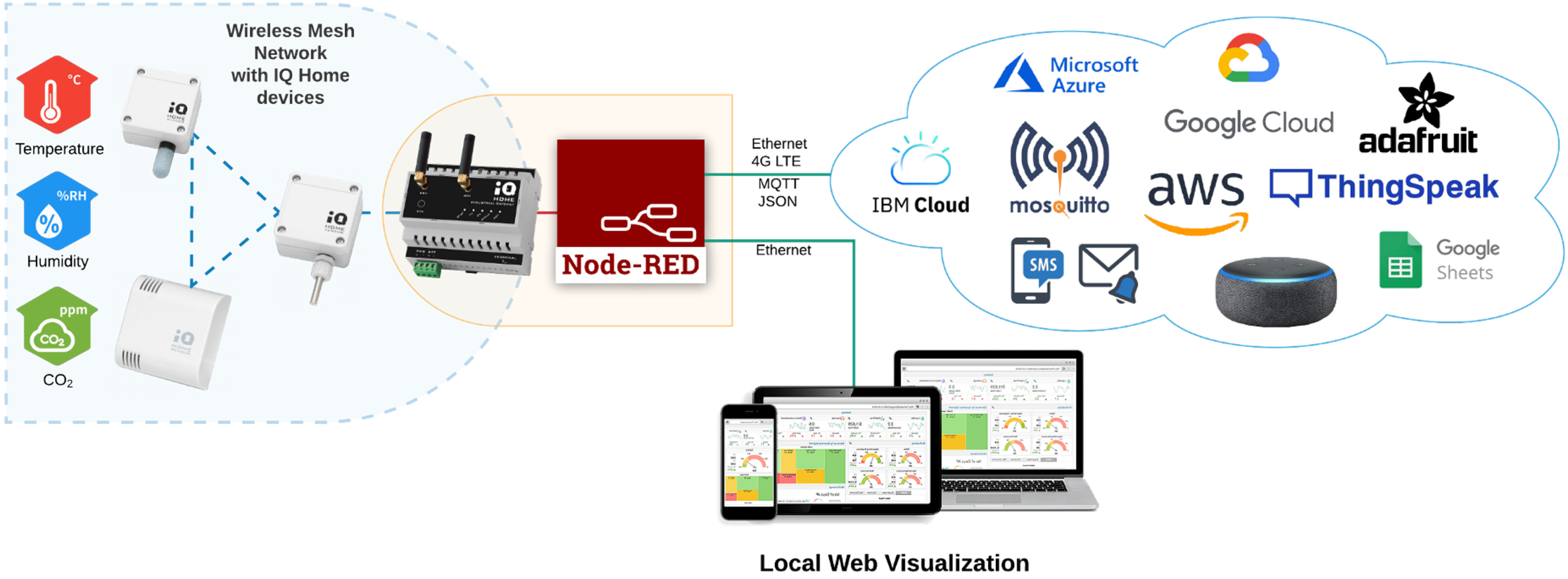
Who created Node-RED?

- Originally developed in 2013 by two IBM employees, Nick O'Leary and Dave Conway-Jones.
- They open sourced the source code which made available on GitHub https://github.com/node-red/node-red



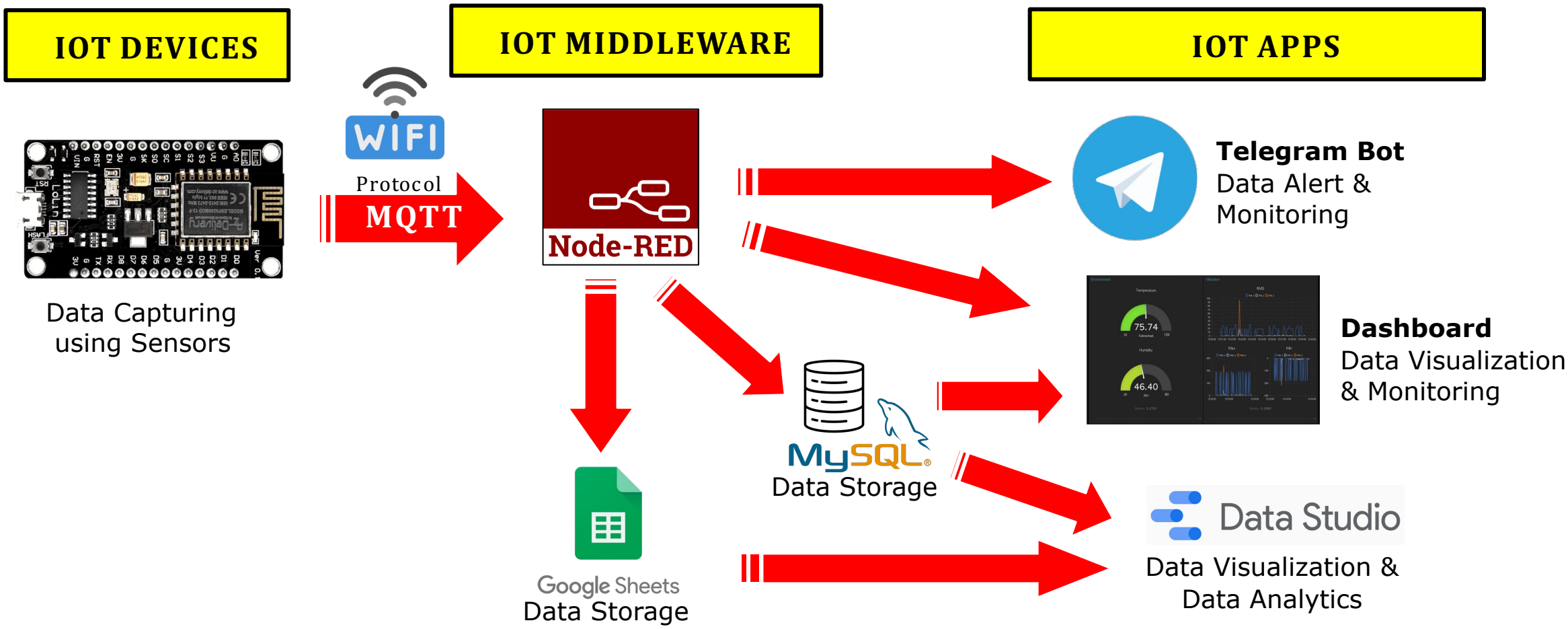
Introduction to Node-RED

Node-RED is the IoT middleware between the physical world, where data is collected by the smart devices and provision the data to the cloud computing or Internet services.



Introduction to Node-RED

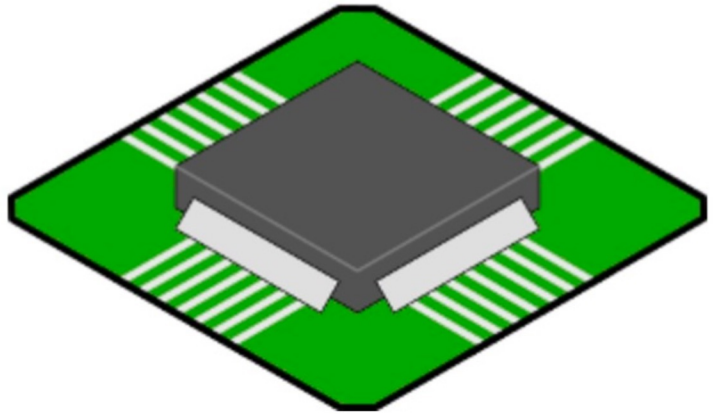
Basic IoT architecture using Node-RED as the middleware.



Introduction to Node-RED

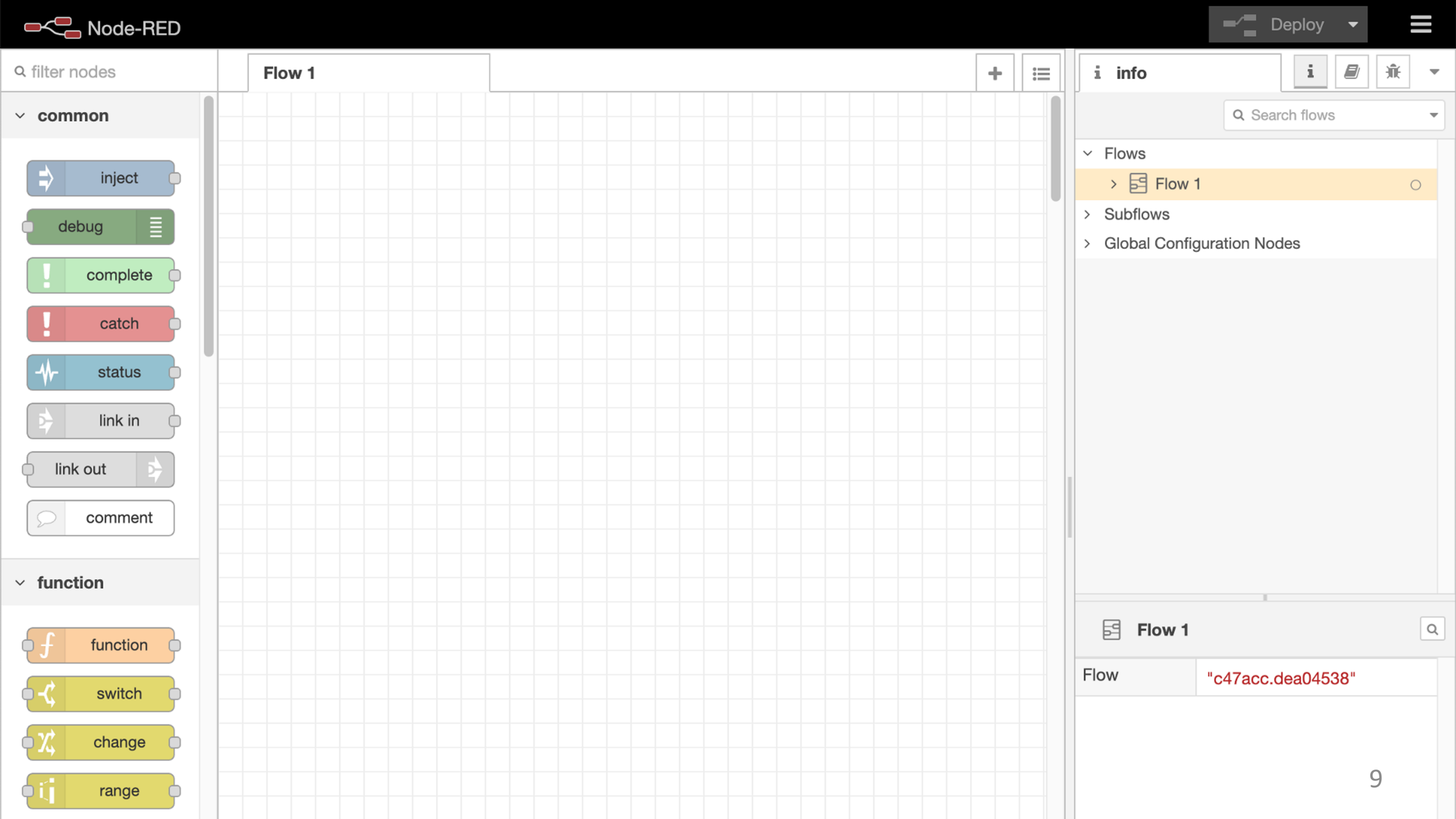
- Node-RED is a flexible middleware that can run on multi-cross platform such as:
  - Personal computer (PC running on Windows, Linux or Mac OS)
  - Local server or tiny computer (e.g. Raspberry Pi, Beagle Bone, Orange Pi)
  - Cloud computing services.

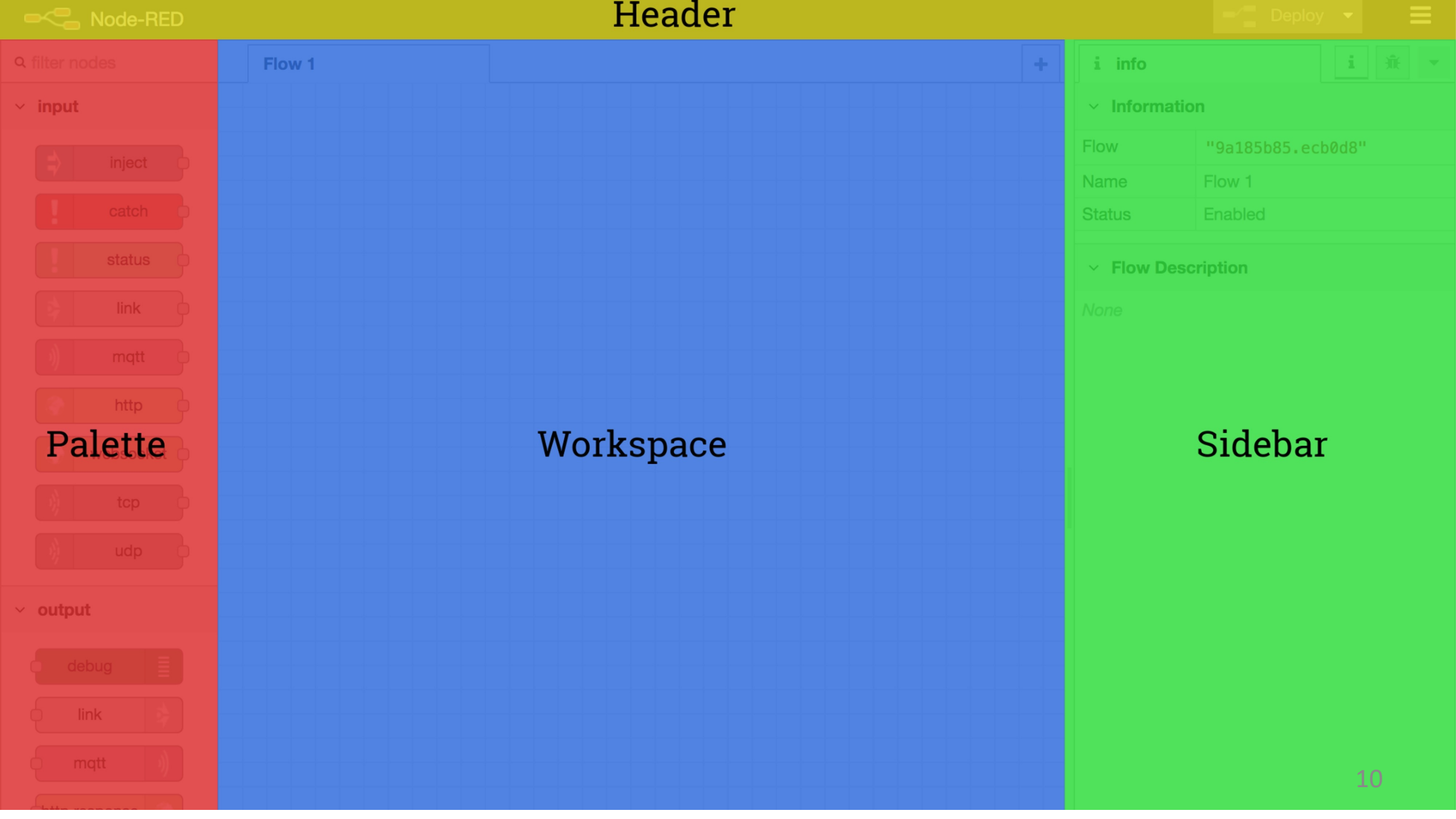
Run locally



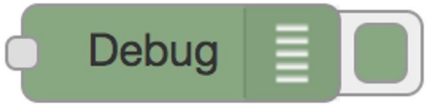
On a device

In the cloud

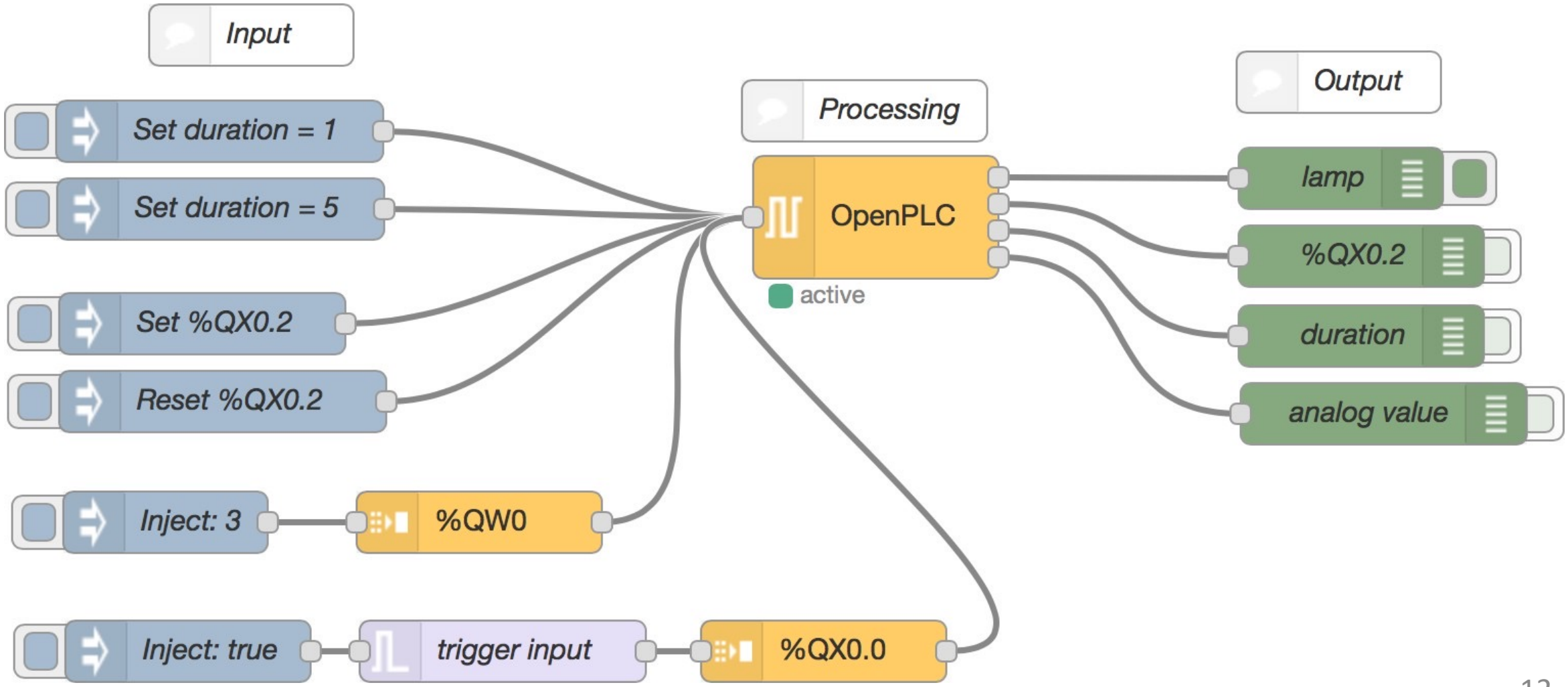

Node-RED
Deploy
filter nodes
Flow 1
info
common
inject
debug
complete
catch
status
link in
link out
comment
function
function
switch
change
range
Search flows
Flows
Flow 1
Subflows
Global Configuration Nodes
Flow 1
Flow
"c47acc.dea04538"


Node-RED
Header
Deploy
filter nodes
Flow 1
info
input
Information
inject
Flow
"9a185b85.ecb0d8"
catch
Name
Flow 1
status
Status
Enabled
link
Flow Description
mqtt
None
http
Palette
Workspace
Sidebar
tcp
udp
output
debug
link
mqtt

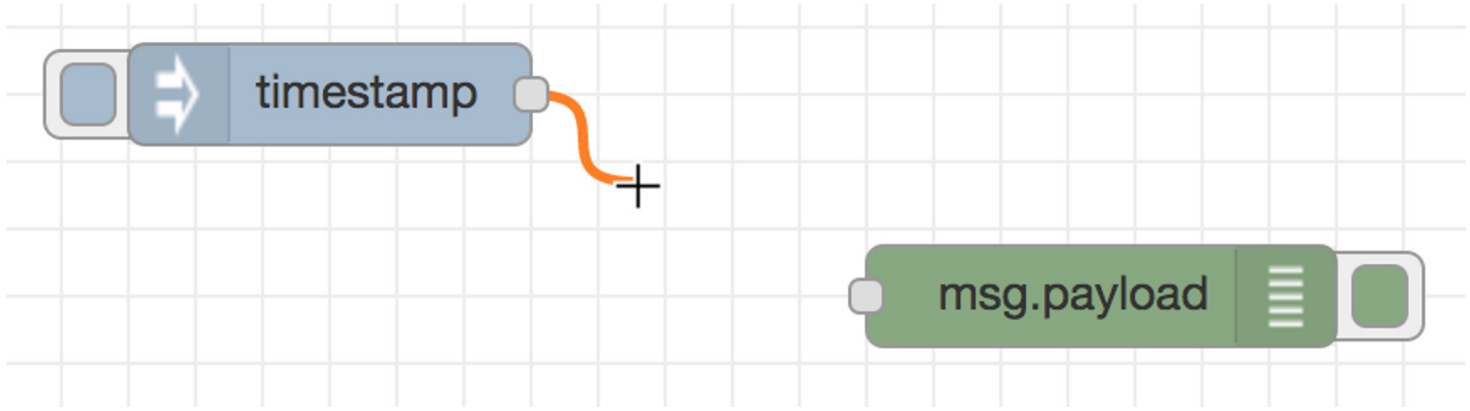
Common Node-RED Nodes

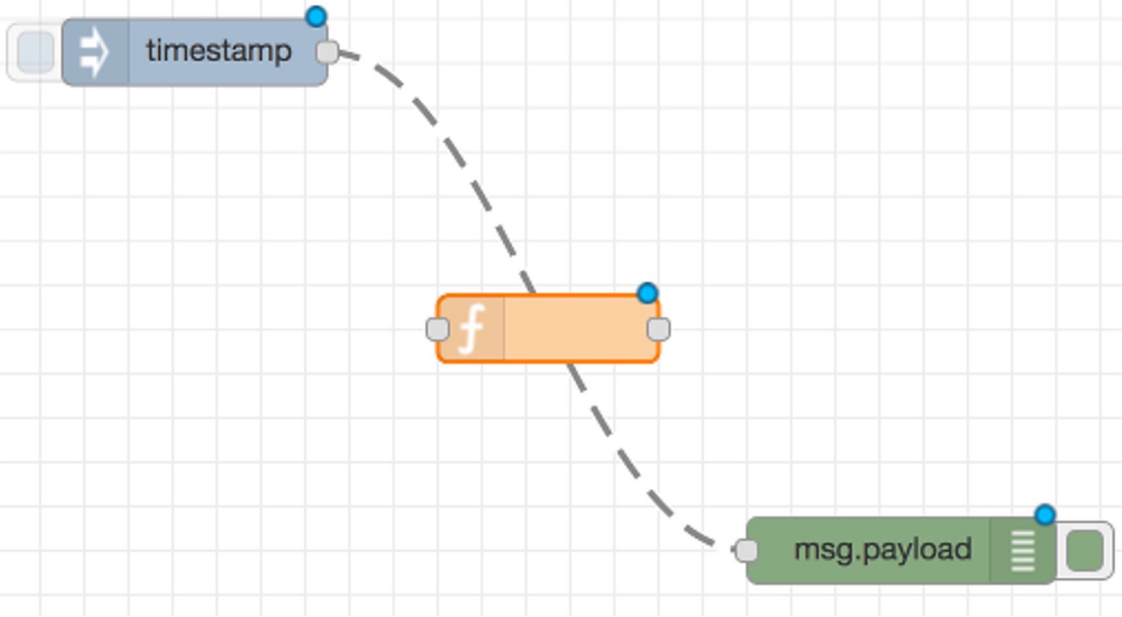
Node	Description
	The **Inject node** can be used to manually trigger a flow by clicking the node's button within the editor. It can also be used to automatically trigger flows at regular intervals.
	The **Debug node** is used to display messages which comes from the output of other node, which the messages can be seen by the debug tab, within the Node-RED sidebar. The button on the node can be used to enable or disable the output node. It is recommended to disable or remove any Debug nodes that are not being used.
	The **Function node** allows developers to write JavaScript code to be run on the node, against the messages that are passed through it.

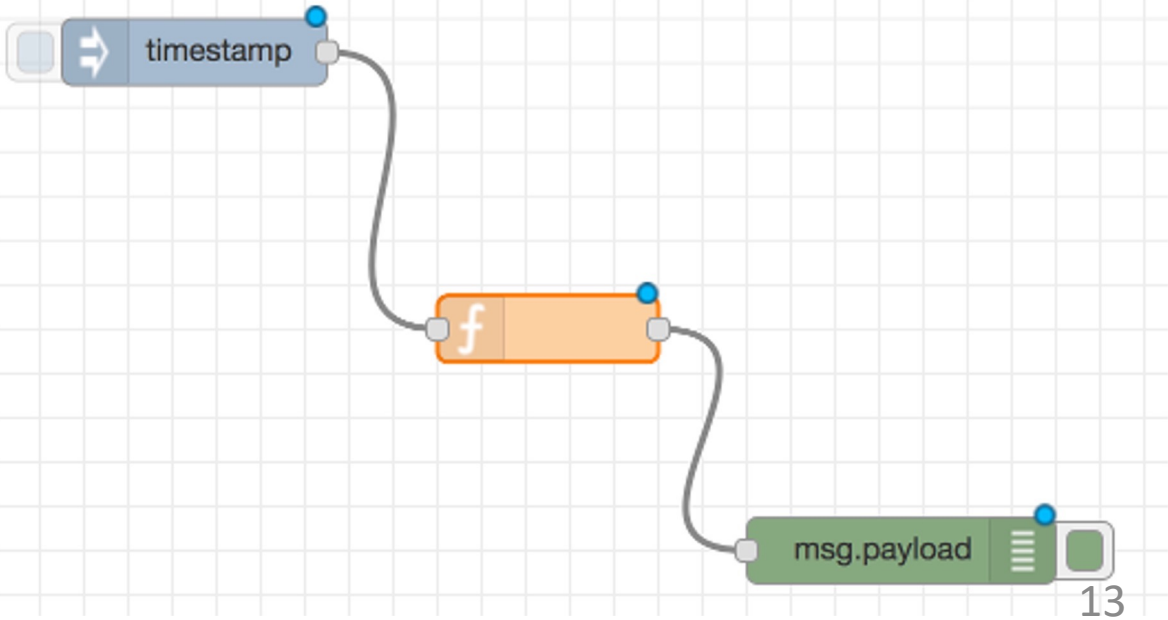
Node-RED Flow Programming Logic



Node-RED Flow Programming Logic

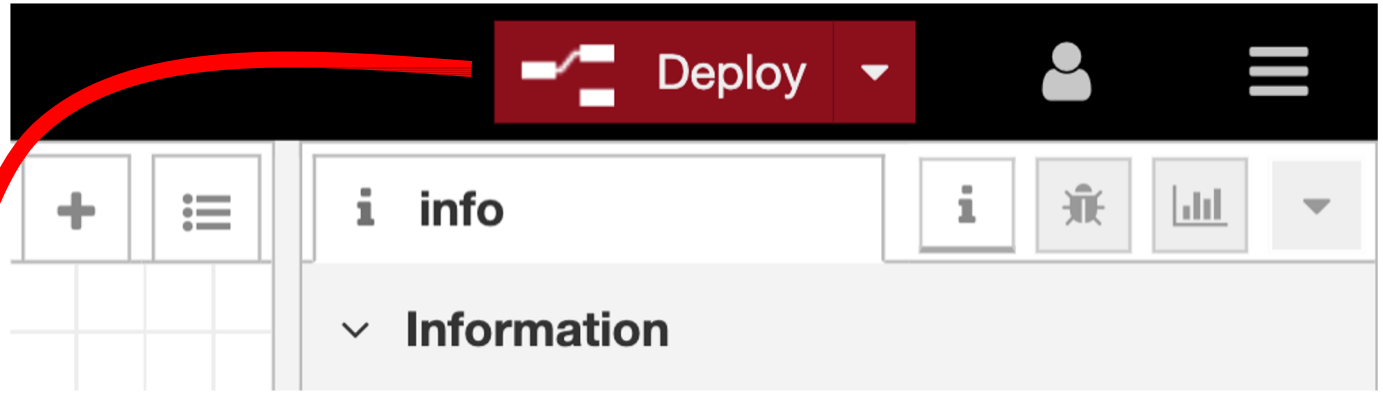




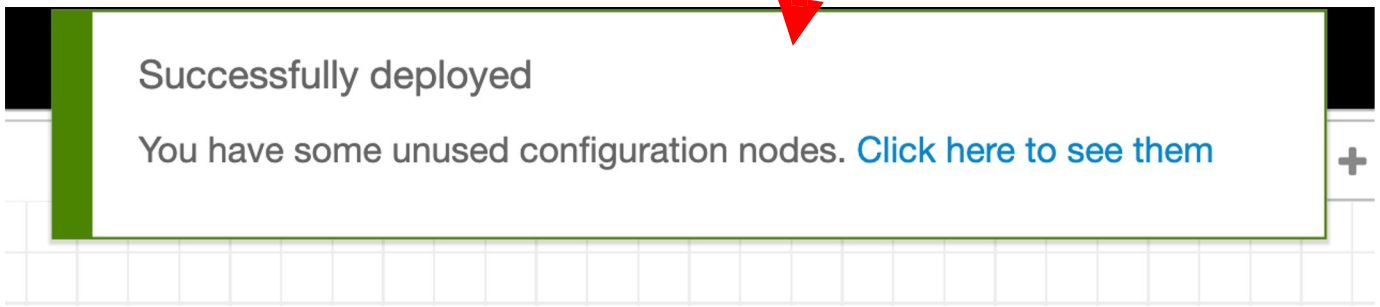


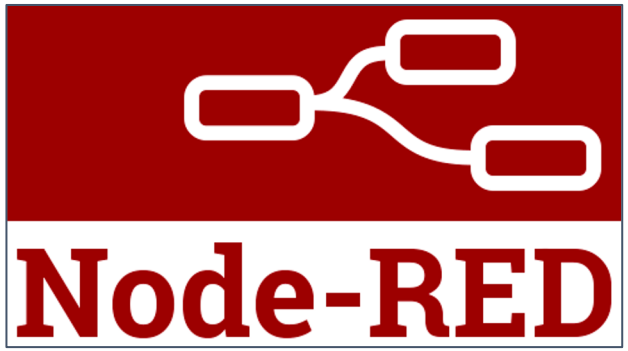
Node-RED Flow Programming Logic

- Very important button to execute the Node-RED flow process function.
- Click for every flow or nodes changes.



Deployed notification



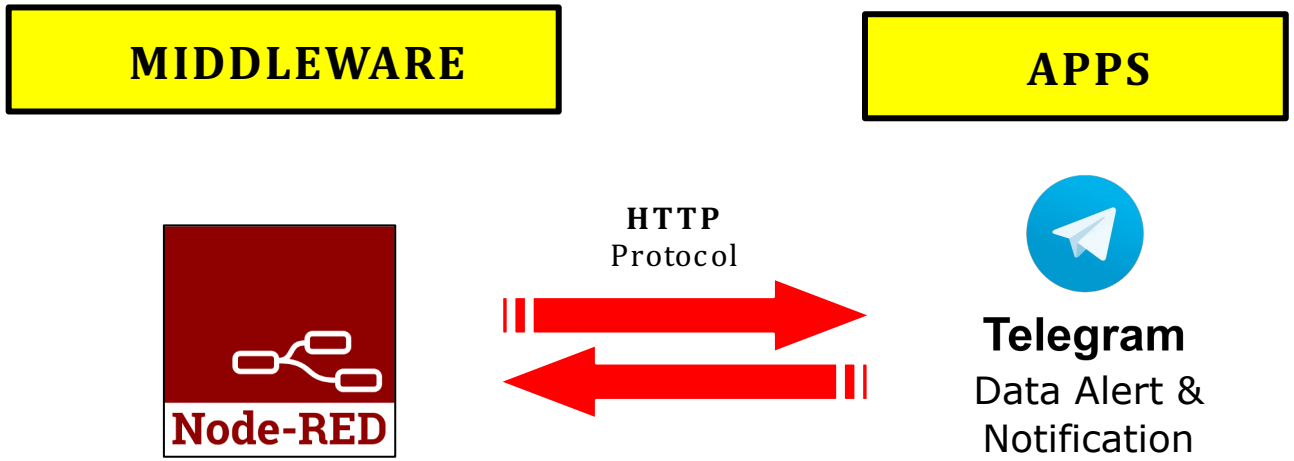


Node-RED & Telegram

Integrating visual based programming with the most popular messaging platform in the world

Introduction to Node-RED

The architecture:

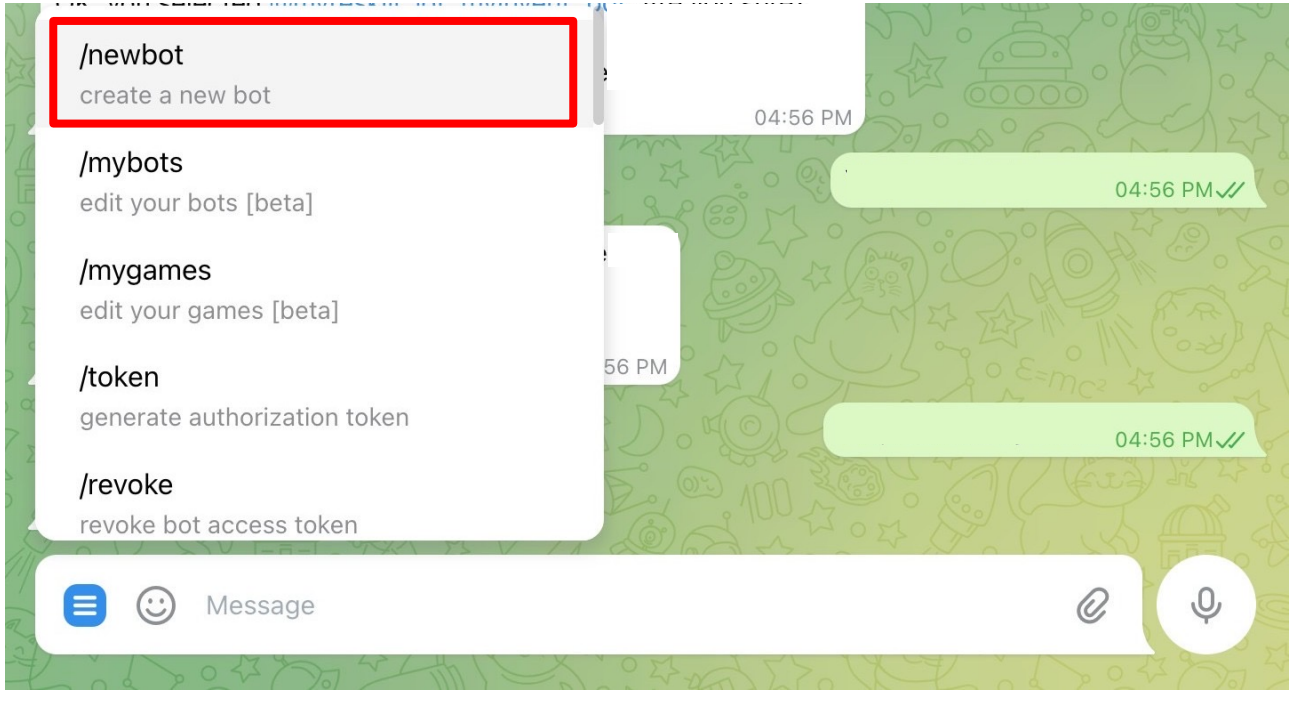


Node-RED with Telegram Bot

Let's Create a Telegram Bot (4 Steps)

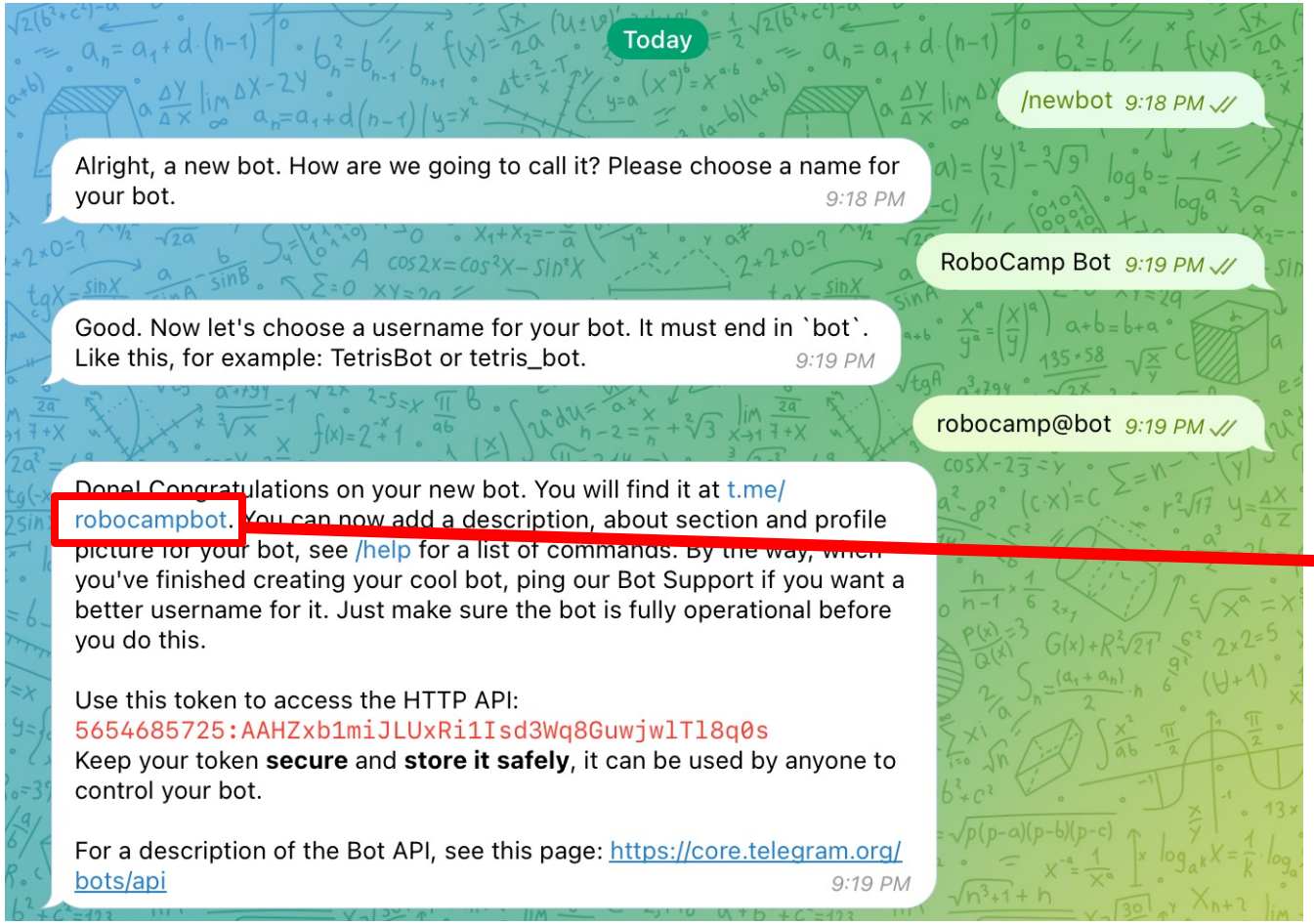
1. In Telegram, search for botfather or go to https://t.me/BotFather

2. Click on the chat menu and click **/newbot** to create new bot.

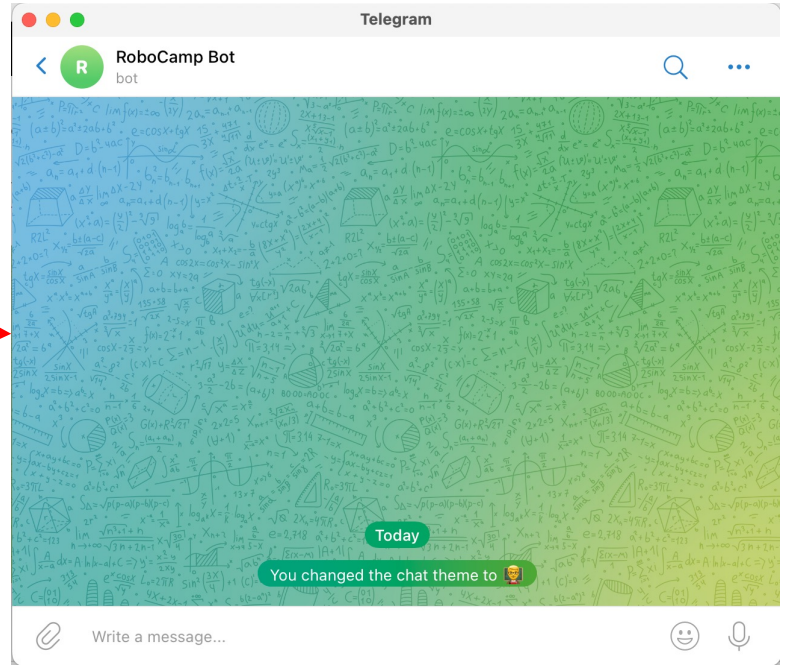


Node-RED with Telegram Bot

3. Follow the instruction to create bot's name and bot's username as below conversation.



4. Click on the created bot's link and the bot is now accessible. Let's connect with Node-RED.

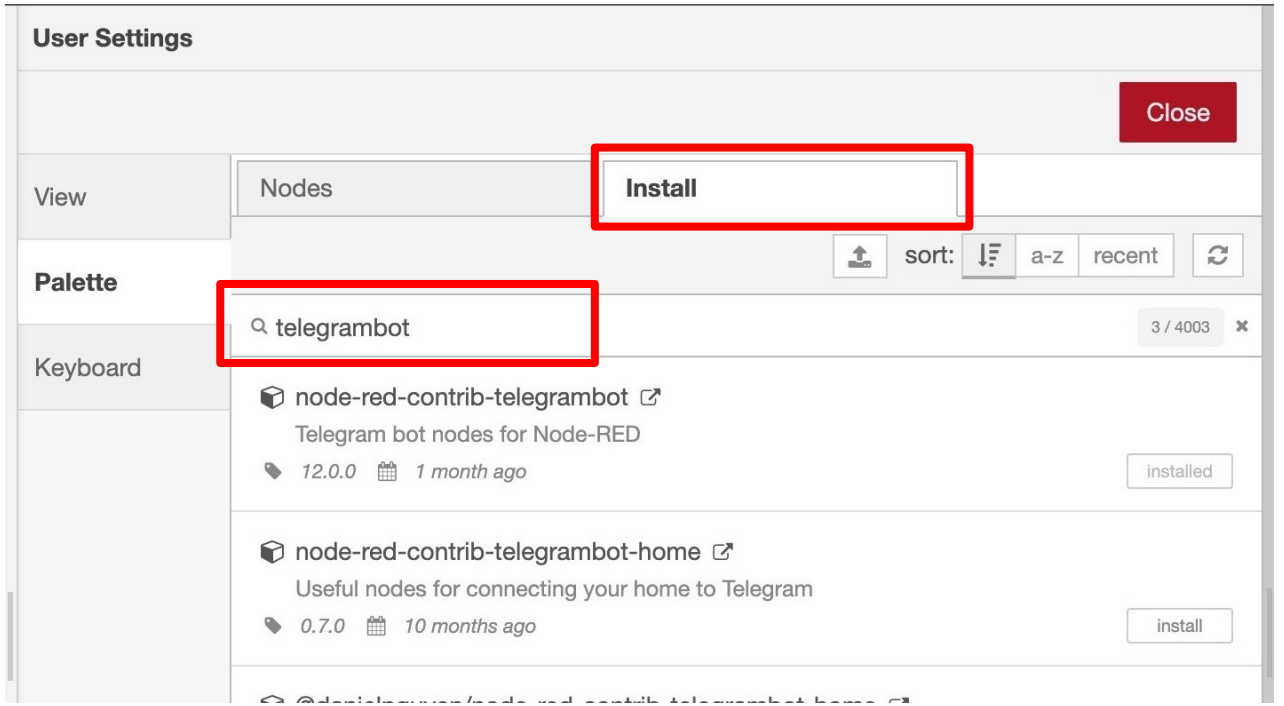


Node-RED with Telegram Bot

- Click on the Node-RED's menu, and click **Manage palette**

Node-RED with Telegram Bot

- Click on the **Install** tab and on the **search modules** fields type **telegrambot** to filter the list of the available modules.
- Click the **install** button to install the **node-red-contrib-telegrambot** module and wait until the installation process is successful.



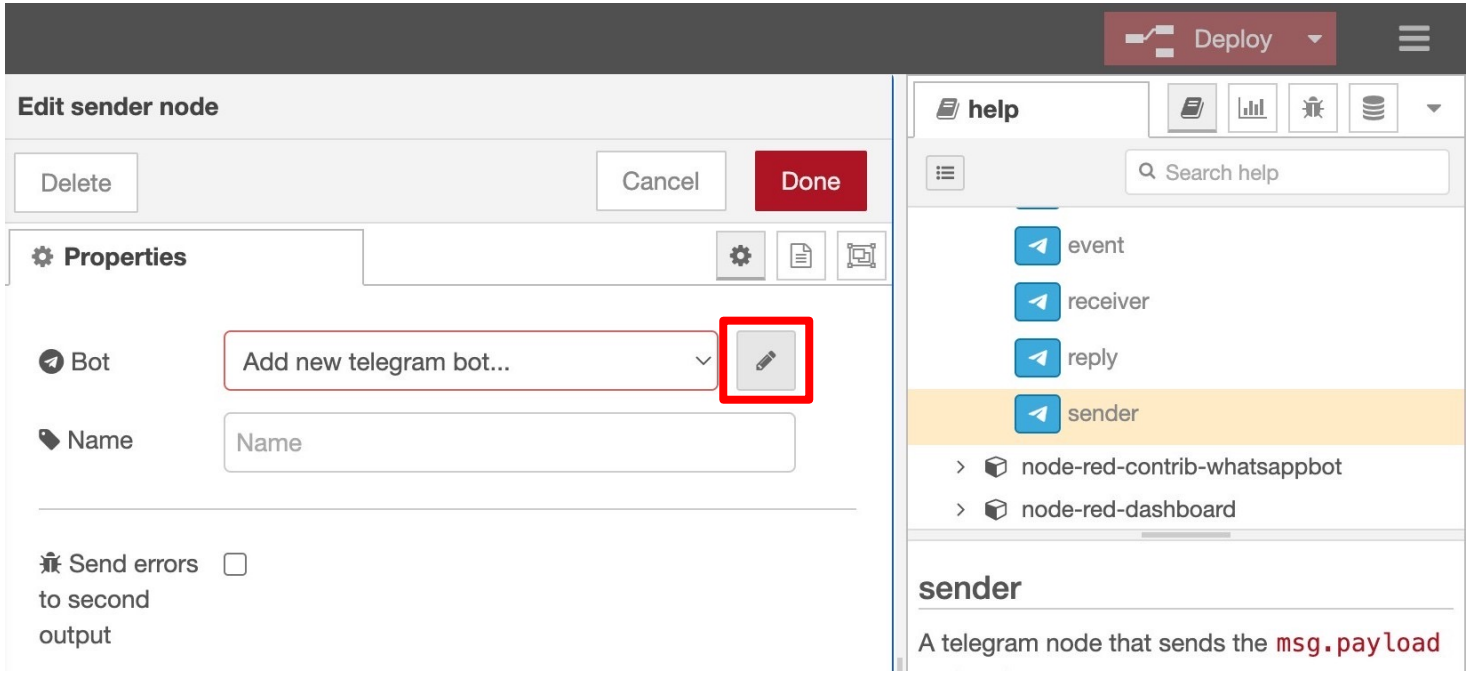
Node-RED with Telegram Bot

- Once the installation is successful, the telegram nodes is available on the palette under category: **telegram**. The nodes includes *receiver*, *command*, *event*, *sender* and *reply*.

Node-RED with Telegram Bot

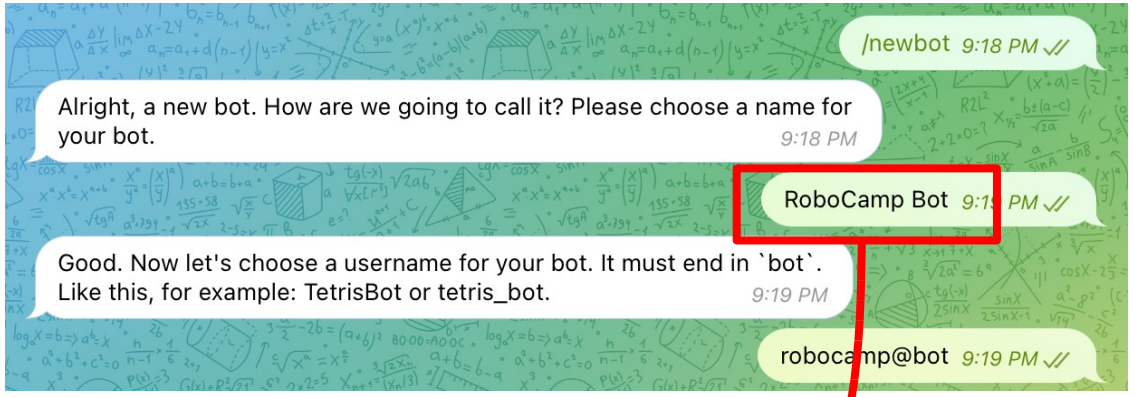
Let's Get Bot's Chat ID (12 Steps)

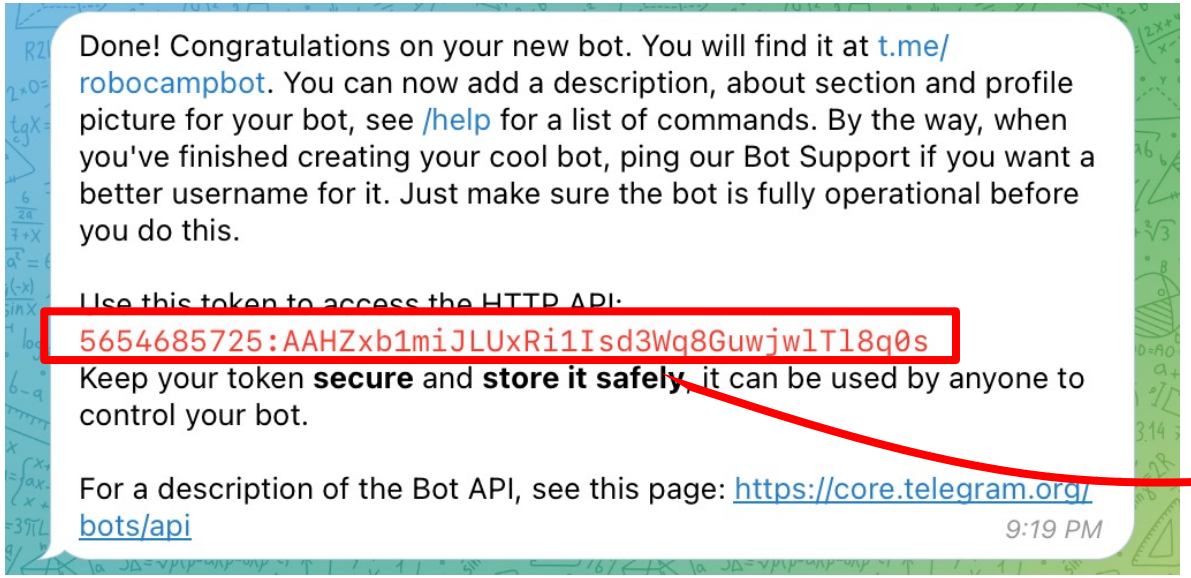
1. Insert **receiver** telegram node into the workspace.
2. Double click the node to edit the Properties.
3. Click **pencil icon** to **Add new telegram bot...**

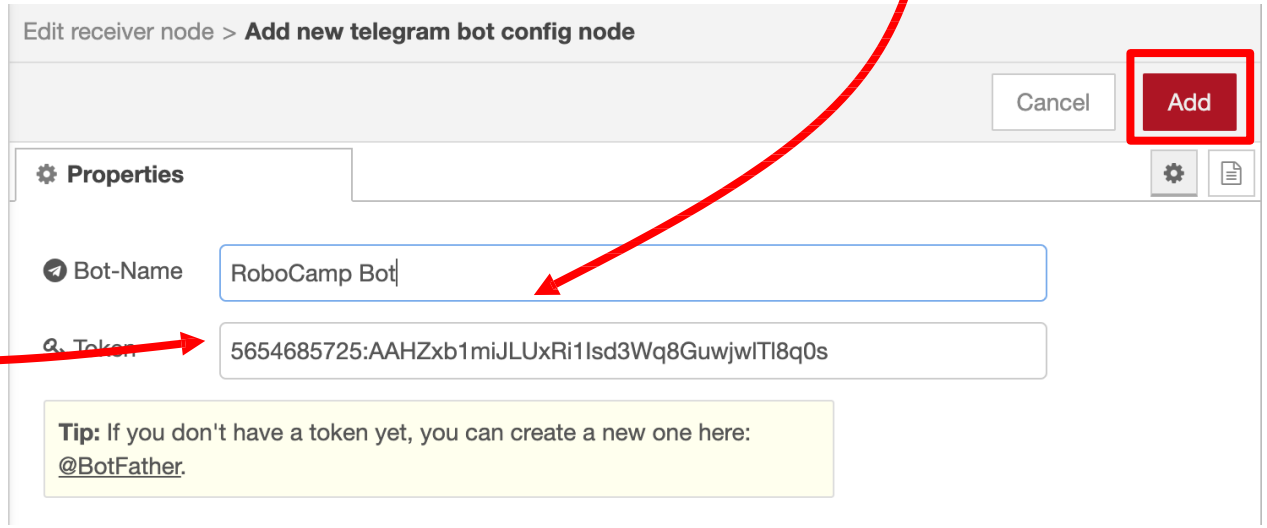


Node-RED with Telegram Bot

4. Insert **Bot-Name**
5. Insert **Token**
6. Click the **Add** button.

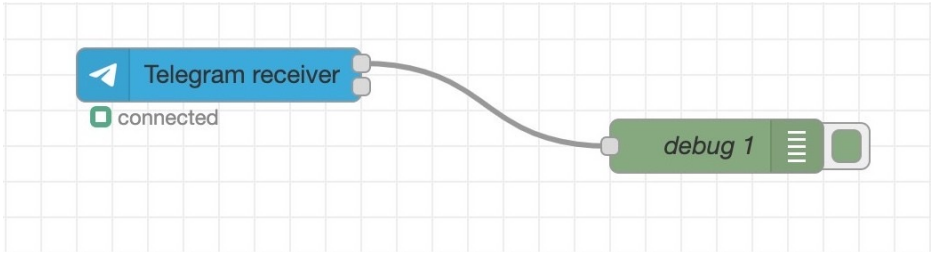


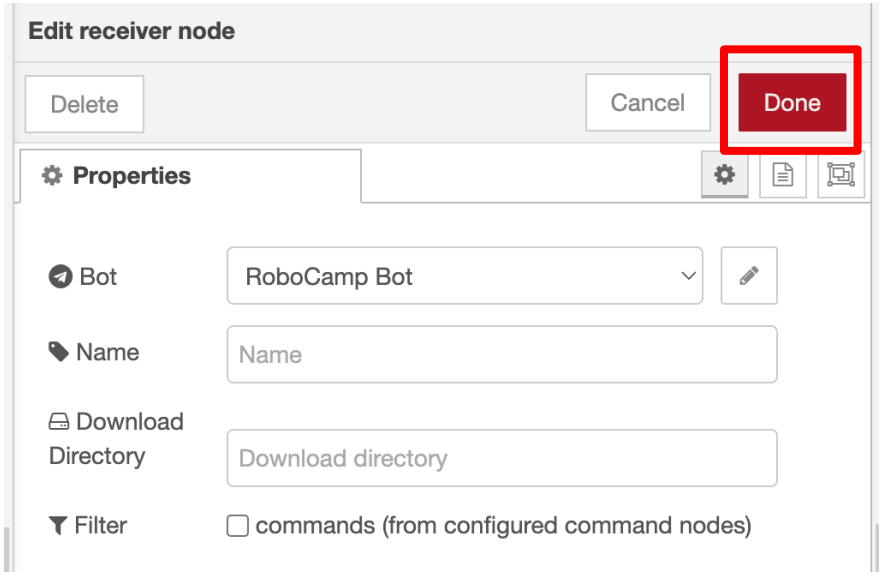




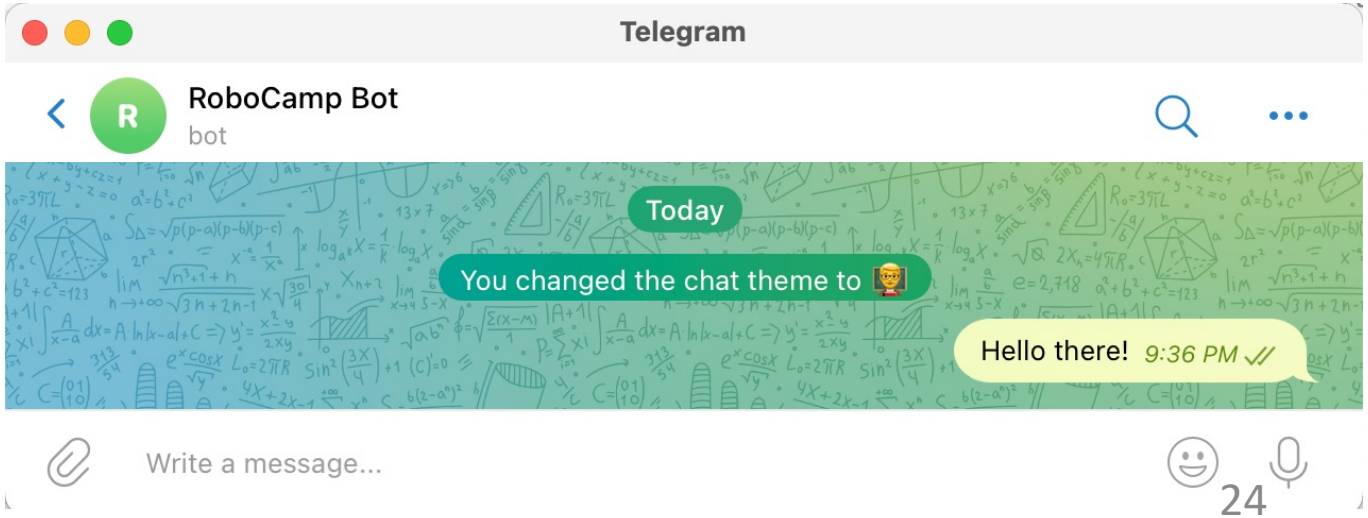
Node-RED with Telegram Bot

7. The **Bot** information is ready.
8. Click the **Done** button.
9. Click the Node-RED's **Deploy** button.



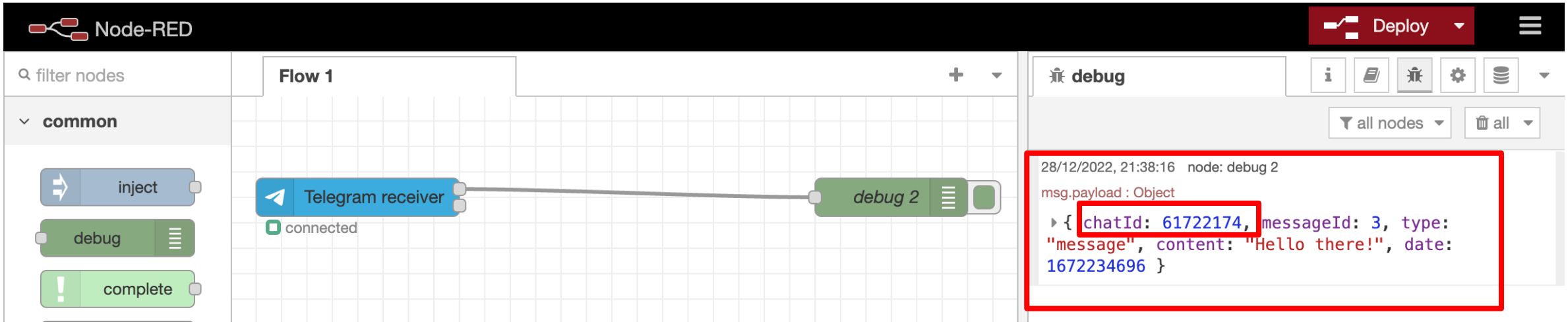


10. Go to Bot chat box and send some message.



Node-RED with Telegram Bot

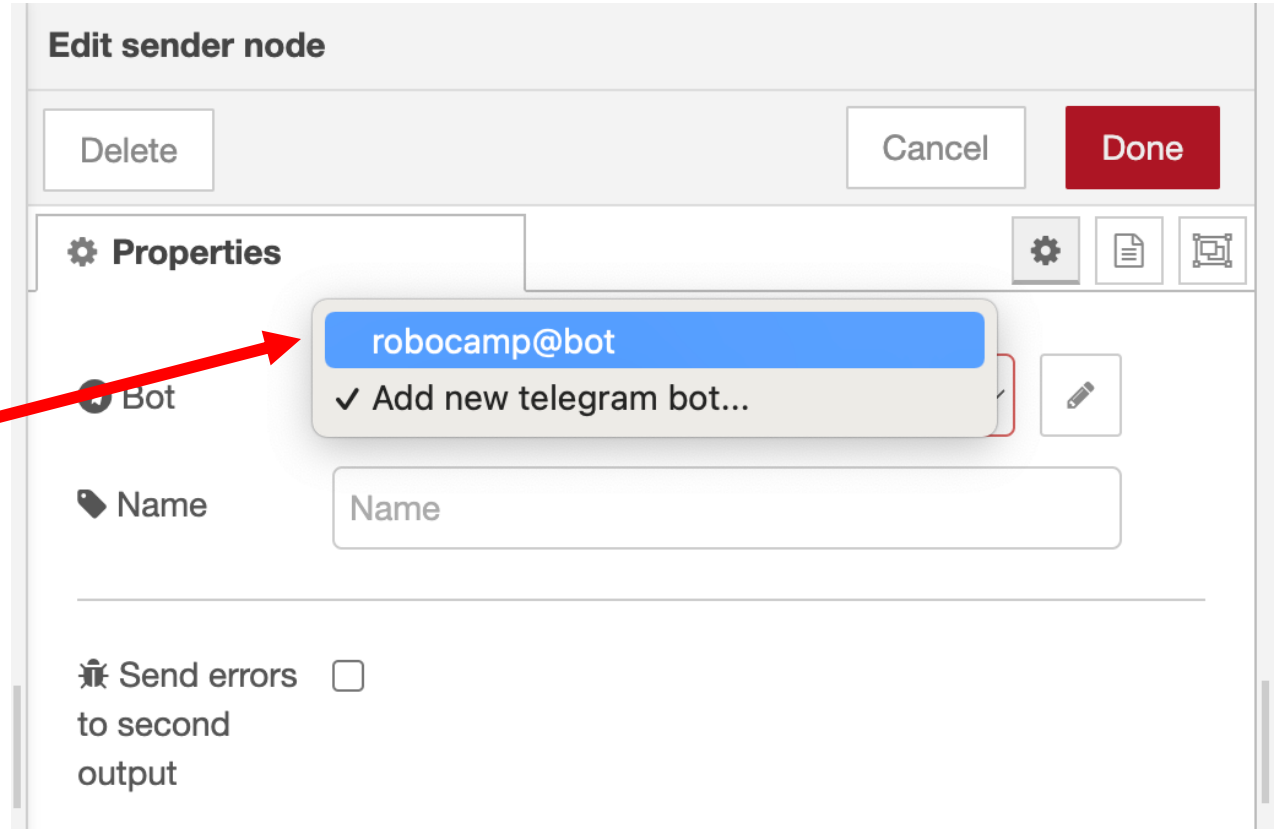
11. Observe the Debug tab output on the Node-RED's sidebar.
12. The Bot's Chat ID is the value of **chatId** key as available on the **msg.payload** object.



Node-RED with Telegram Bot

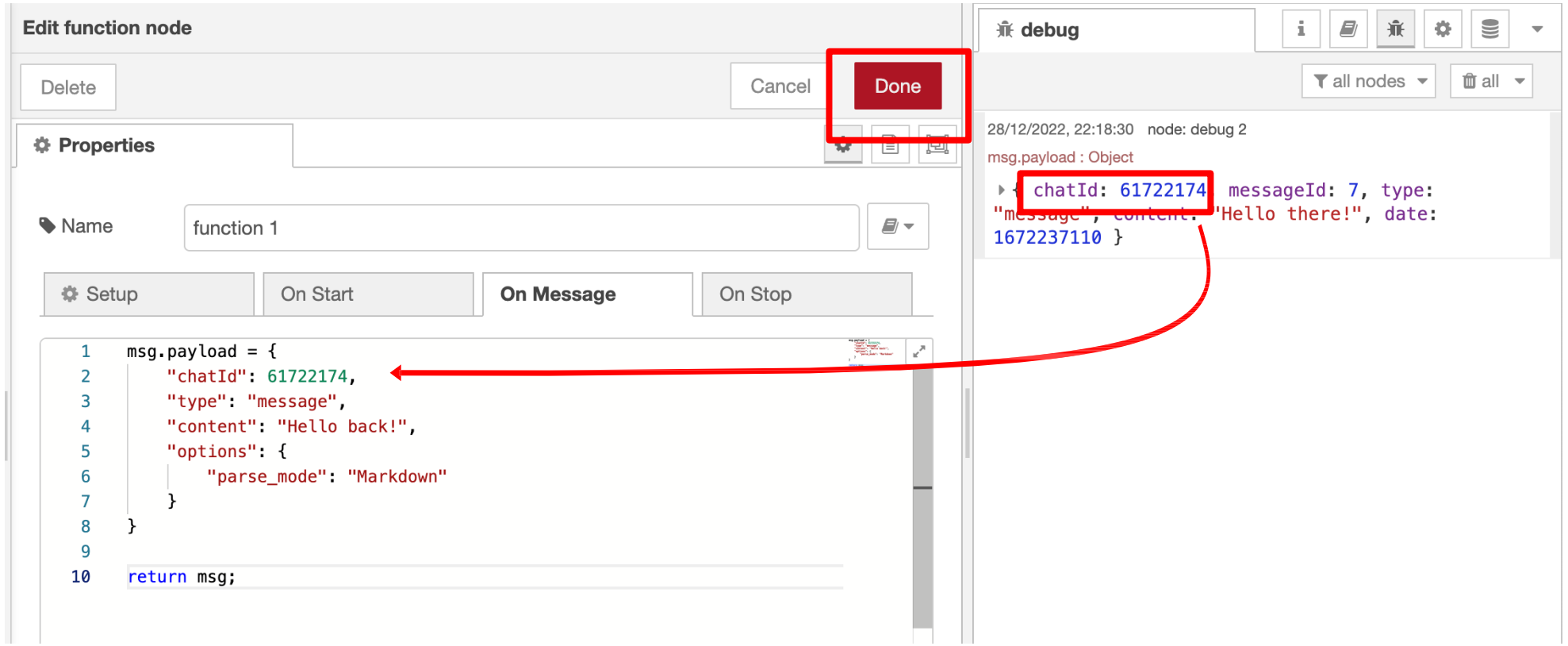
Send Automated Message to the Telegram Bot (7 Steps)

1. Insert **Inject** node, **Function** node and Telegram **sender** node into the workspace.
2. Edit the Telegram **sender** node Bot's properties, select your existing Bot that had been setup before.



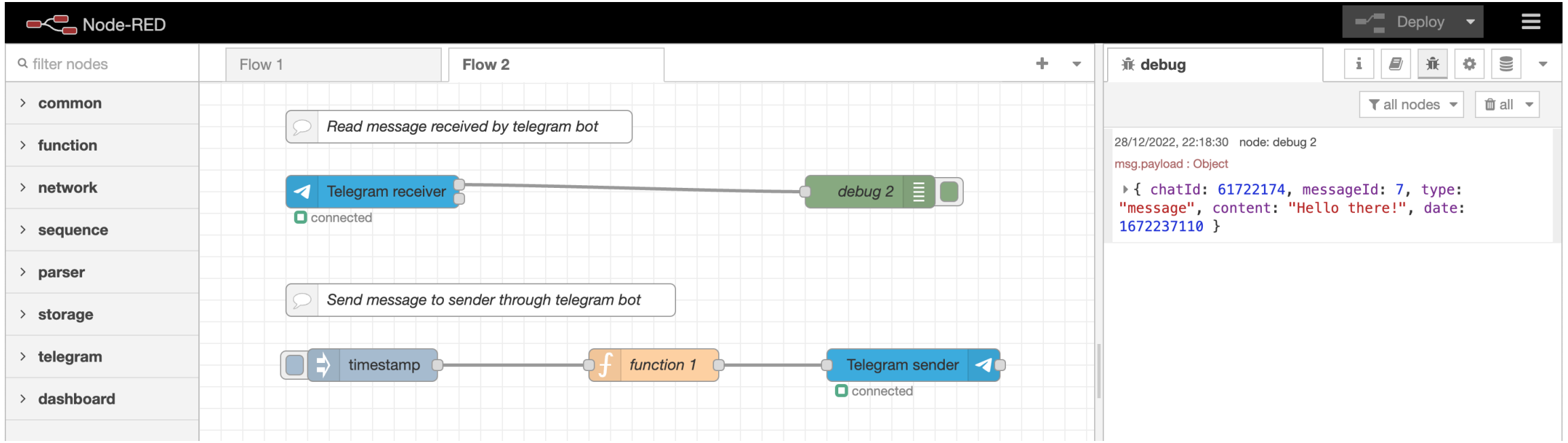
Node-RED with Telegram Bot

3. Open the **Function** node to add Telegram's chat information as in the image below.



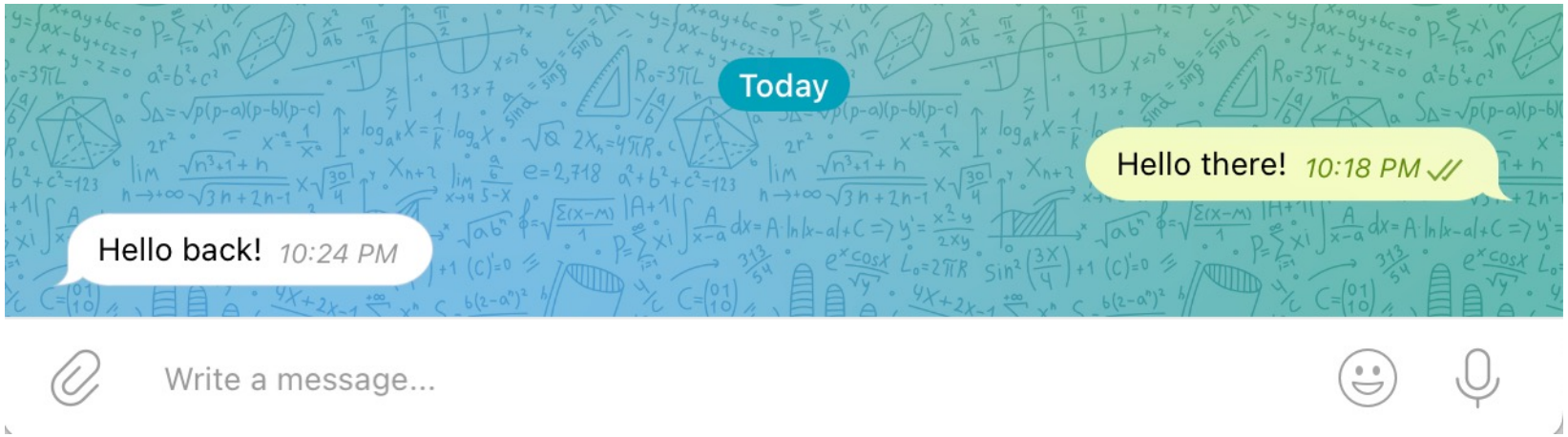
Node-RED with Telegram Bot

4. Connect the nodes.
5. Click the Node-RED's **Deploy** button.
6. Click the **Inject** node button to send the message of **Assalamualaikum** to the Bot.

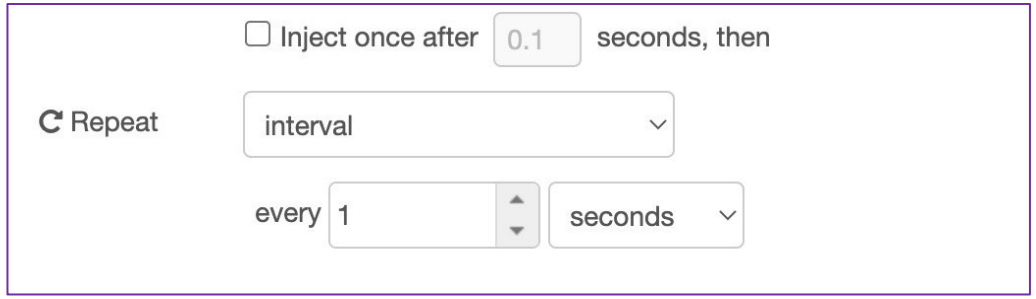


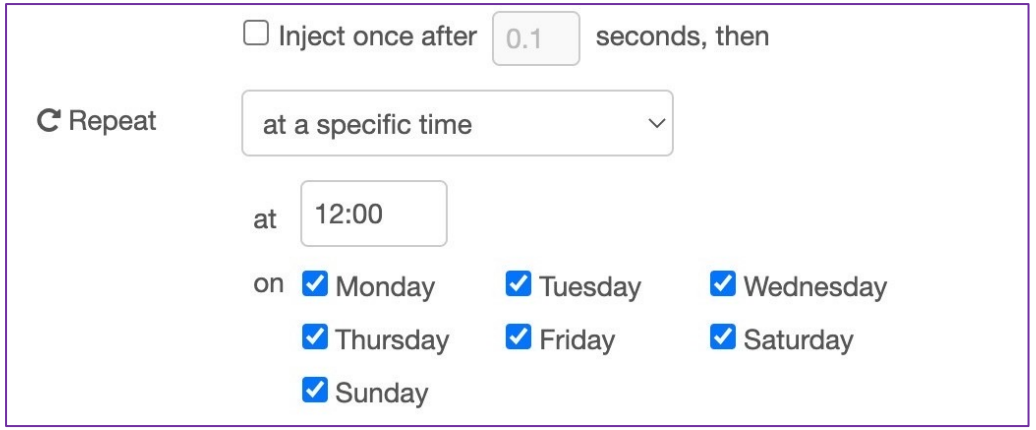
Node-RED with Telegram Bot

7. Check the Telegram, the Bot has received the message from Node-RED.



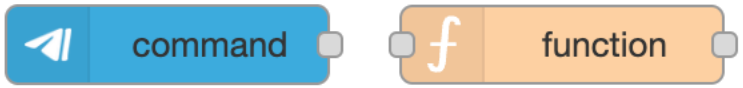
8. Change the Inject node **Repeat** properties to automatically send the message by interval time or specific time.



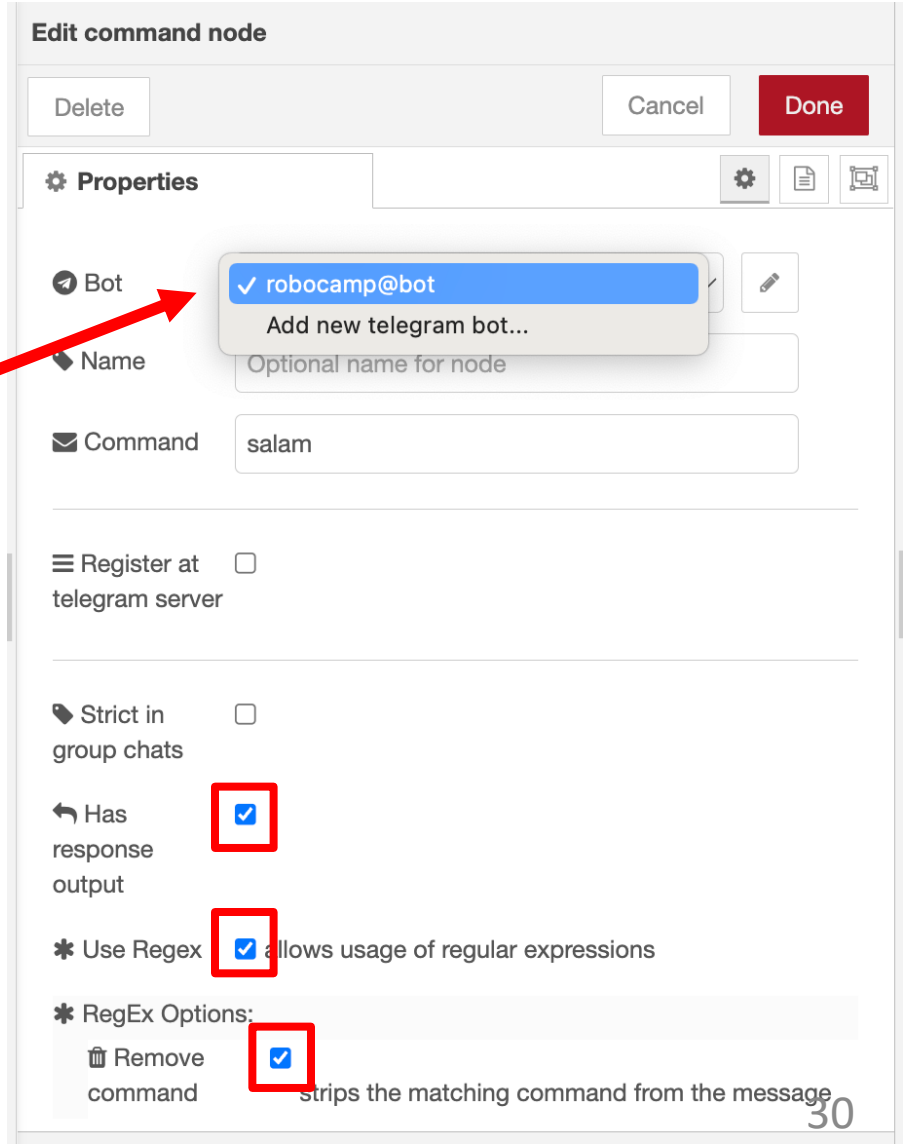


Node-RED with Telegram Bot

Automated Reply Message to the Telegram Bot (8 Steps)

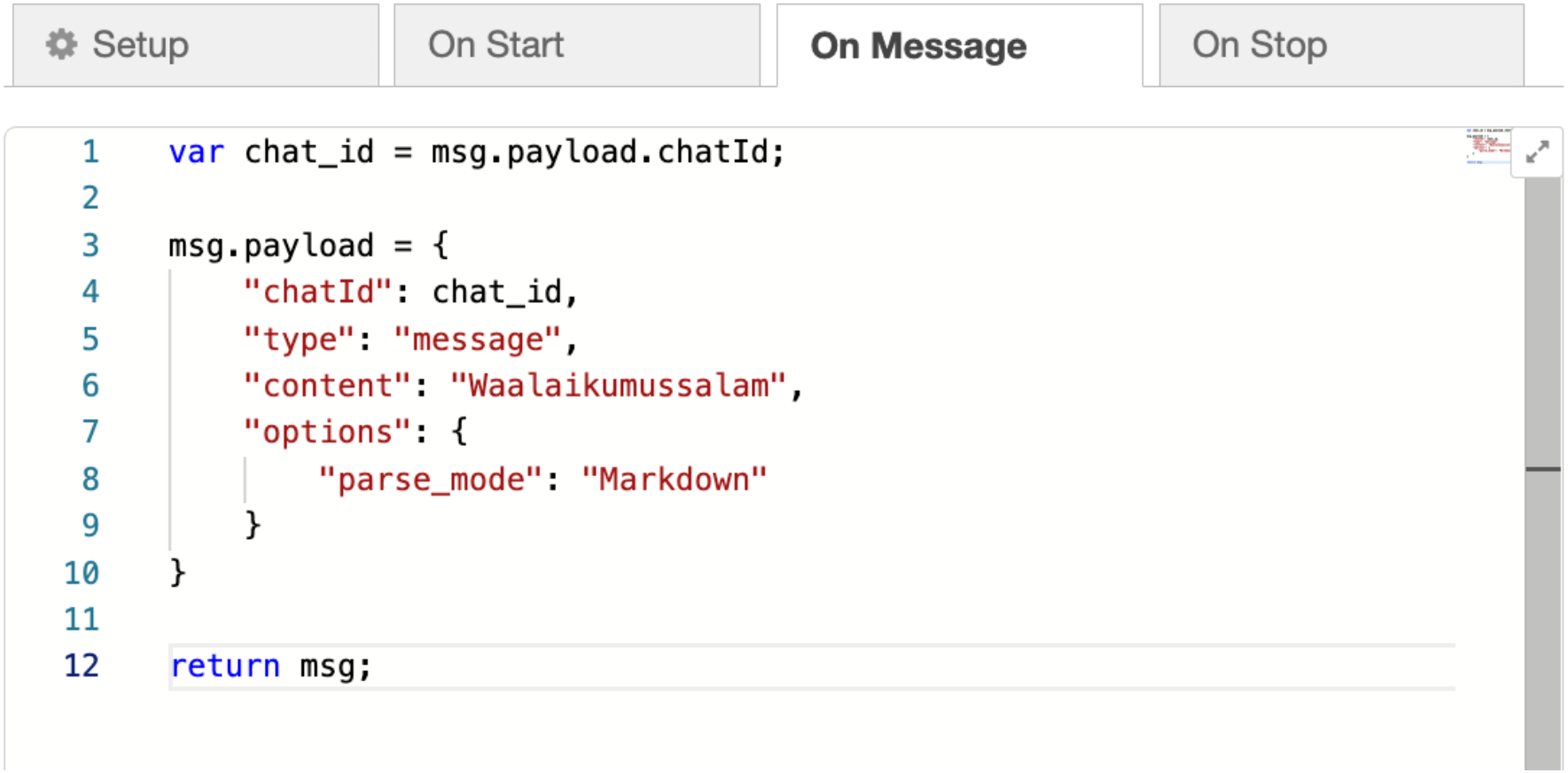


1. Insert telegram **command** node and **Function** node into the workspace.
2. Edit the Telegram **sender** node Bot's properties, select your existing Bot that had been setup before.
3. Command is **salam**.
4. Enable these options:
   - Has response output
   - Allows usage of regular expressions
   - Strips the matching command from the message



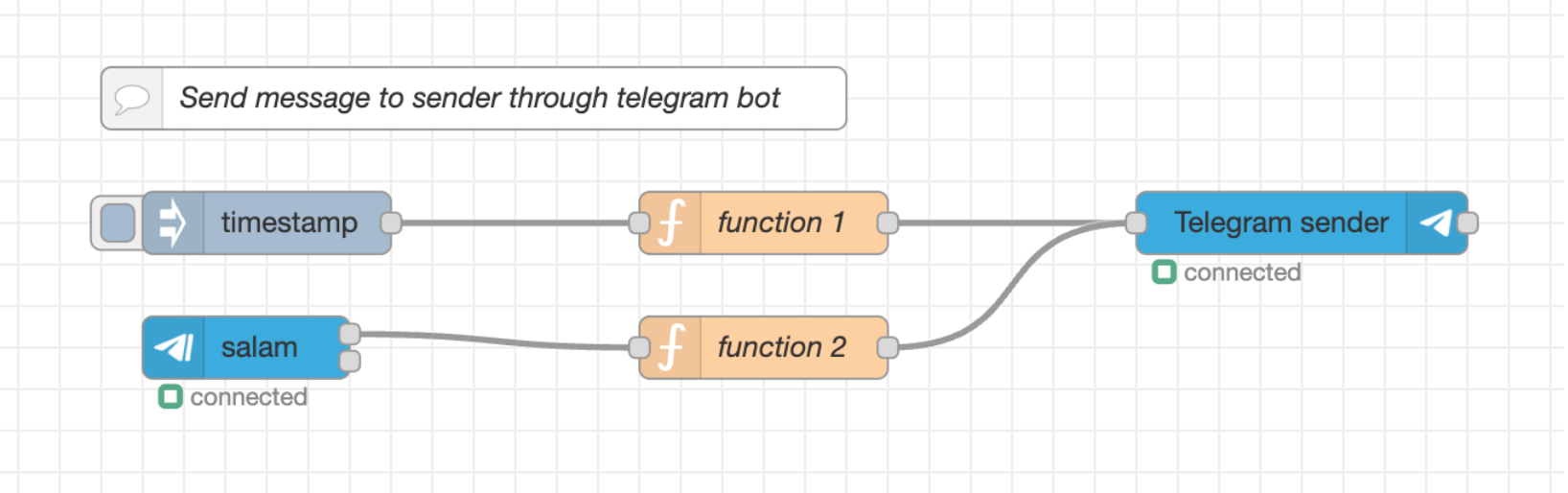
Node-RED with Telegram Bot

5. Open the **Function** node to add the code below in the **On Message** tab. Click Done.



Node-RED with Telegram Bot

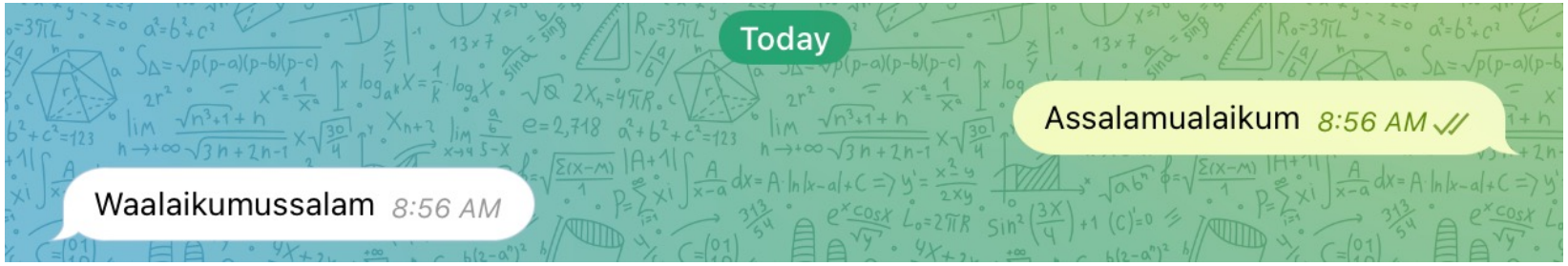
6. Connect the **command (salam)** node to **function** node. Connect **function** node to the previous **sender** node.



7. Click the Node-RED's **Deploy** button.

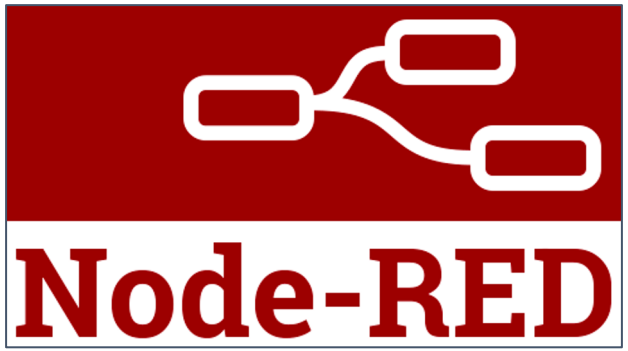
Node-RED with Telegram Bot

8. Open the Telegram, send "*Assalamualaikum*" message, the Bot should reply "*Waalaikumussalam*" automatically.



Note:

The command node will look for any message containing the command word "*salam*". If it find any, function node will send "*Waalaikumussalam*" text to the sender **ChatId**. Sender ChatId is obtained from **msg.payload.chatId** received by the command node.



Node-RED Dashboard

Set of nodes in Node-RED to quickly create a live data dashboard.

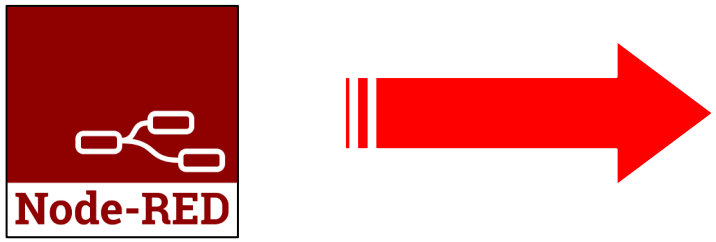
It provides nodes to quickly create a UI (user interface) with buttons, text input, sliders, charts, gauges, etc.

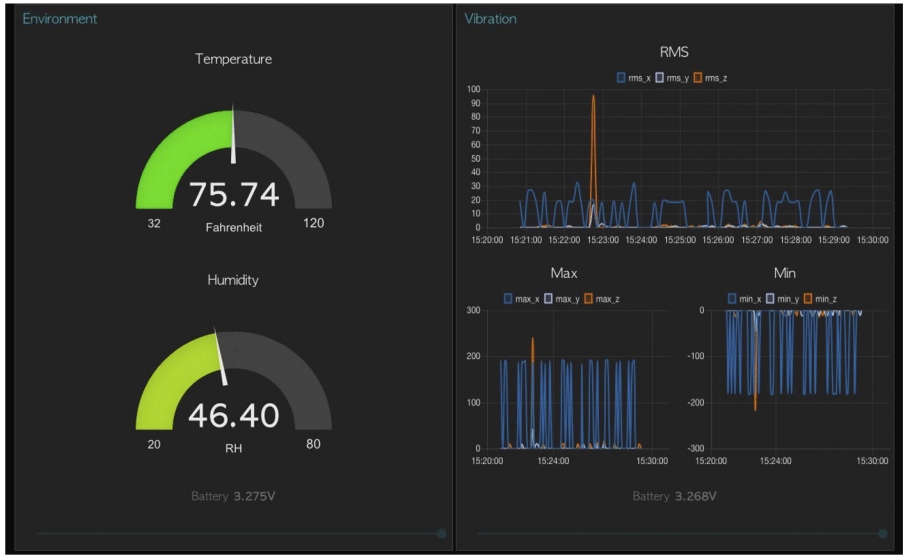
Introduction to Node-RED

The architecture:

MIDDLEWARE

DATA VISUALISATION





Data Visualization (Dashboard)
Data Monitoring

Environment

Temperature

Humidity

Battery 3.275V

Vibration

RMS

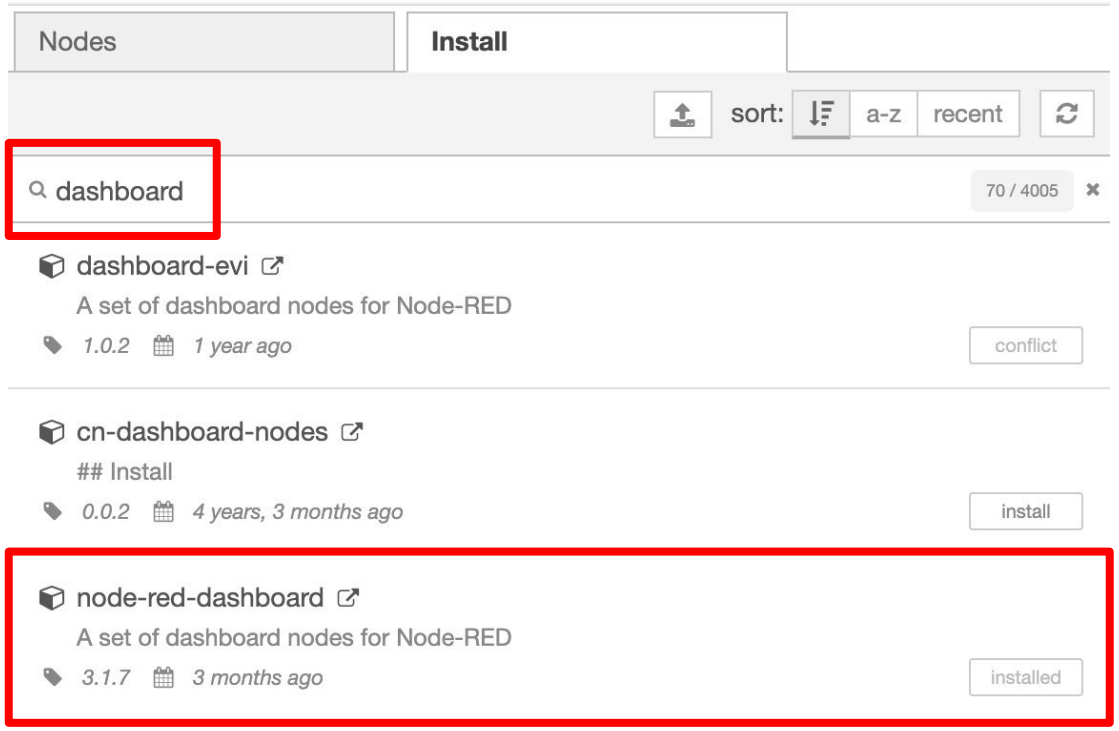
Max

Min

Battery 3.268V

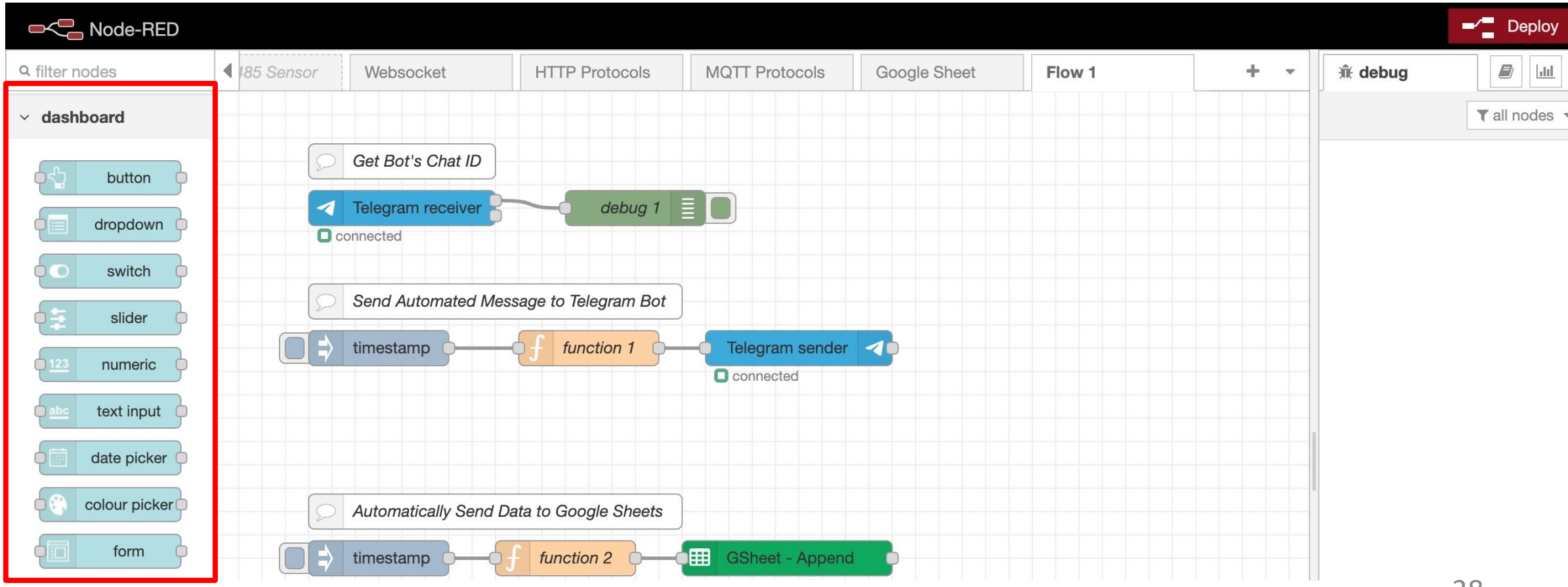
Install Node-RED Dashboard

- Open Node-RED Palette Manager.
- Click on the **Install** tab and on the **search modules** fields type **dashboard** to filter the list of the available modules.
- Click the **install** button to install the **node-red-dashboard** module and wait until the installation process is successful.



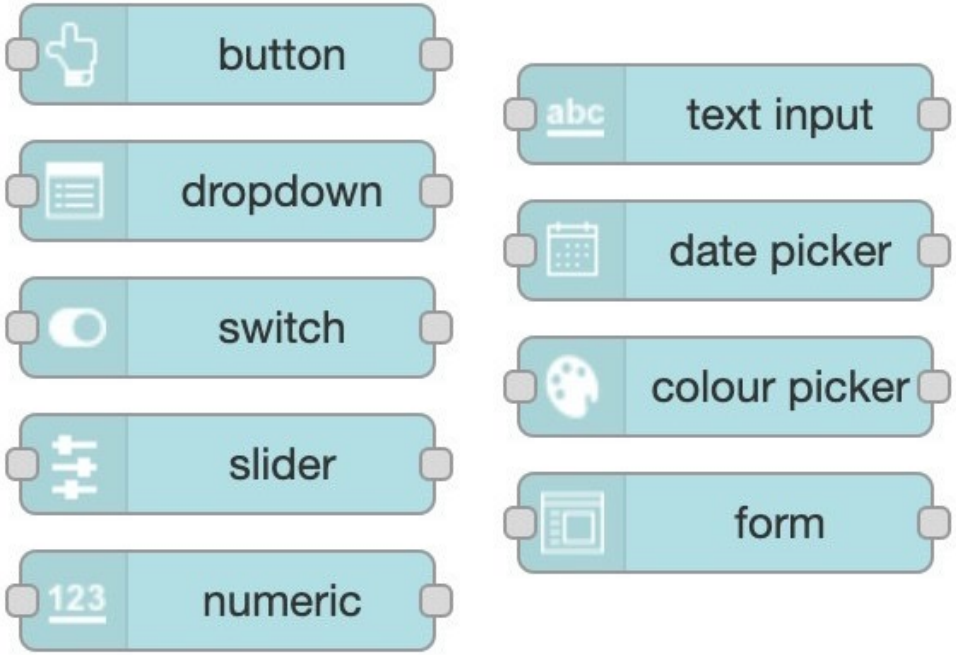
Node-RED Dashboard

- Once the installation is successful, the dashboard's widget as a set of Node-RED Dashboard nodes is available on the palette under new category: **dashboard**.



Node-RED Dashboard

Input UI nodes for the Dashboard



Example of Input UI on the Dashboard

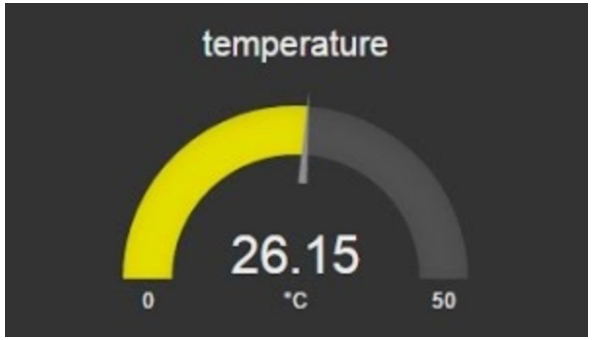
- Form and
- button

Node-RED Dashboard

Output UI nodes for the Dashboard

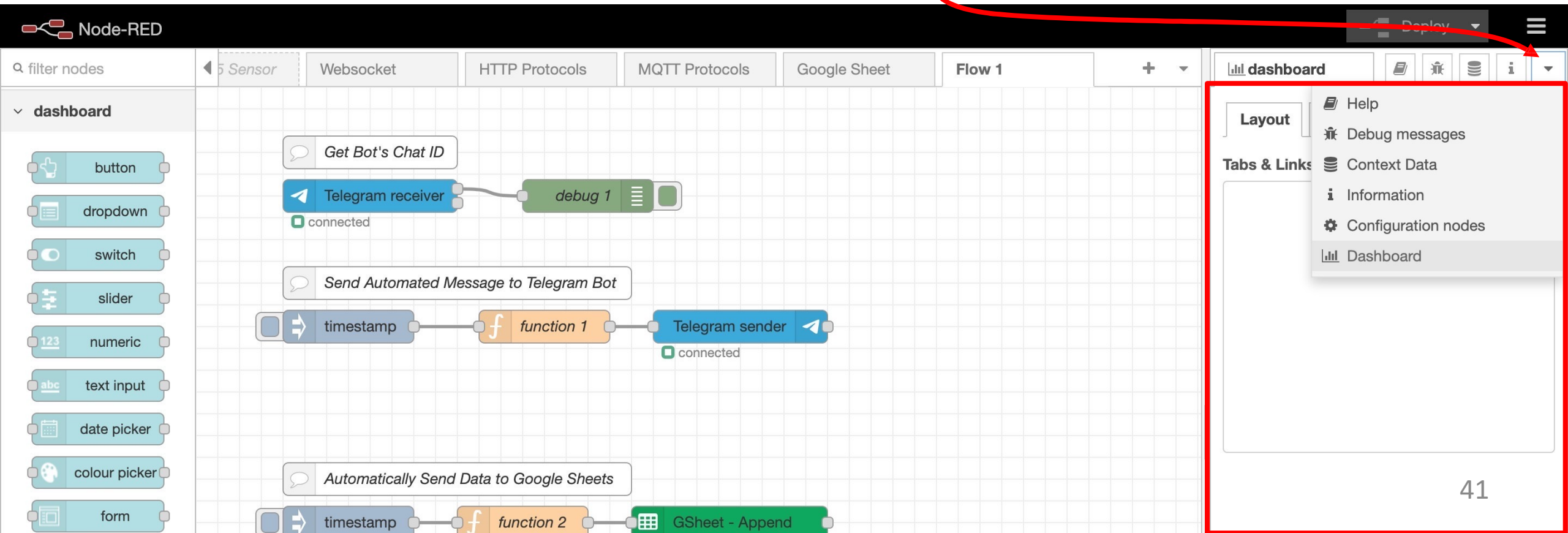
Example of Input UI on the Dashboard

- Gauge and
- Chart



Node-RED Dashboard

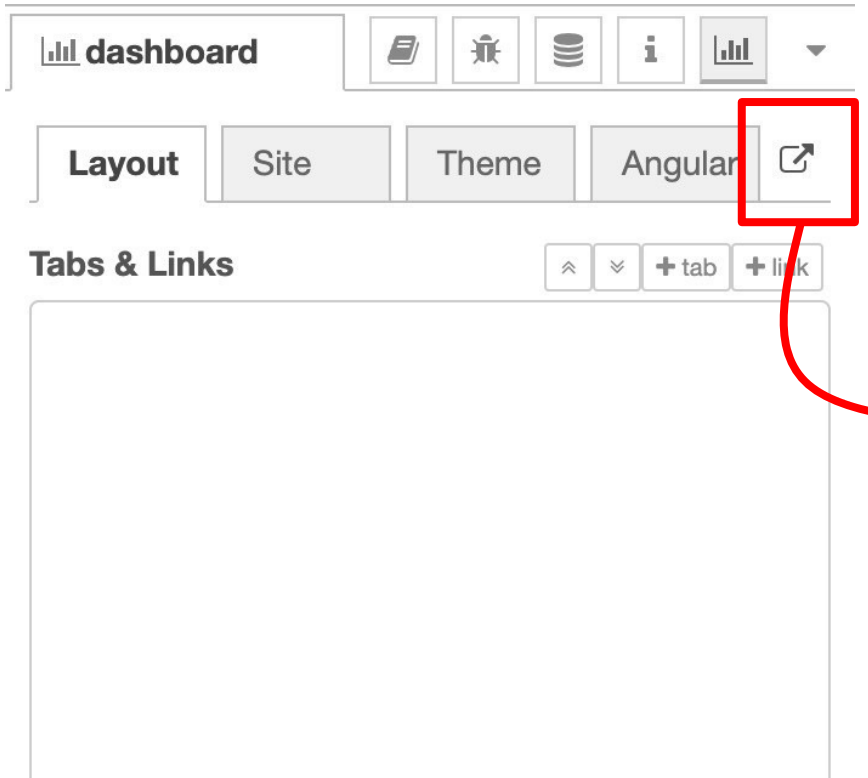
- Node-RED Sidebar also has a new tab specifically for dashboard configurations.
- The **dashboard** tab can be access by clicking the dropdown and select Dashboard.

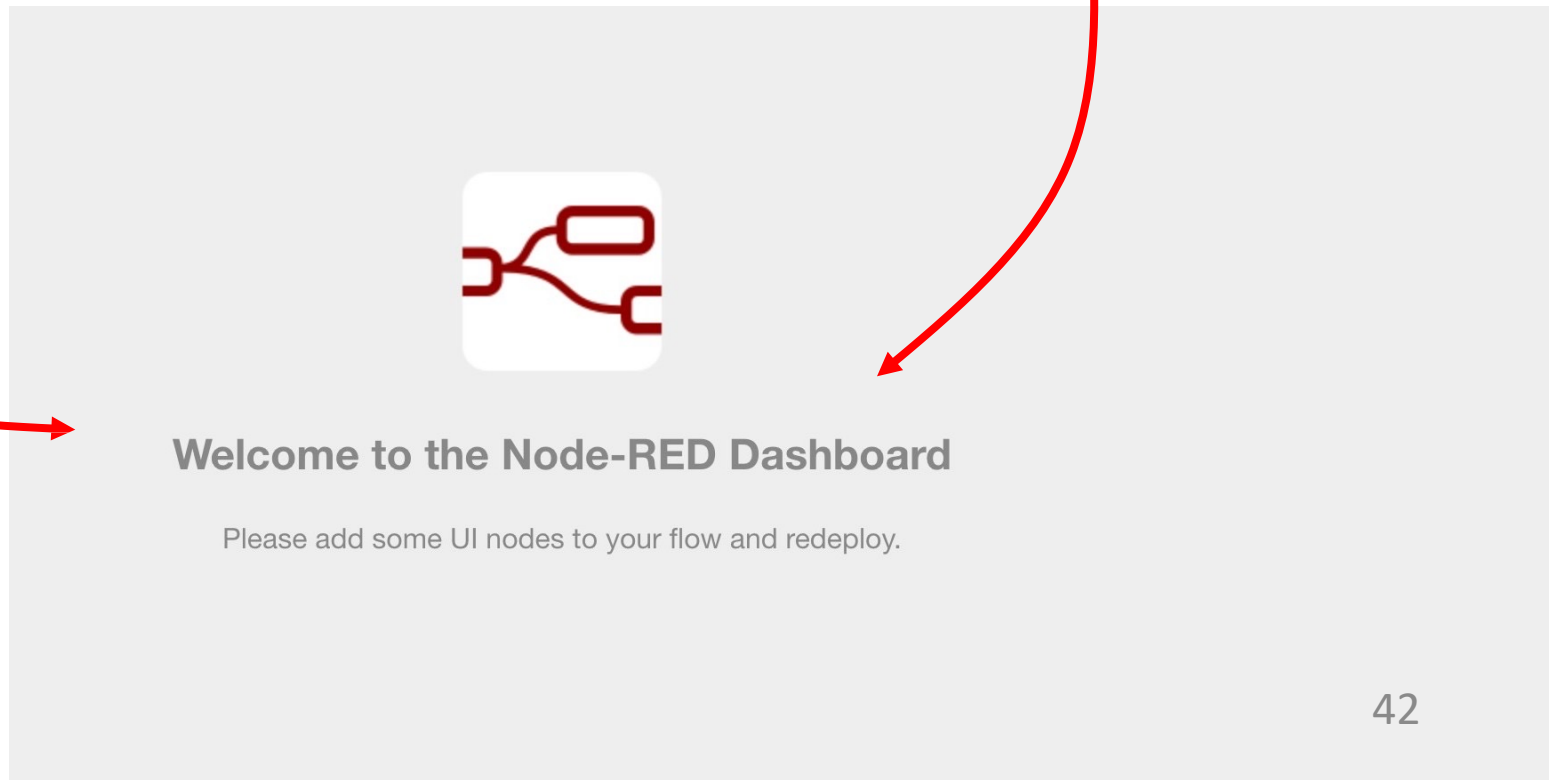


Node-RED Dashboard

There are two ways to access to Node-RED dashboard.

1. Go to the Node-RED URLs followed by **/ui**. For example, http://localhost:1880/ui
2. Click the **external icon** on the Dashboard tab under Node-RED sidebar.

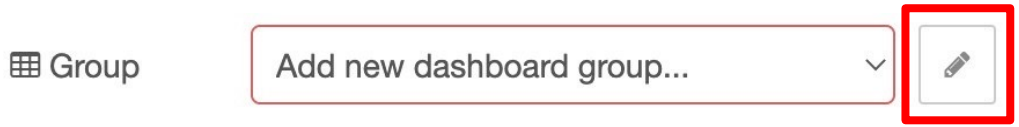




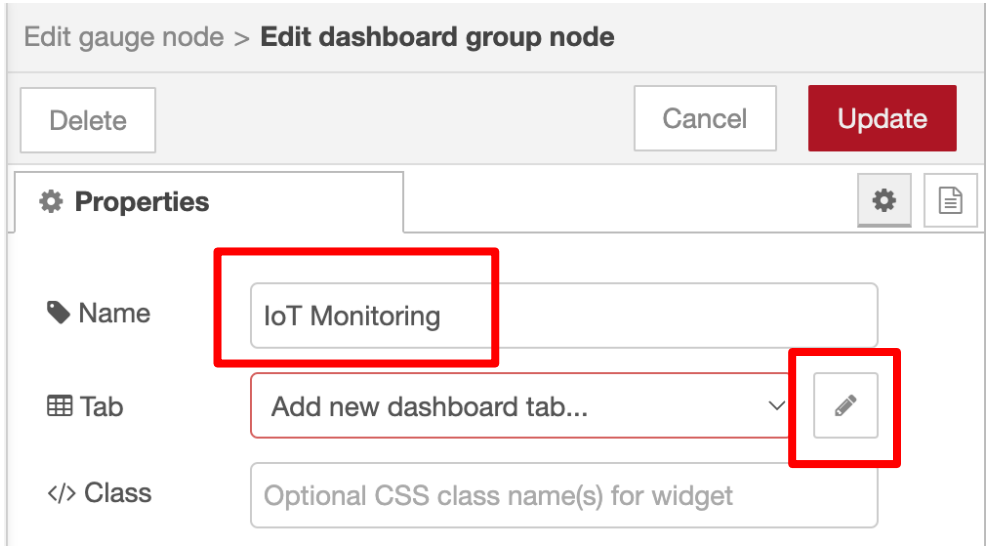
Node-RED Dashboard

Add Gauge widget to the Node-RED Dashboard (17 Steps)

1. Insert **gauge** node into the workspace.
2. Double click the node to edit the properties.
3. Setup the dashboard's group by clicking the pencil icon.

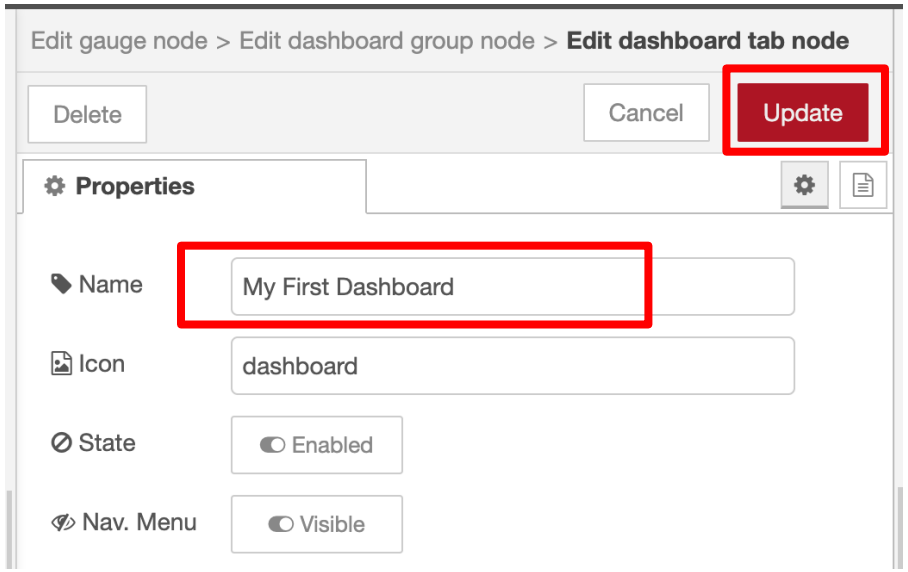


4. Rename the group's name, such as **IoT Monitoring**.
5. Click the pencil icon to add new dashboard tab.



Node-RED Dashboard

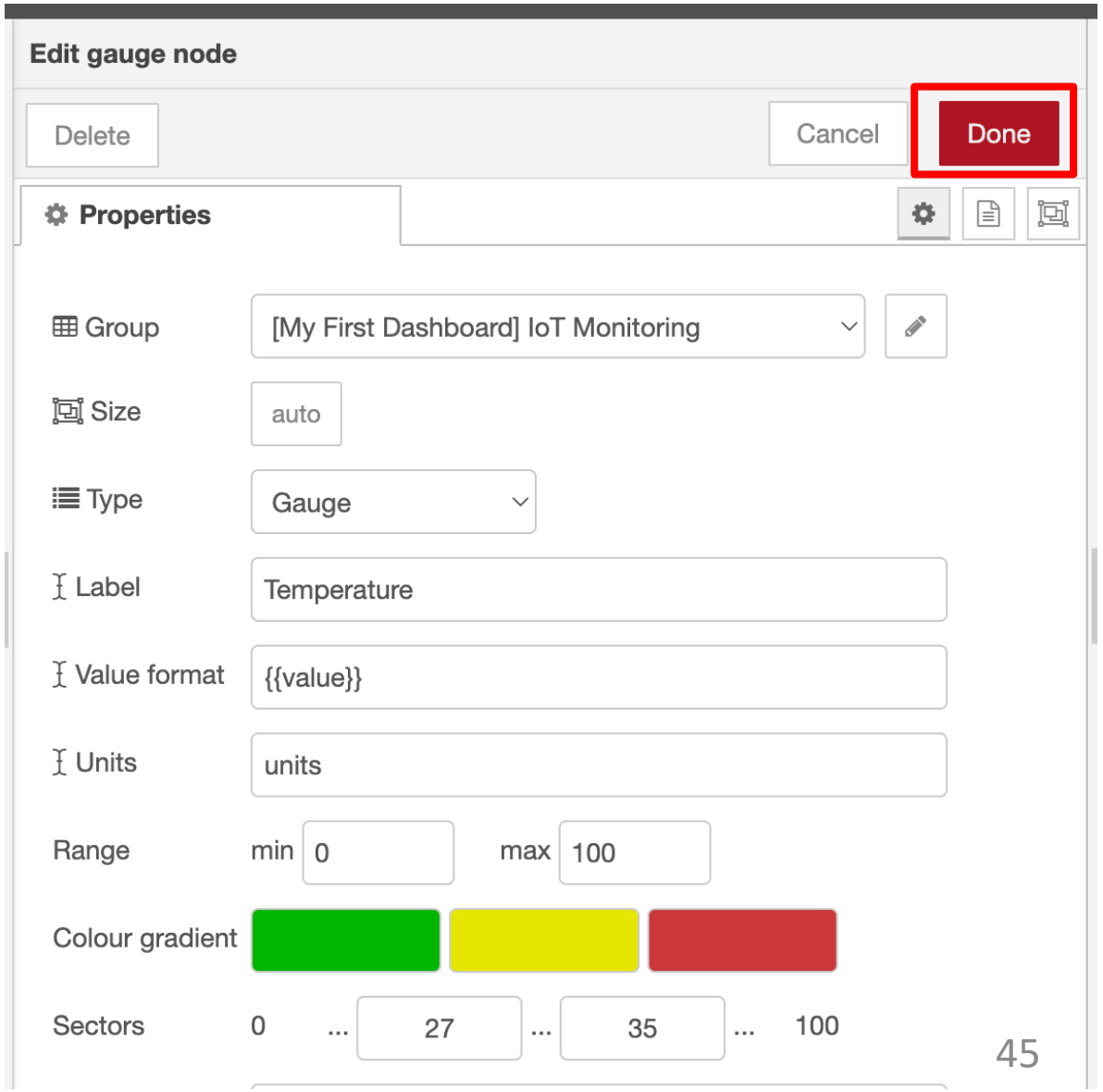
6. Rename the dashboard's tab name, such as **My First Dashboard**.
7. Click the **Add** button to add the new dashboard's tab.



8. Click the **Add** button to add the group.

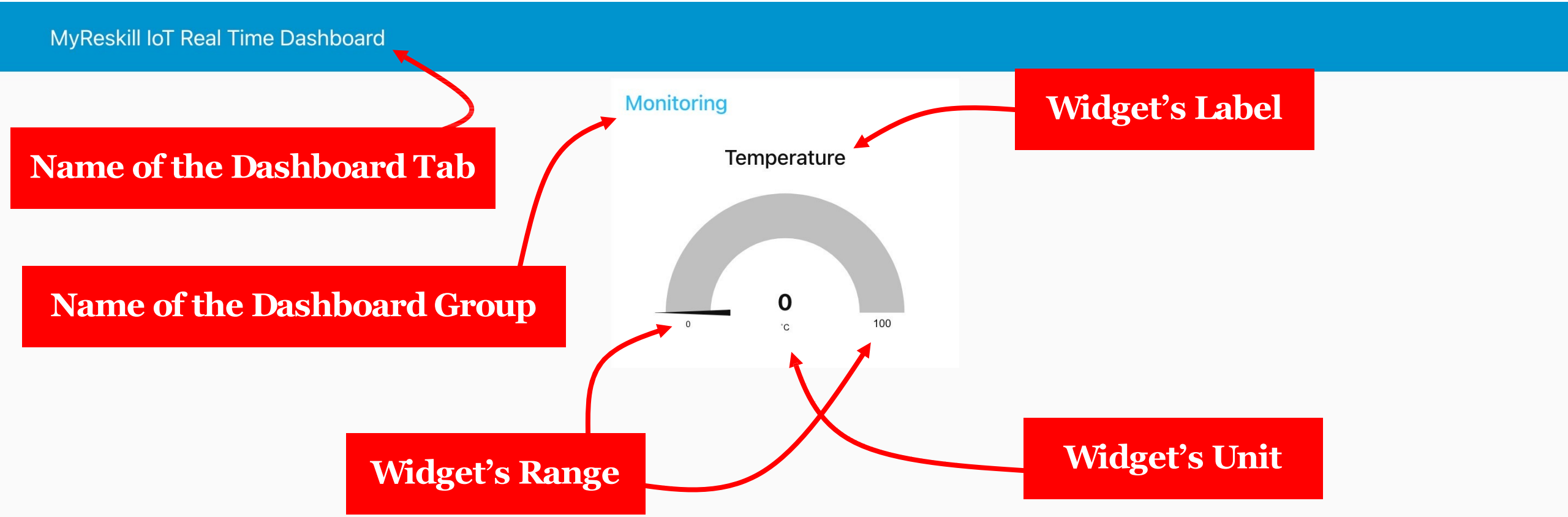
Node-RED Dashboard

9. Remain the type of the gauge is **Gauge**.
10. Rename the label to **Temperature**.
11. Format of the value remain **{{value}}**.
12. Insert the unit as Degree Celsius symbol.
13. Range of the value, min: 0 and max: 100.
14. Other configurations is default.
15. Click the **Done** button to save the configurations.
16. Click the **Deploy** button.
17. Access the Node-RED Dashboard URL to view the dashboard with the Gauge widget.



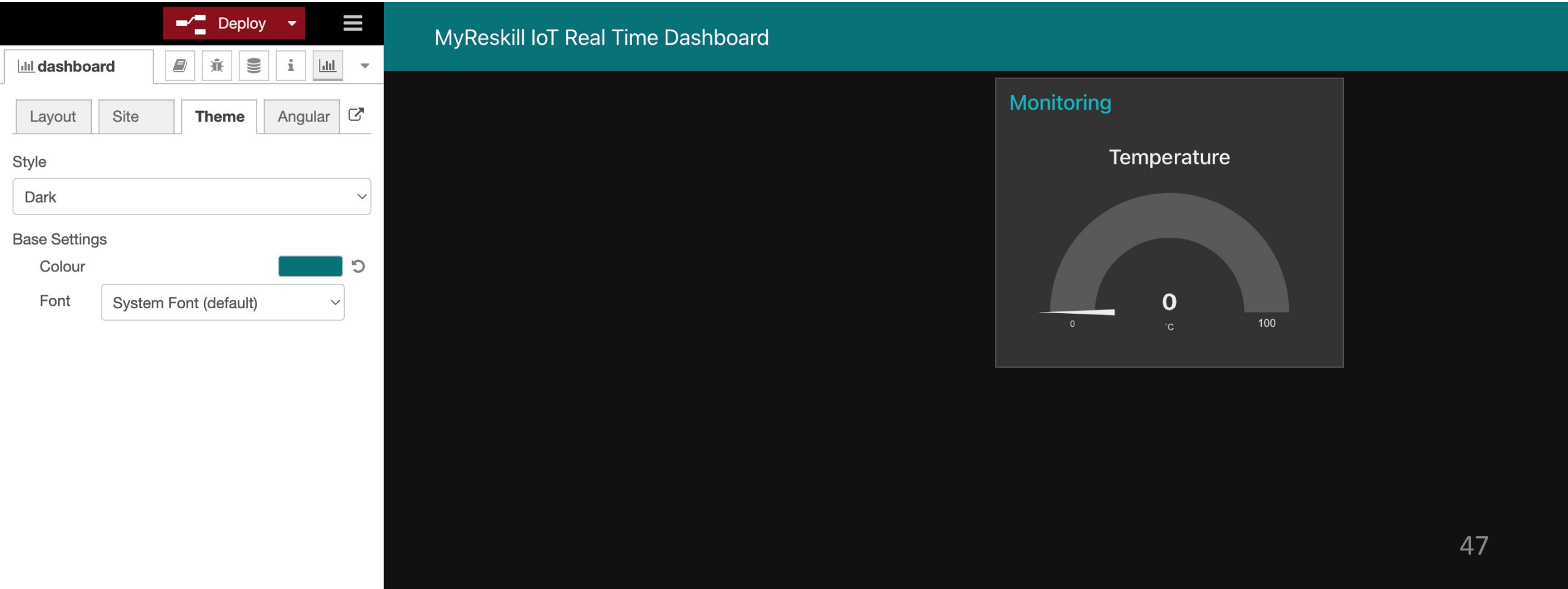
Node-RED Dashboard

Node-RED Dashboard light theme (default) with the Gauge widget.



Node-RED Dashboard

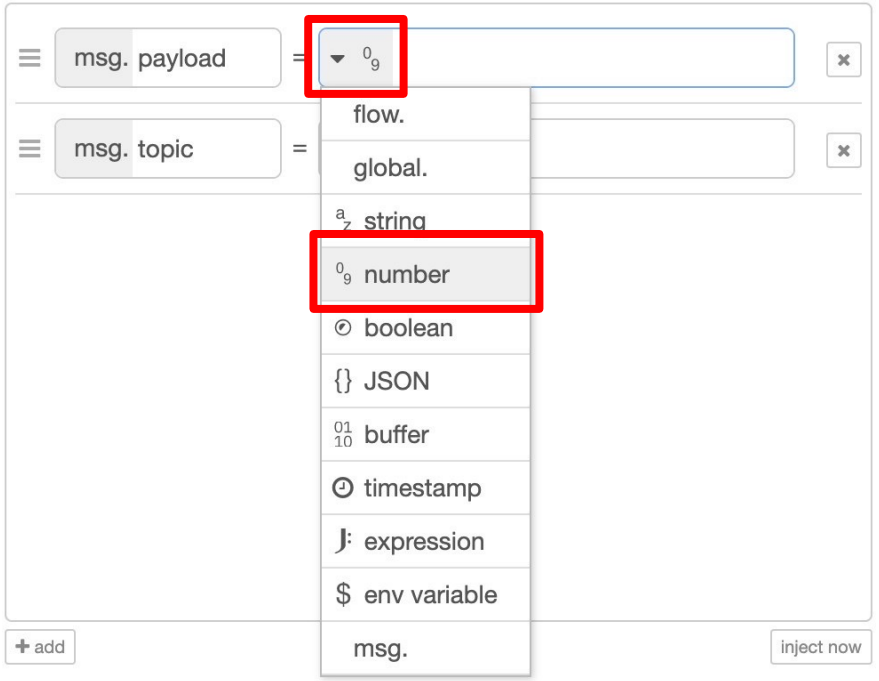
Node-RED Dashboard Theme Dark.

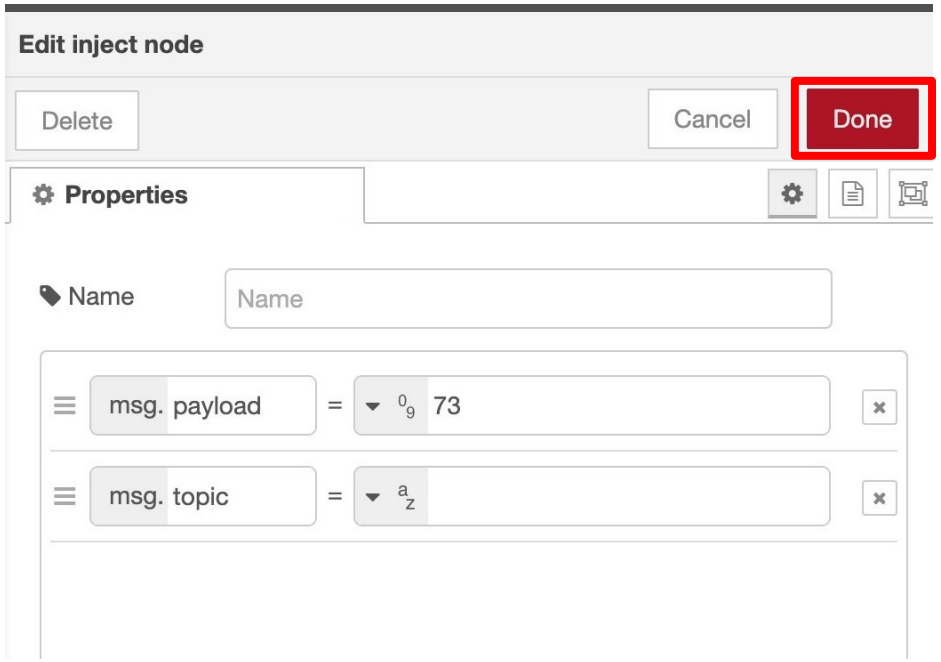


Node-RED Dashboard

Send Dummy Data to Gauge Widget (6 Steps)

1. Insert the **Inject** node into the workspace.
2. Double click the node and change the **msg.payload** data type to **number** and insert dummy value, such as 73 or 28 and click the **Done** button.

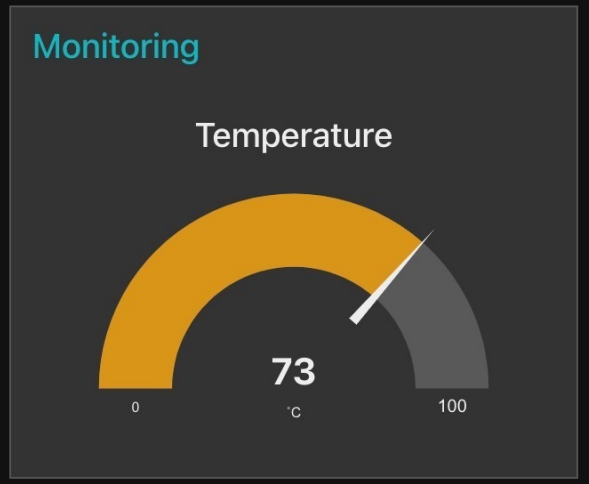




Node-RED Dashboard

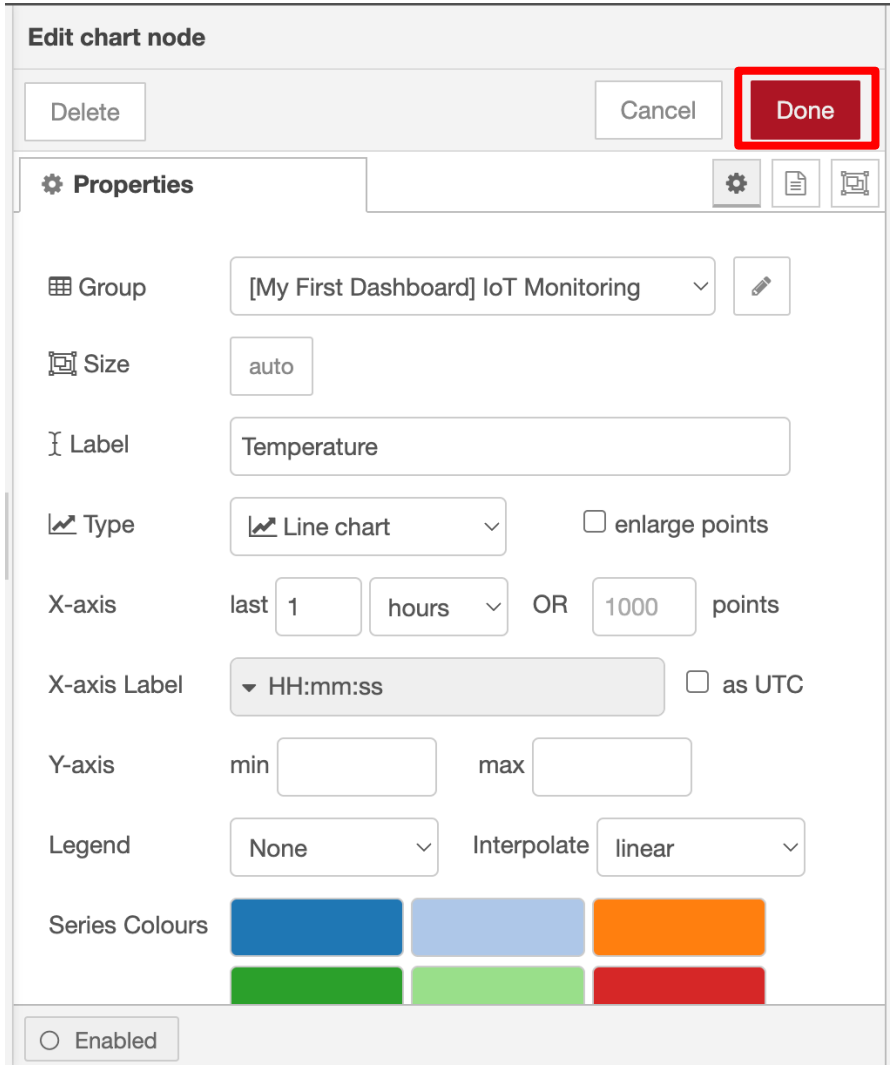
3. Connect both of the nodes, the inject node and the gauge node.
4. Click the **Deploy** button.
5. Click the inject node button to inject the data to the gauge node.
6. View the dashboard for real-time data update.

MyReskill IoT Real Time Dashboard

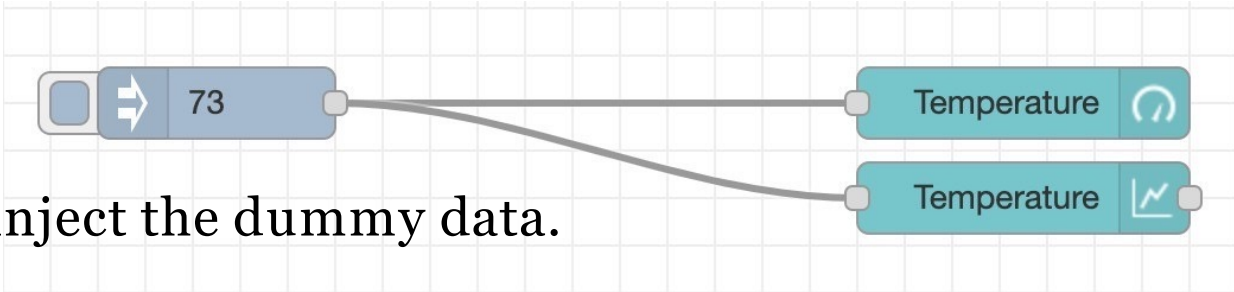


Add Chart widget to the Node-RED Dashboard (13 Steps)

1. Insert **chart** node into the workspace.
2. Double click the node to edit the properties.
3. Click the group and choose **Add new dashboard group...**
4. Click the **pencil icon**.
5. Change the default group name to **Time Series Data**.
6. Click the **Add** button to continue.
7. Change the label to **Temperature**.
8. Y-axis, min value is 0 and max value is 100.
9. Click the **Done** button.
10. Connect the **chart** node with the dummy data **inject** node.

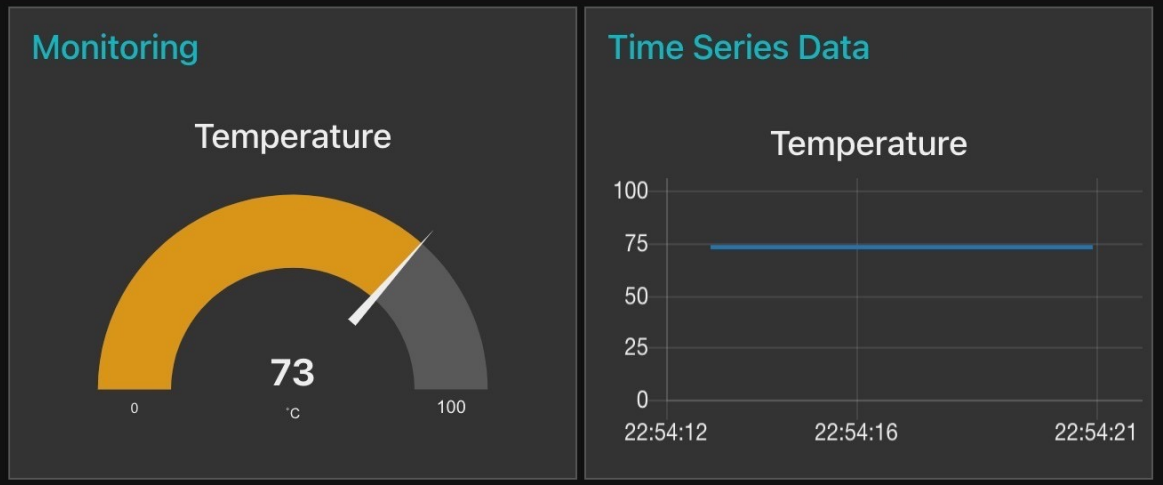


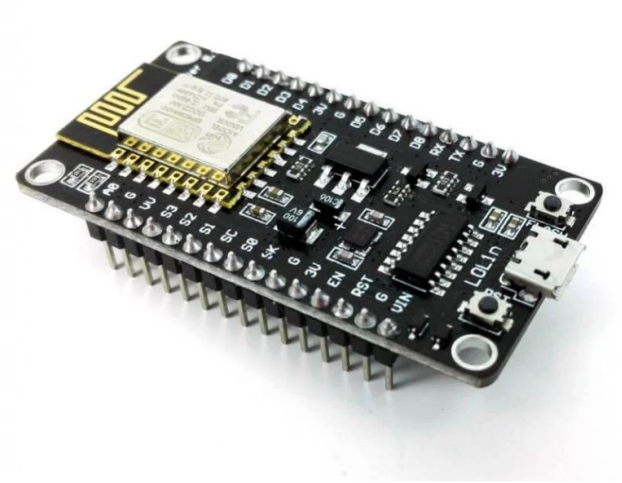
Node-RED Dashboard



11. Click the **Deploy** button.
12. Click the **inject** node button several time to inject the dummy data.
13. View the dashboard for real-time data update.

MyReskill IoT Real Time Dashboard

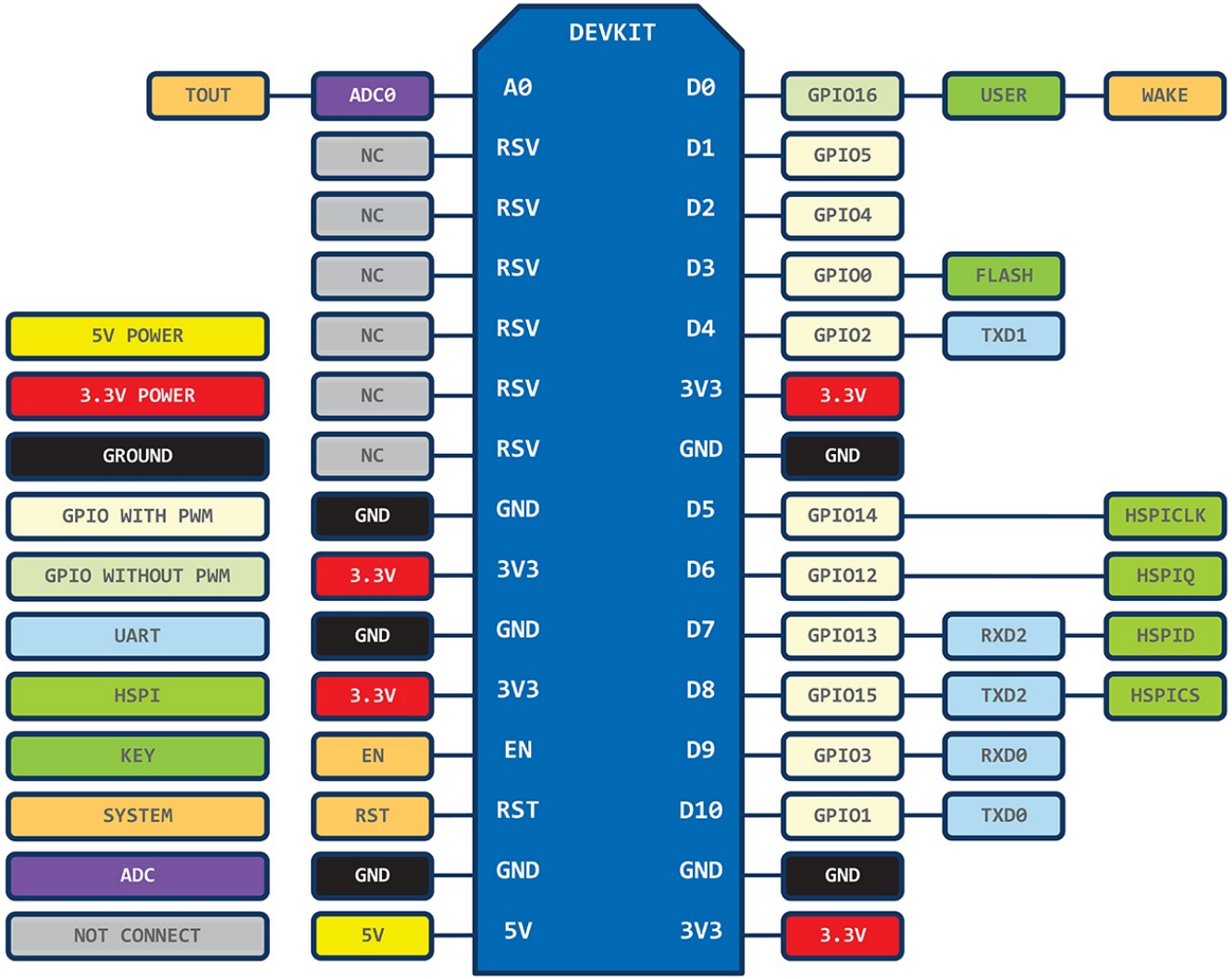




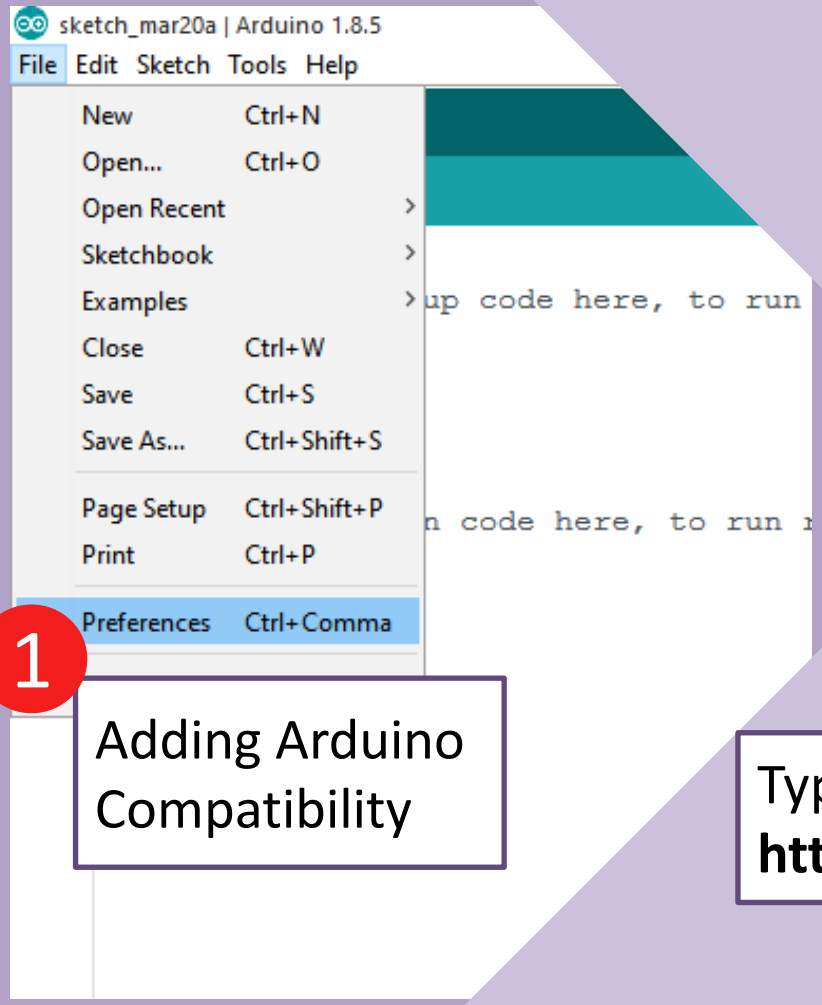
Getting Started with NodeMCU

Setting up NodeMCU with input and output

NodeMCU PIN Mapping

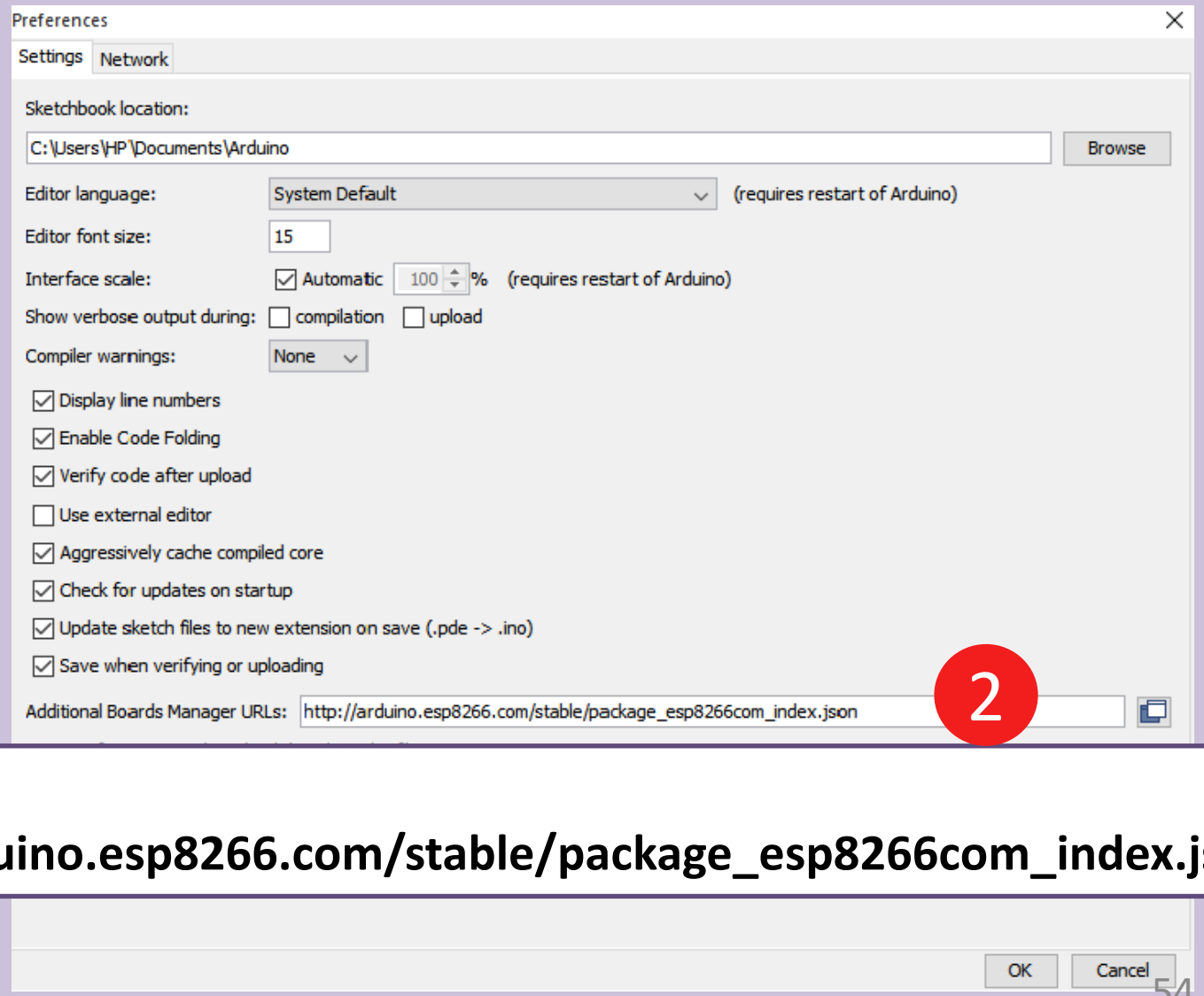


NodeMCU Setup



1

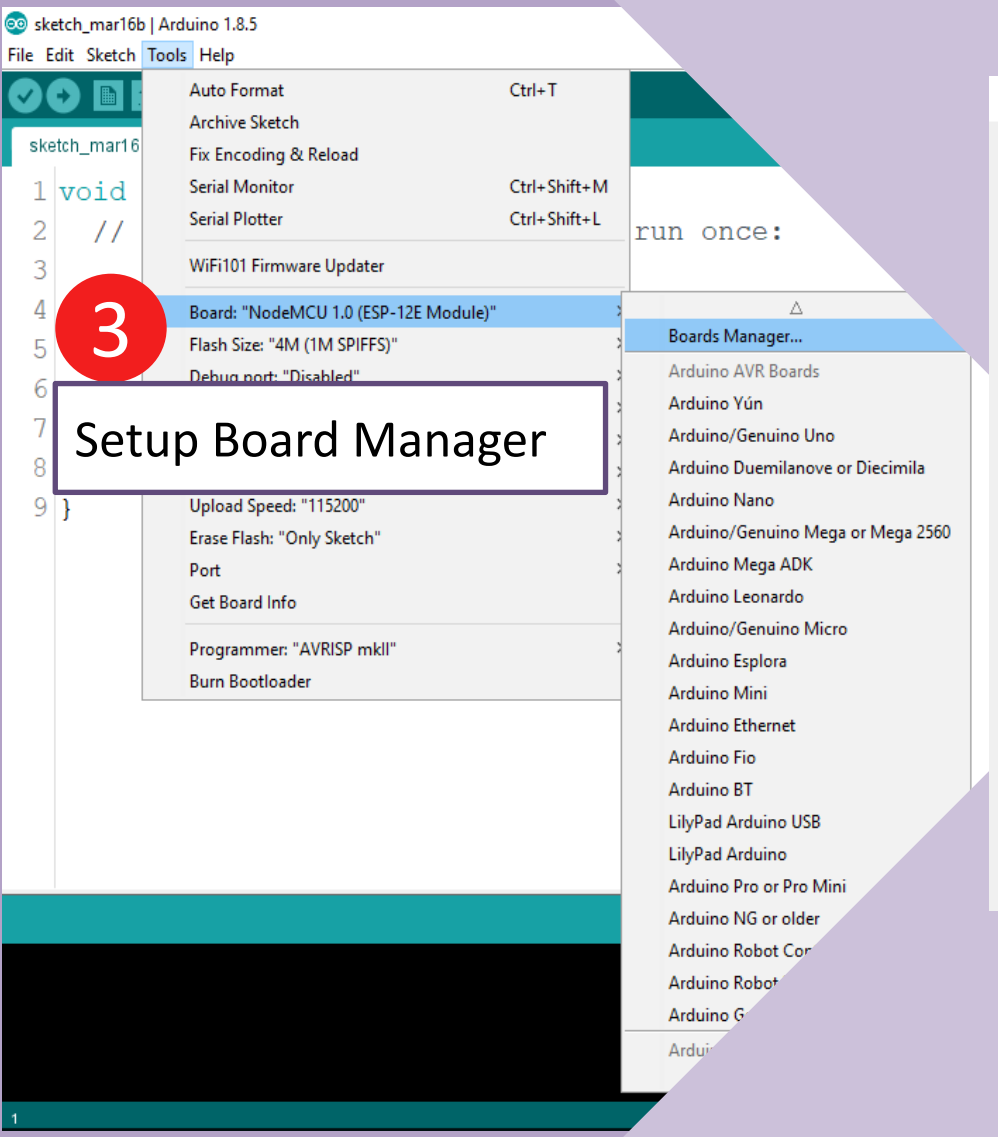
Adding Arduino Compatibility

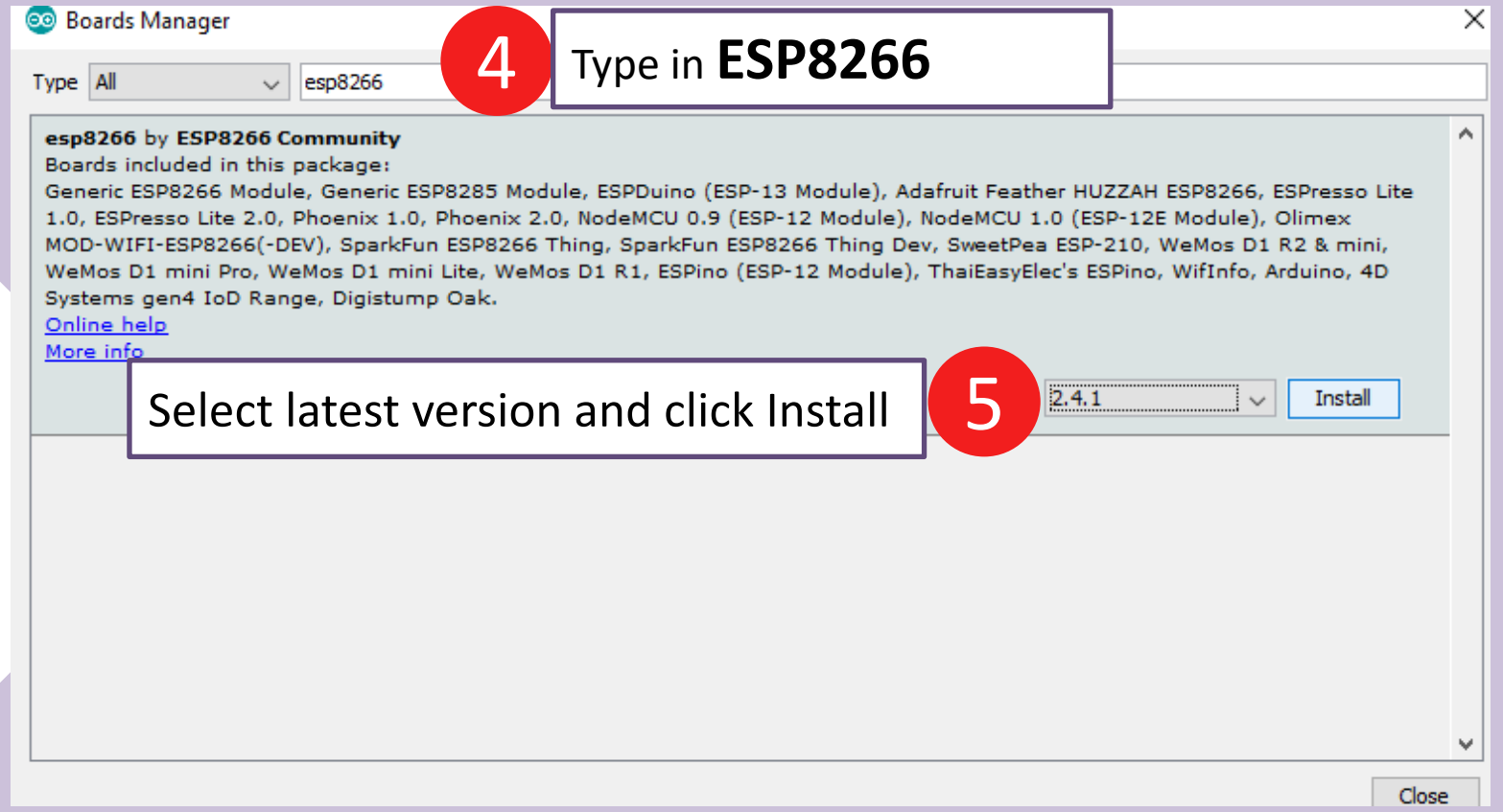


2

Type in:
http://arduino.esp8266.com/stable/package_esp8266com_index.json

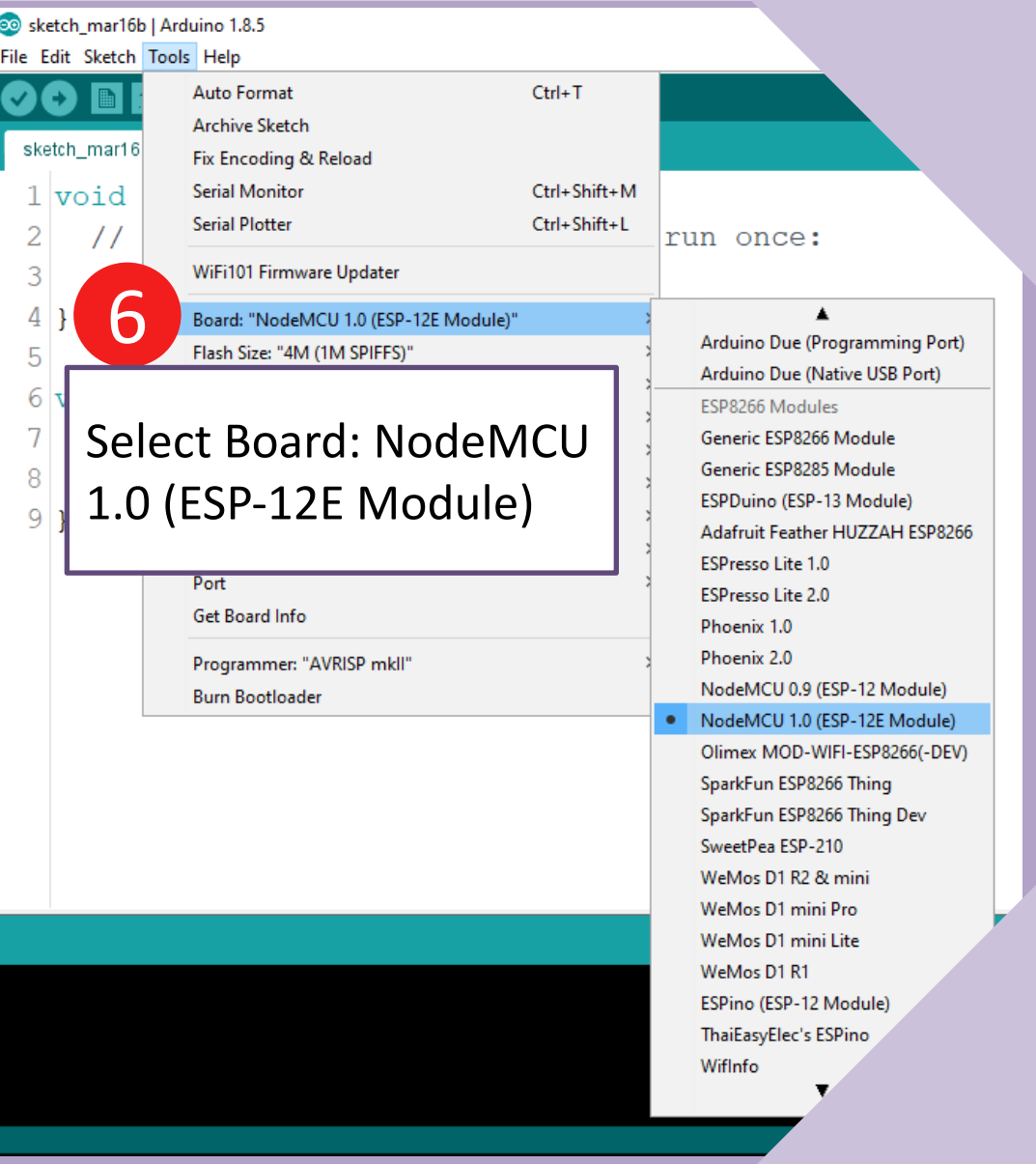
NodeMCU Setup

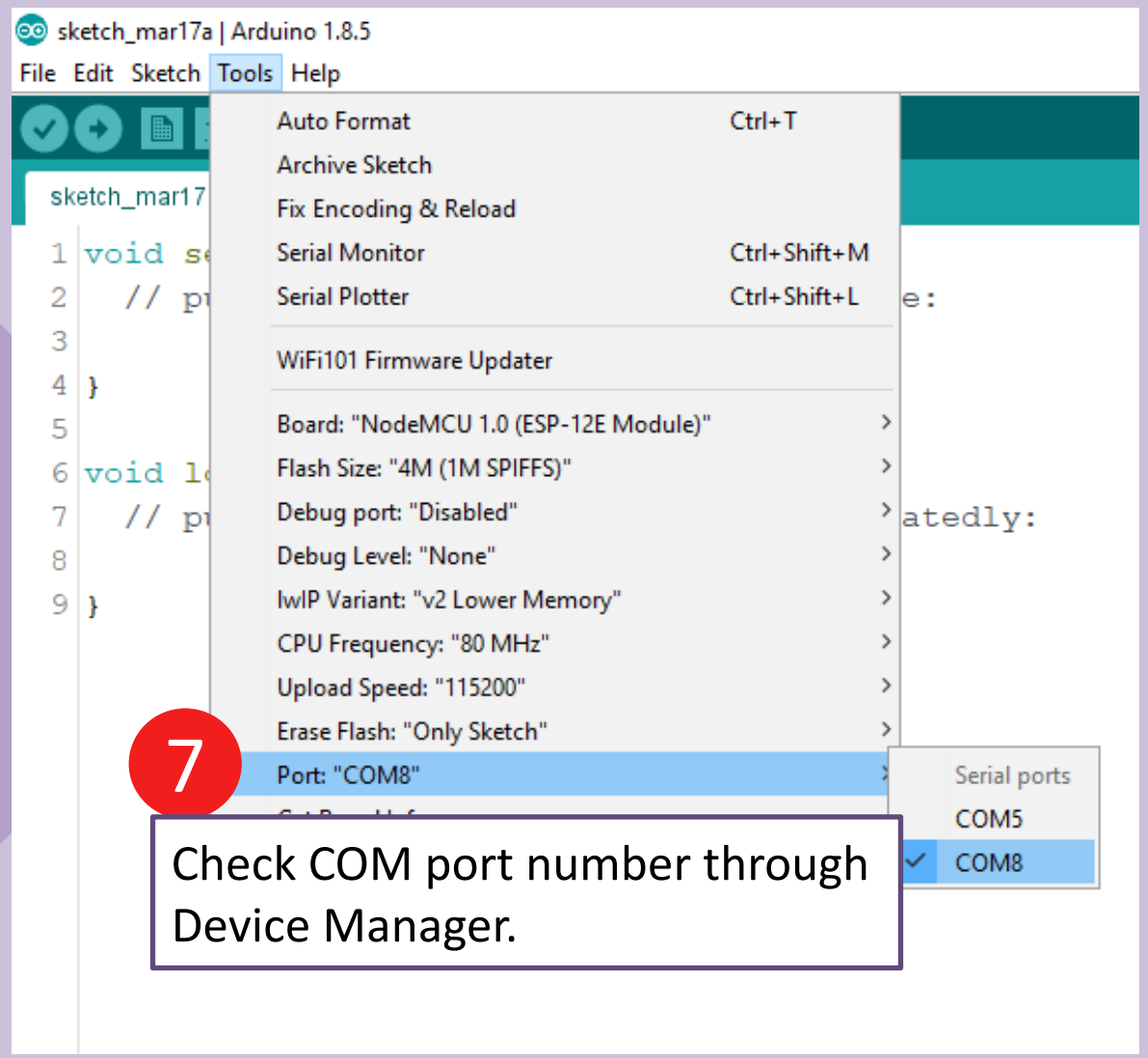




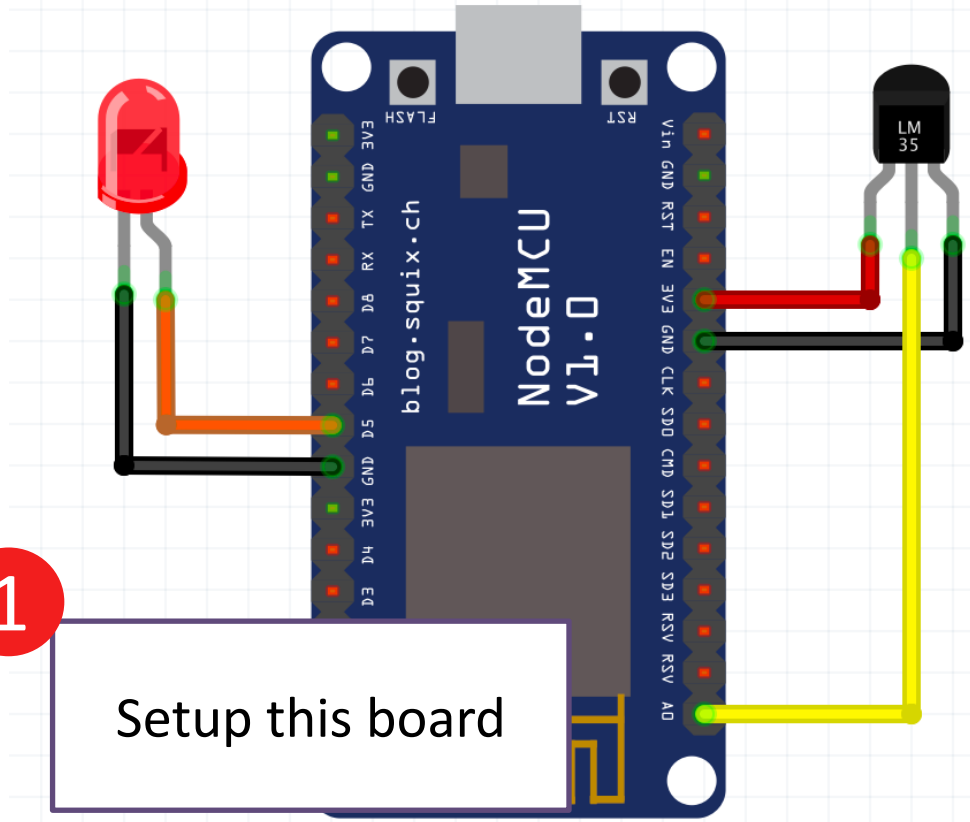
✓ Click **OK**, when the installation process is completed.

NodeMCU Setup





Getting Started with NodeMCU



1

Setup this board

Upload to board

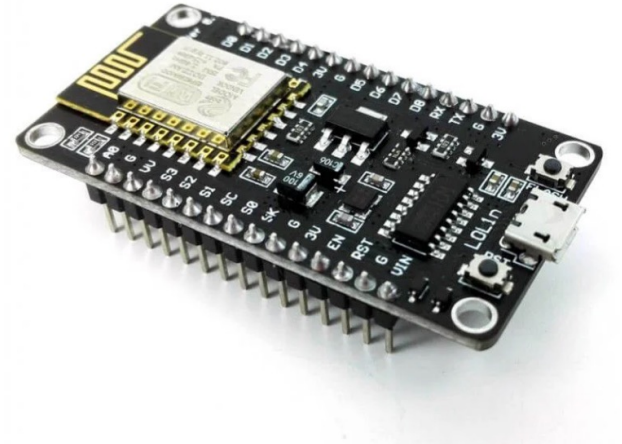
3

```
robopro_nodemcu
#include "ESP8266WiFi.h"

int temp_pin= A0;
int led1_pin= D5;

void setup(void)
{
  Serial.begin(9600);
  pinMode(led1_pin, OUTPUT);
  digitalWrite(led1_pin, LOW);
}

void loop() {
  int analogValue = analogRead(temp_pin); // get reading from LM35 temperature sensor
  float millivolts = (analogValue/1024.0) * 3300; //3300 is the voltage provided by N
  float celsius = millivolts/10;

  Serial.print("Temperature = ");
  Serial.print(celsius);
  Serial.println(" °C");

  if(digitalRead(led1_pin)==0)
    digitalWrite(led1_pin, HIGH);
  else
    digitalWrite(led1_pin, LOW);

  delay(2000);
}
```

2

Write this code

Download the **robopro_nodemcu.ino** file from the WhatsApp group.



Node-RED

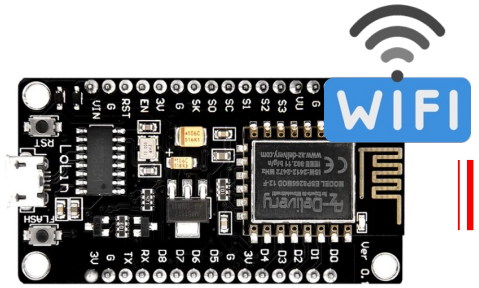
NodeMCU & Node-RED

Integrating the IoT devices with the IoT middleware

Introduction to Node-RED

Basic IoT architecture using Node-RED as the middleware.

IOT DEVICES



Data Capturing
using Sensors

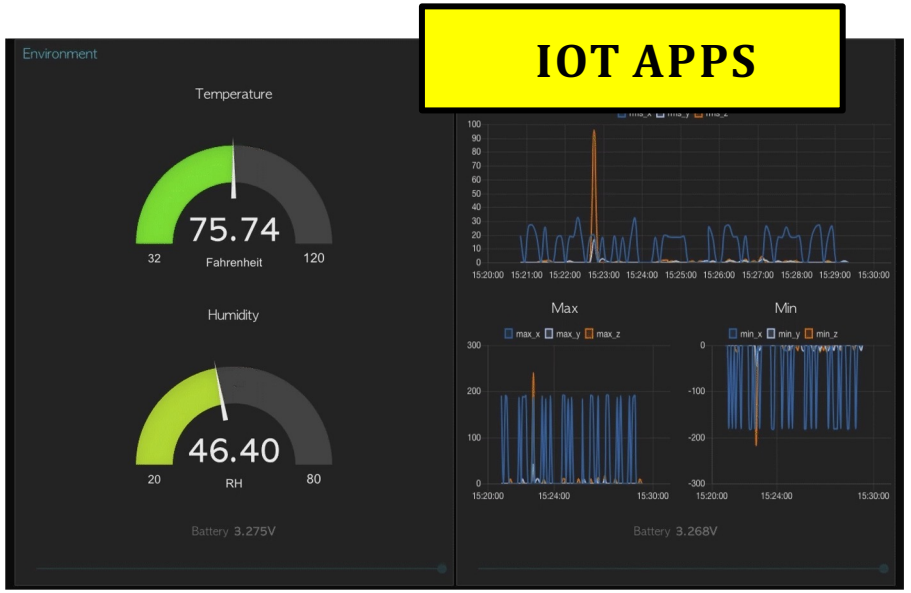
Protocol
MQTT

IOT MIDDLEWARE

Node-RED

Protocol
HTTP

IOT APPS



Data Visualization (Dashboard)
Data Monitoring

Telegram
Data Alert & Notification

Google Sheets
Data Storage

Data Studio

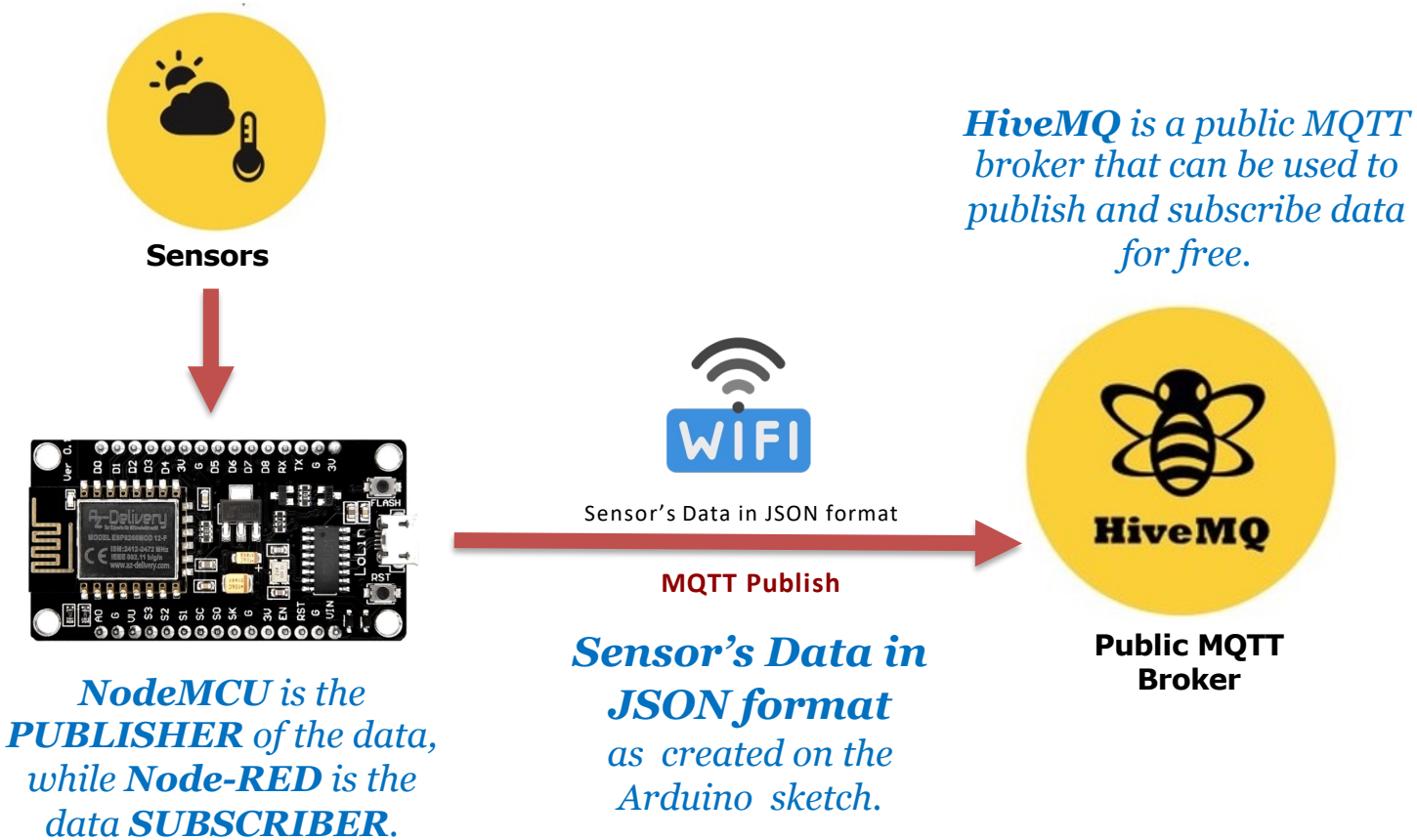
Data Visualization (Dashboard)
Data Analytics

DATA FROM THINGS, MACHINE | **HUMAN, DECISION MAKING**

NodeMCU publish MQTT to Broker

The connectivity of NodeMCU with HiveMQ public MQTT broker.

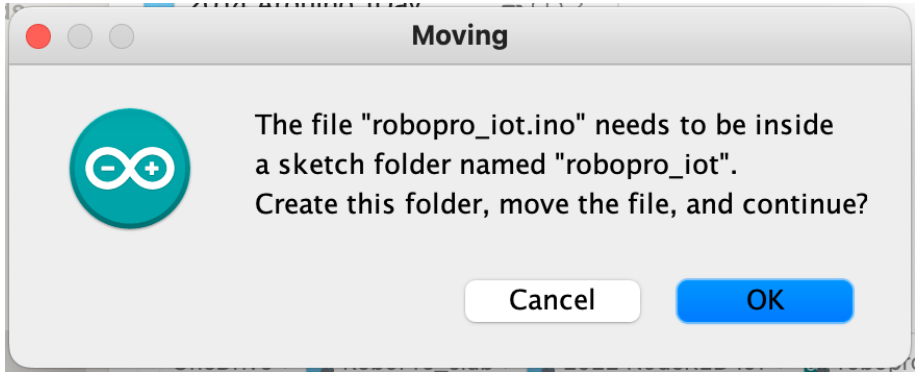
The architecture:

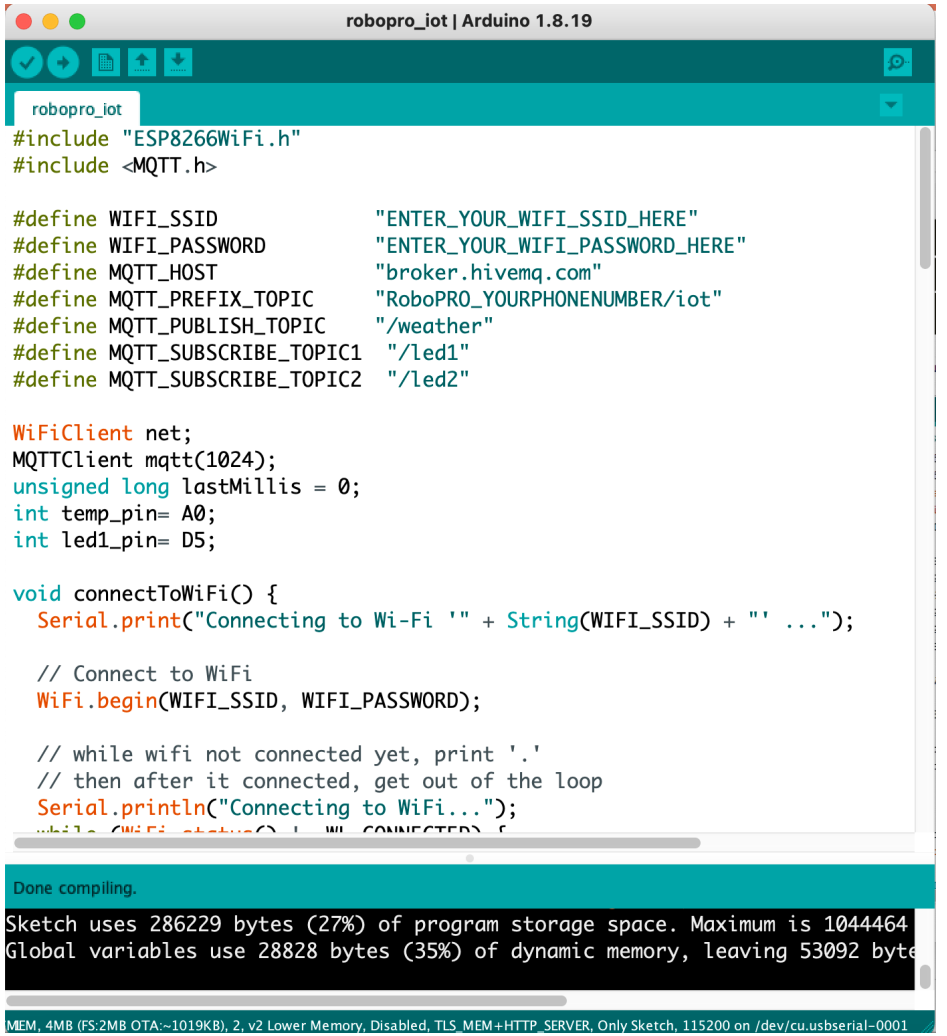


NodeMCU publish MQTT to Broker

Open NodeMCU Example Sketch (3 Steps)

1. Download the **robopro_iot.ino** file from the WhatsApp group.
2. Double click the file to open it on Arduino IDE.
3. Click **OK** on the Moving pop-up window. The file will be added inside a folder with named followed by the file name.

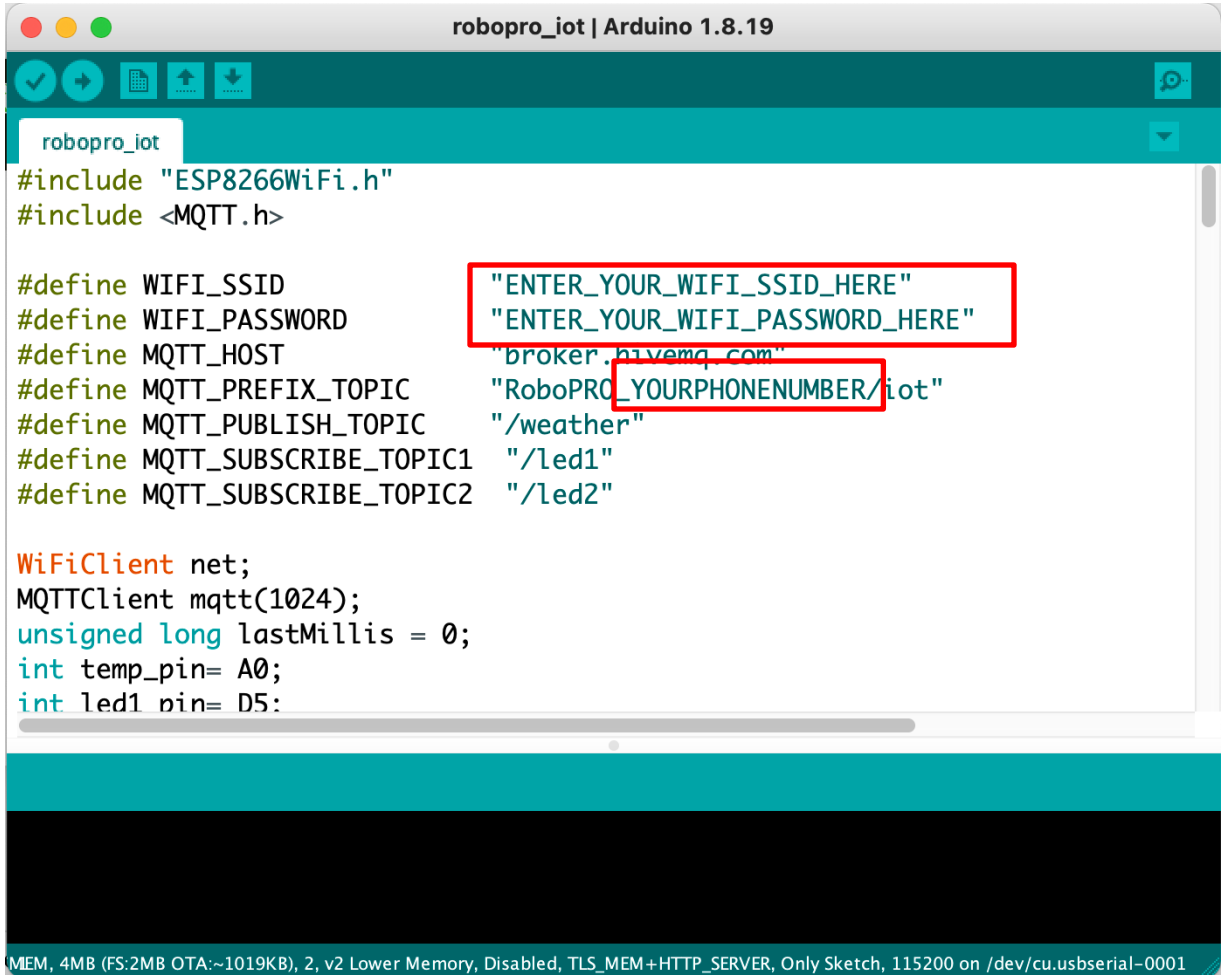




NodeMCU publish MQTT to Broker

ARDUINO SKETCH EXPLAIN

It is a complete program, as sets of function for ESP8266 microcontroller to execute tasks:



EXECUTE ONCE

1. Connect to nearest Wi-Fi access point.
2. Connect to public MQTT broker.

EXECUTE REPEATEDLY

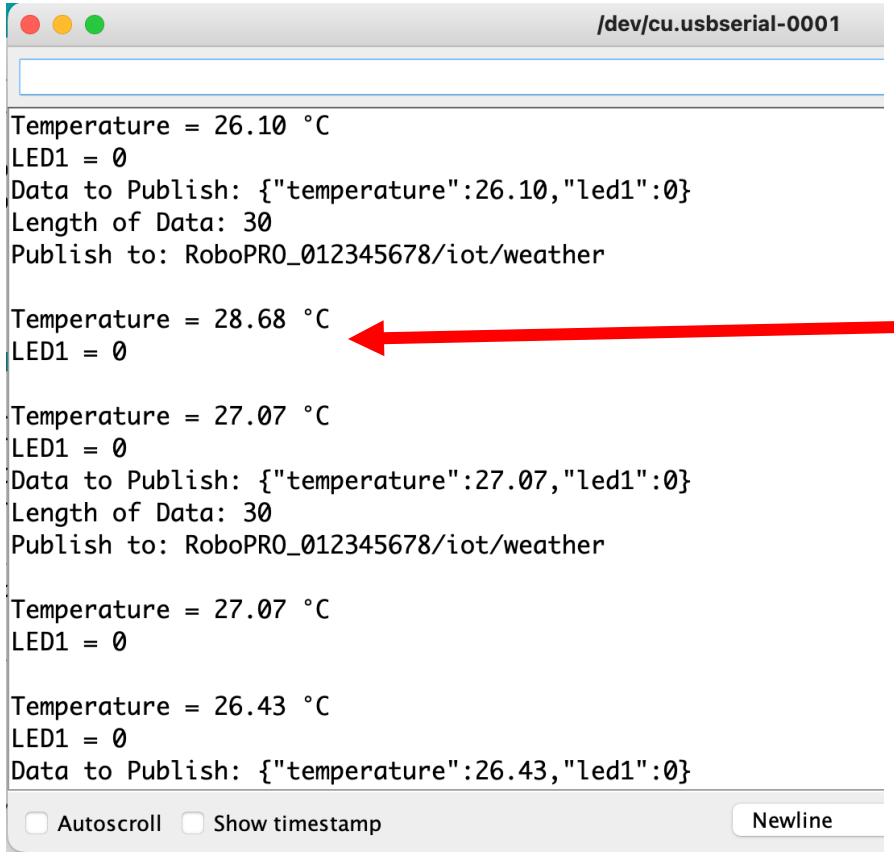
3. Check if ESP8266 Wi-Fi's disconnected to reconnect to the Wi-Fi access point.
4. Check if connection to public MQTT broker disconnected to reconnect.

NodeMCU publish MQTT to Broker

5. Read LM35 sensor value:
   - Temperature, °C
6. Read LED1 pin value
7. Print the sensor's value and LED1 pin values on the Serial Monitor.
8. Create JSON format data from the sensor's value to send to the Node-RED over MQTT protocol.
9. The JSON format data is publish to MQTT topic by 10 seconds interval.

NodeMCU publish MQTT to Broker

The result of **Step No. 7** on the Serial Monitor as shown in the image below.



NodeMCU publish MQTT to Broker

The result of **Step No. 8** and **Step No. 9** on the Serial Monitor as shown in the image.

robopro_iot | Arduino 1.8.19

robopro_iot

```
if (millis() - lastMillis > 10000) { // publish every 10 seconds
  lastMillis = millis();

  String dataInJson = "{";
  dataInJson += "\"temperature\":" + String(celsius) + ",";
  dataInJson += "\"led1\":" + String(digitalRead(led1_pin));
  dataInJson += "}";

  Serial.println("Data to Publish: " + dataInJson);
  Serial.println("Length of Data: " + String(dataInJson.length()));
  Serial.println("Publish to: " + String(MQTT_PREFIX_TOPIC) + String(MQTT_PUBLISH_TOPIC));

  mqtt.publish(String(MQTT_PREFIX_TOPIC) + String(MQTT_PUBLISH_TOPIC), dataInJson);
}
```

/dev/cu.usbserial-0001

```
Temperature = 26.10 °C
LED1 = 0
Data to Publish: {"temperature":26.10,"led1":0}
Length of Data: 30
Publish to: RoboPRO_012345678/iot/weather

Temperature = 28.68 °C
LED1 = 0

Temperature = 27.07 °C
LED1 = 0
Data to Publish: {"temperature":27.07,"led1":0}
Length of Data: 30
Publish to: RoboPRO_012345678/iot/weather

Temperature = 27.07 °C
LED1 = 0

Temperature = 26.43 °C
LED1 = 0
Data to Publish: {"temperature":26.43,"led1":0}
```

Autoscroll Show timestamp Newline

NodeMCU publish MQTT to Broker

What is JSON?

- JSON (JavaScript Object Notation).
- Lightweight format for storing and transporting data.
- JSON syntax rules:
  - Data is in name/value pairs
  - Data is separated by commas
  - Curly braces hold objects
  - Square brackets hold arrays
- ”Self-describing" as it has key and value in pair, better than unexplainable plain text.

```
31
```

Plain Text
No description or no meaning data

```
{"age": 31 }
```

JSON
Self describe value, with helps of the pair of key and value

NodeMCU publish MQTT to Broker

JSON for Sensor's Data

- Sending the sensor's data to the cloud in JSON format is better than only plain text.
- NodeMCU read LM35 sensor and LED1 data and format them in JSON format as below.

```
{
  "temperature": 28.17,
  "led1": 0
}
```

- The data is self-described and easy to access the value from the object, just call the key. For example, call temperature key, will give the result of 28.17.
- The JSON key calling method is applicable on any system, including Node-RED.

NodeMCU publish MQTT to Broker

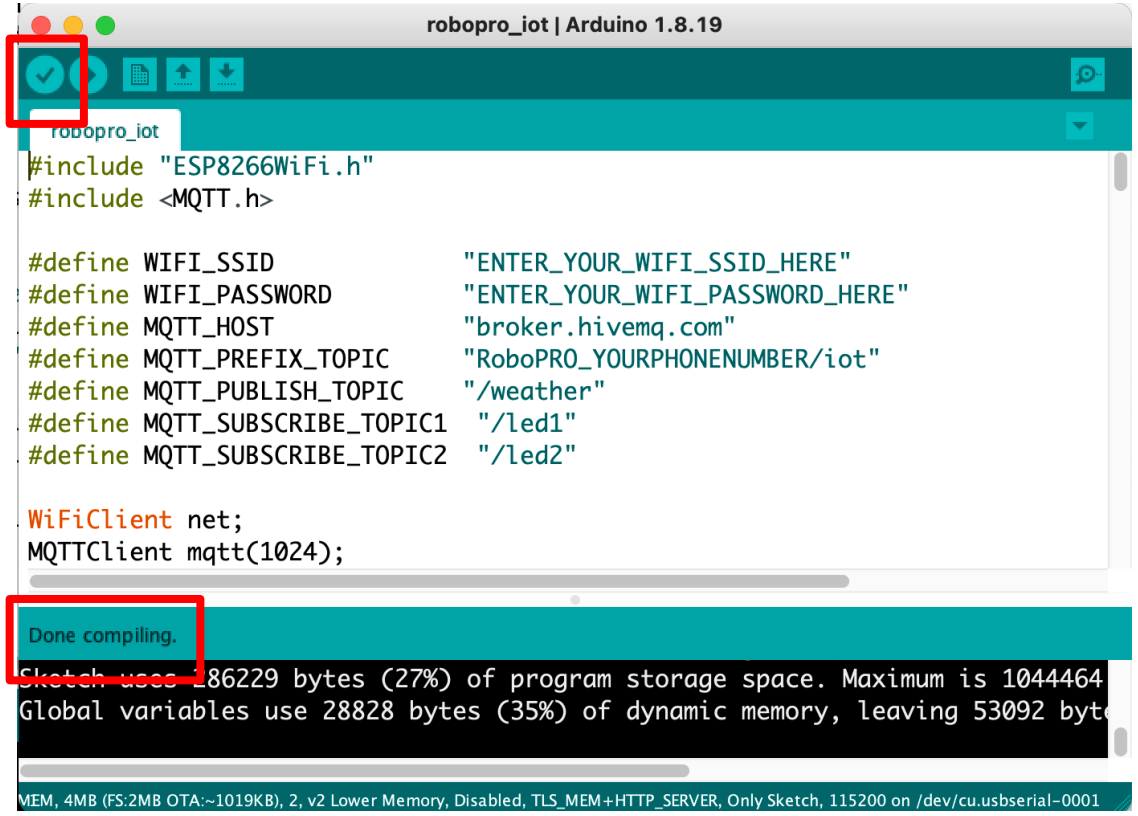
Install MQTT Library (4 Steps)

1. Go to menu, **Tools > Manage Libraries ...**

2. On Library Manager window, type in **MQTT** on the field to filter the list of libraries.
3. Click the **Install** button on the MQTT Library and wait until it is done.

NodeMCU publish MQTT to Broker

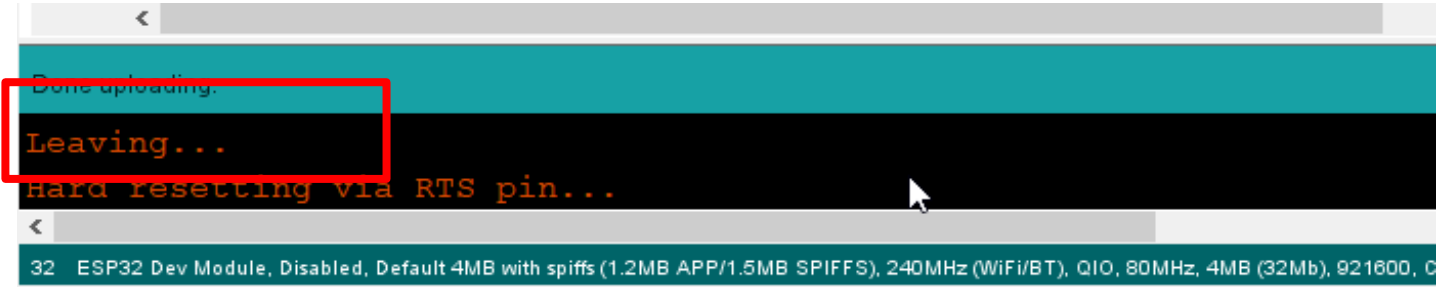
4. Click the verify button to compile the example sketch.
   If the status is **Done compiling**, the library installation is successful without error.



NodeMCU publish MQTT to Broker

Upload Example Sketch (4 Steps)

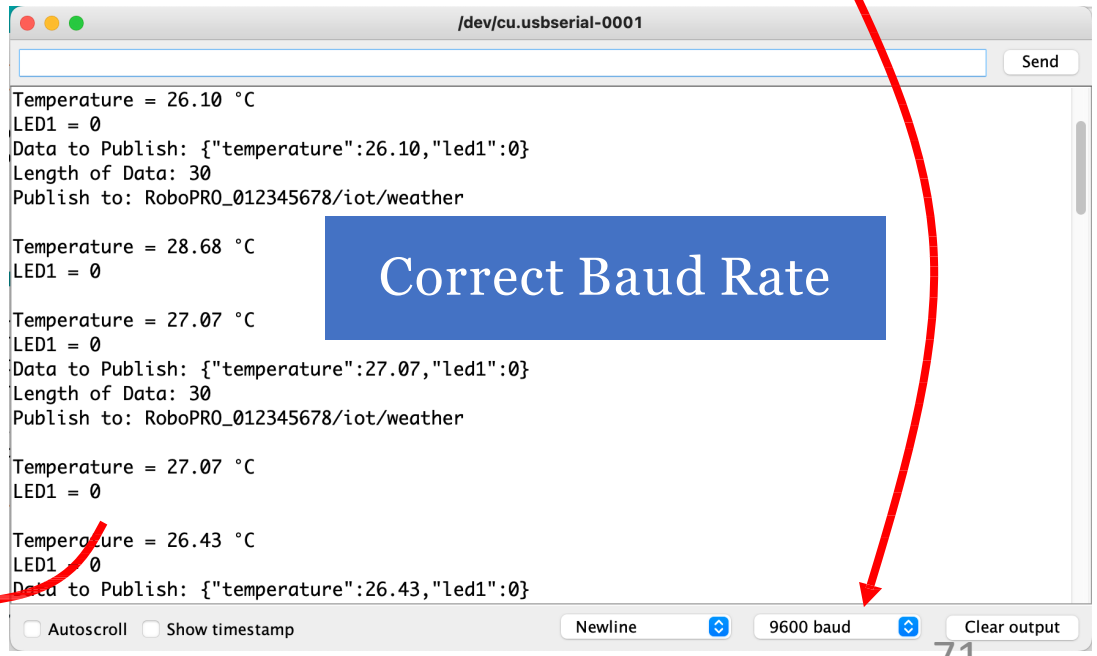
1. Replace the **WIFI_SSID** value to the correct information of your Wi-Fi name.
2. Replace the **WIFI_PASSWORD** value to the correct information your Wi-Fi password.
3. Replace the **MQTT_PREFIX_TOPIC** of mobile number with your own mobile number.
4. Click the **Upload** button to compile and upload the program into the ESP32 microcontroller, wait until the status is **Done uploading**.

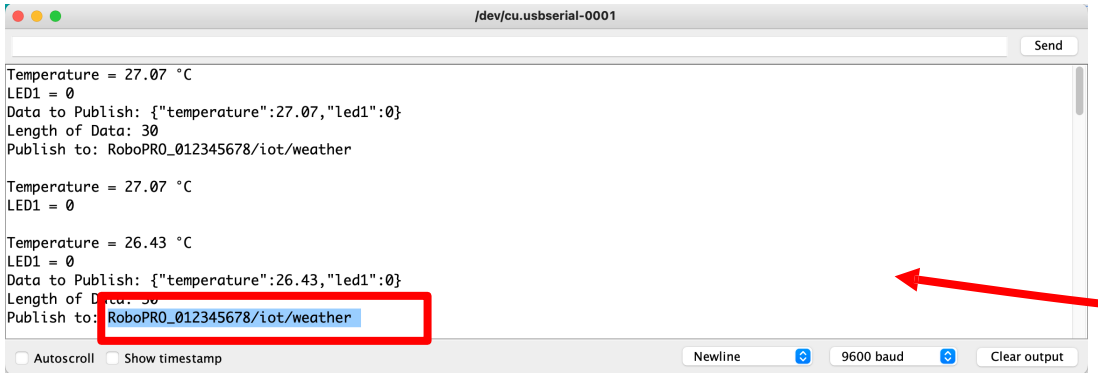


NodeMCU publish MQTT to Broker

Get the MQTT Publish Topic (3 Steps)

1. Open the Serial Monitor. If the result is not right (gibberish) as shown on the right image below, make sure the baud rate is **9600**.
2. Wait until around 10 seconds to see the data publish to MQTT topic as shown below.
3. Copy the **Publish to:** MQTT topic (as highlighted) below. This topic will be use to subscribe by Node-RED node.

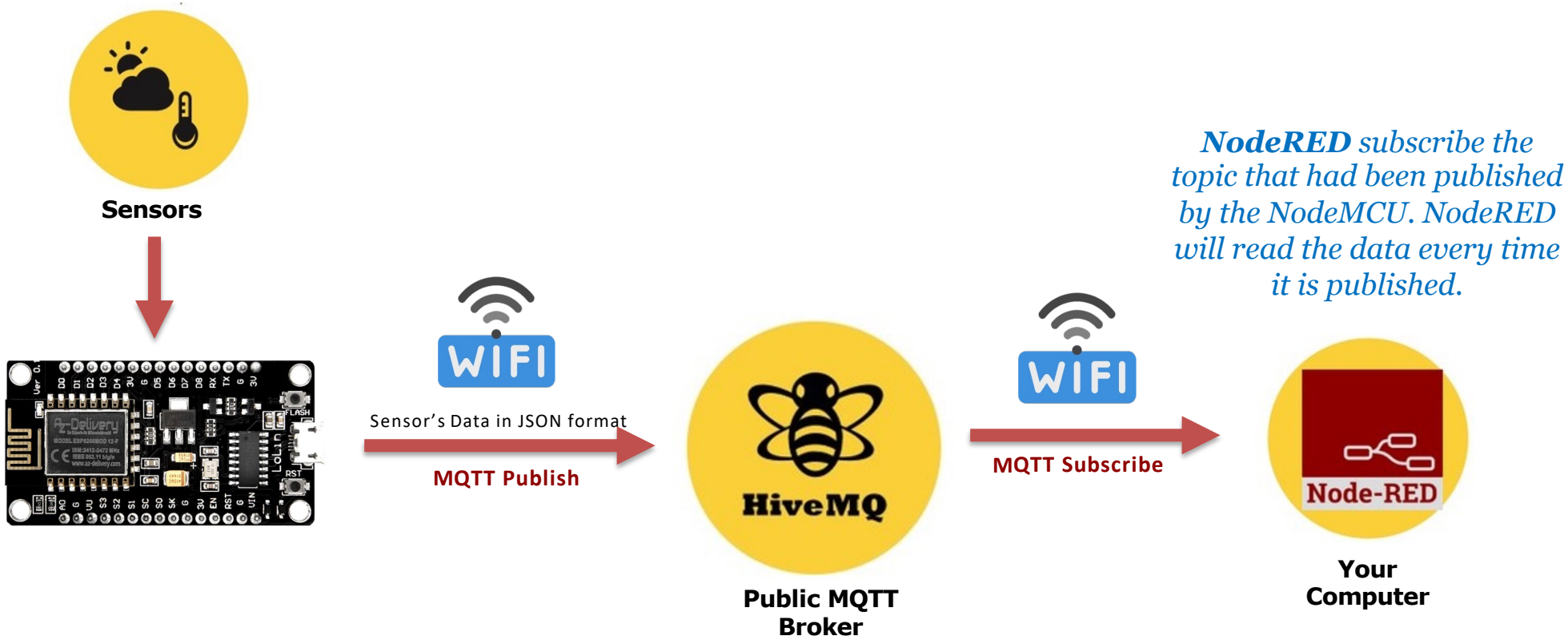




Node-RED subscribe MQTT from Broker

The connectivity of NodeMCU and Node-RED is via HiveMQ public MQTT broker.

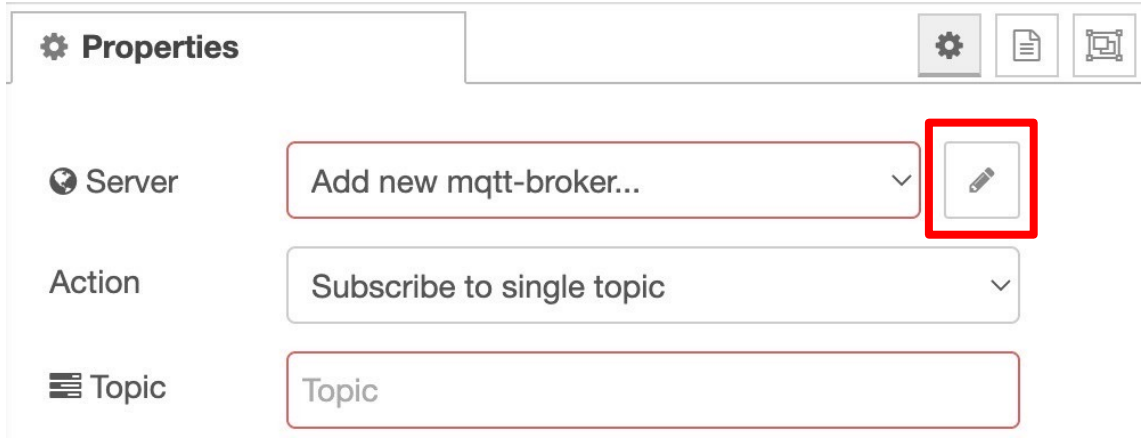
The architecture:



Node-RED subscribe MQTT from Broker

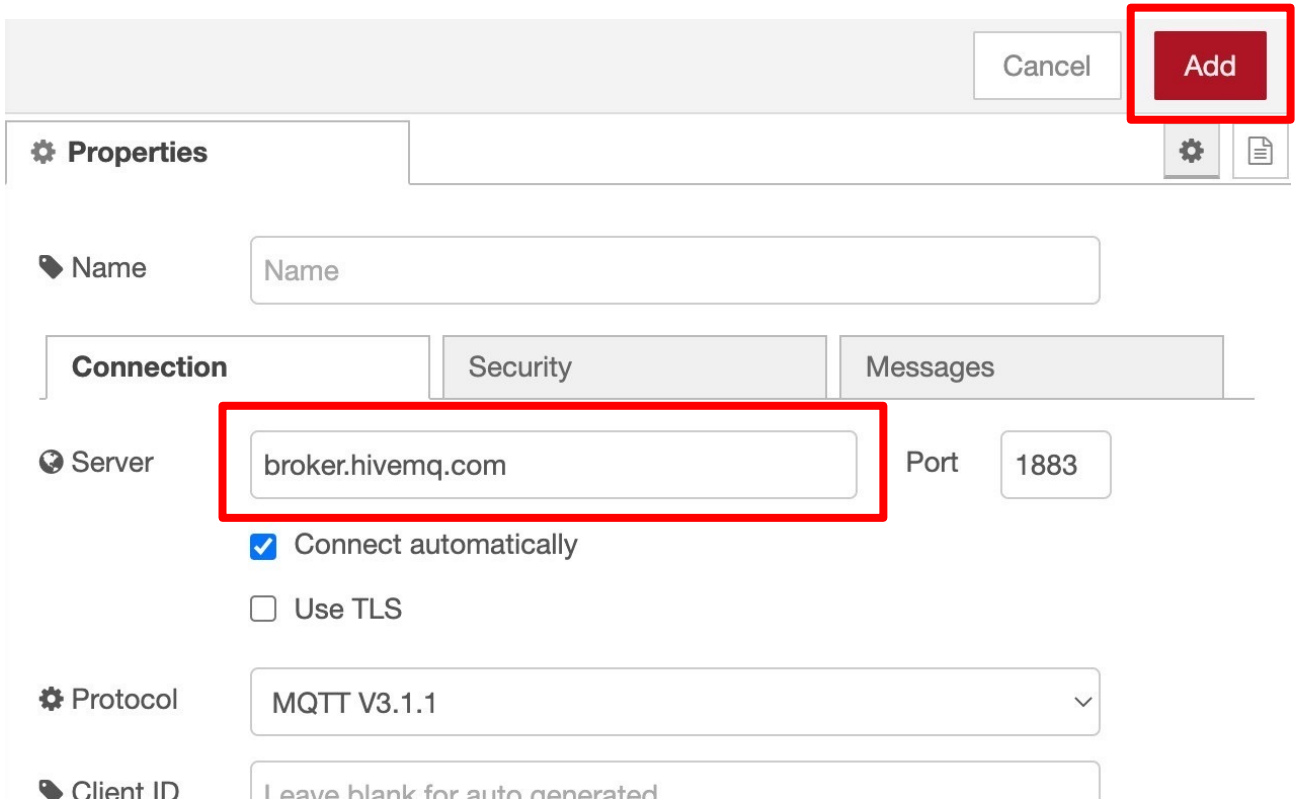
Node-RED Receive JSON Data from NodeMCU (10 Steps)

1. Insert **mqtt in** node into the workspace.
2. Double click the node to edit the properties.
3. Add new MQTT broker setup by clicking the pencil icon.



Node-RED subscribe MQTT from Broker

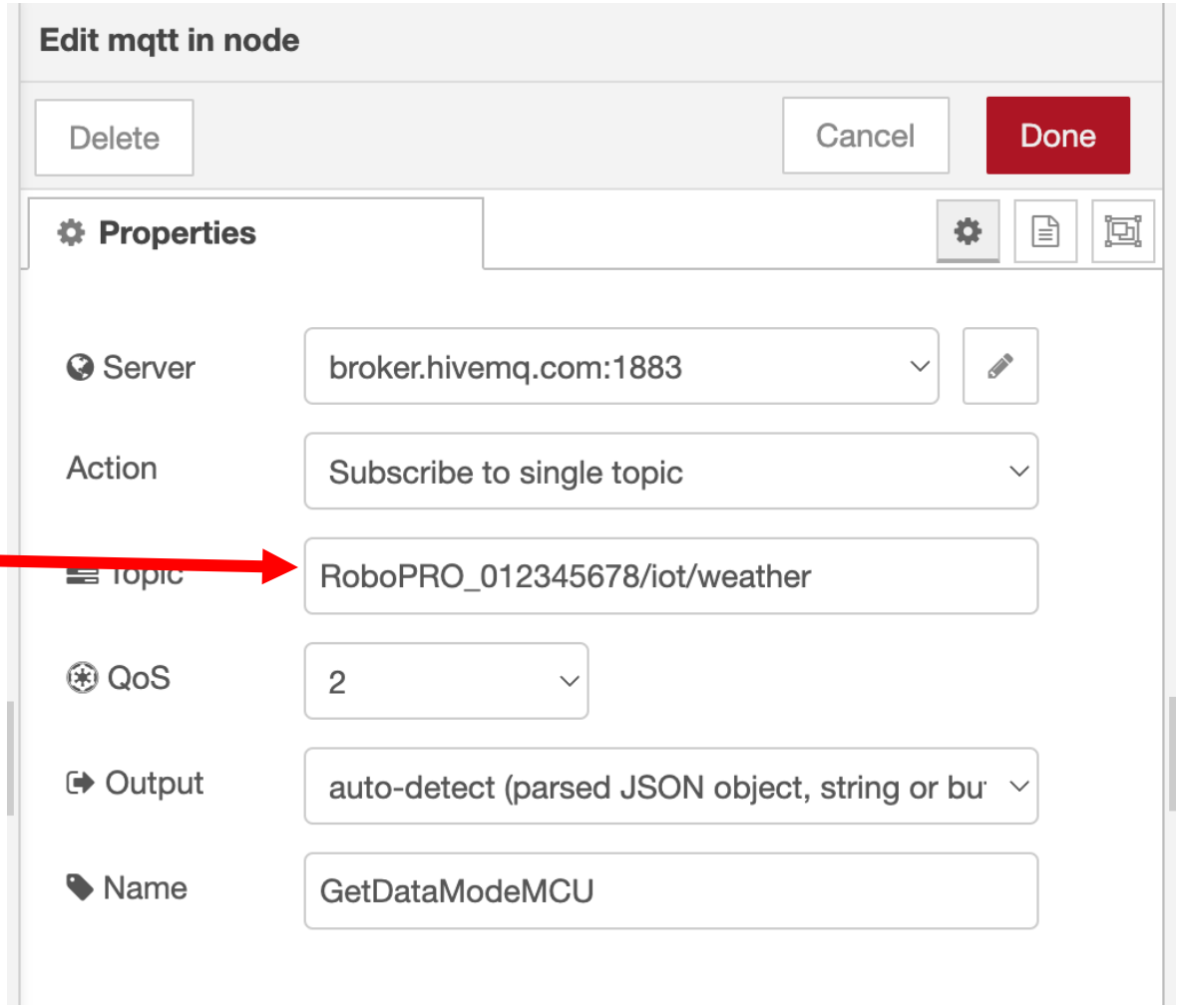
4. Type in **broker.hivemq.com** to the **Server** field.
5. Left others configuration as default.
6. Click the **Add** button to confirm adding the MQTT broker.



Node-RED subscribe MQTT from Broker

4. Set MQTT topic to provided MQTT topic by NodeMCU, from the Serial Monitor.

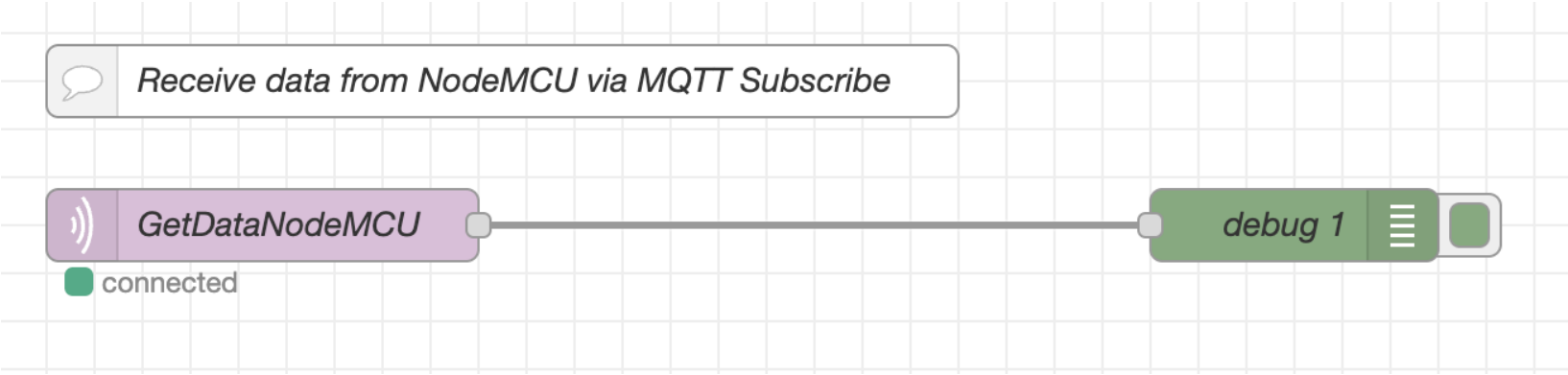
```
Temperature = 27.07 °C
LED1 = 0
Data to Publish: {"temperature":27.07,"led1":0}
Length of Data: 30
Publish to: RoboPRO_012345678/iot/weather
```

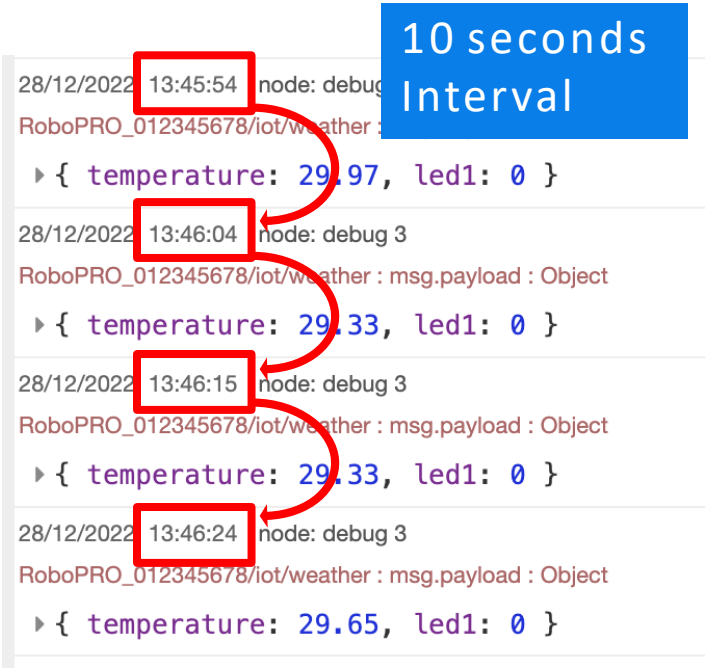


5. The Name of the node is **GetDataNodeMCU**.
6. Left other configurations as default.
7. Click the **Done** button to confirm the **mqtt in** node properties configuration.

Node-RED subscribe MQTT from Broker

8. Insert **debug** node into the workspace and connect with **mqtt in** node.



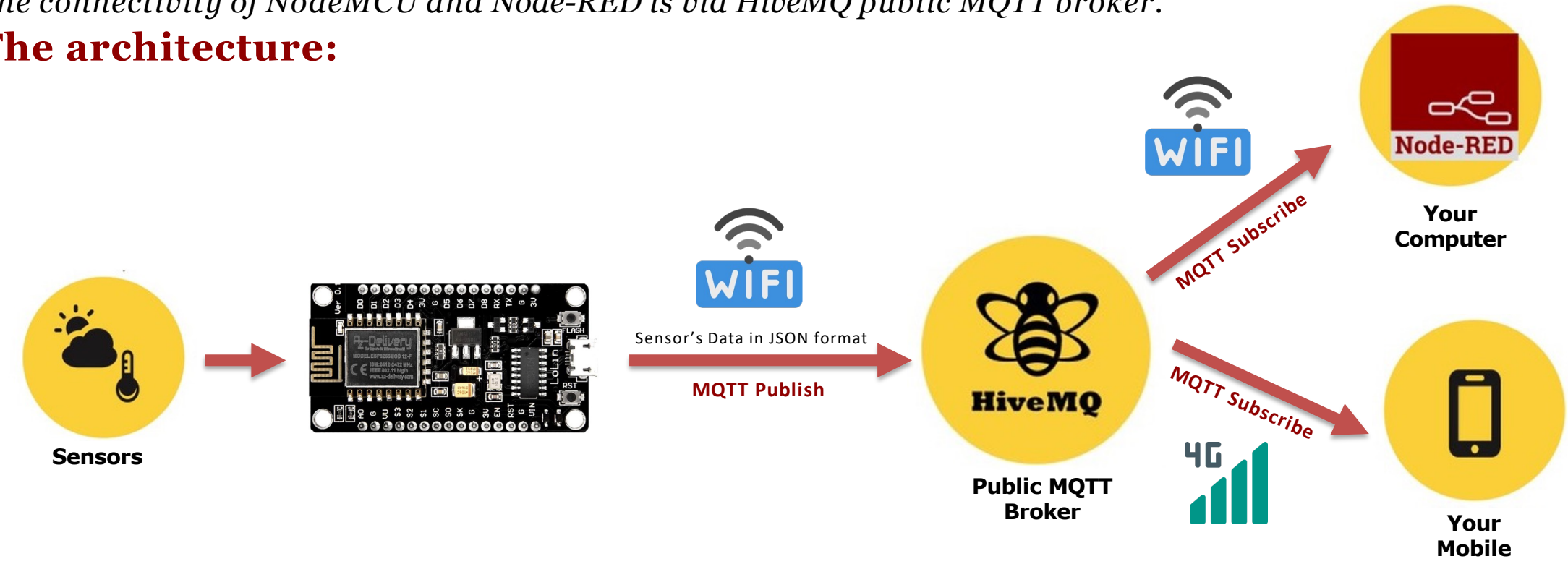


9. Click the **Deploy** button to redeploy the flow.
10. Observe the incoming JSON payload on the Node-RED's debug sidebar from the MQTT topic published by NodeMCU, where the time interval as it is **10 seconds**.

Mobile Apps subscribe MQTT from Broker

The connectivity of NodeMCU and Node-RED is via HiveMQ public MQTT broker.

The architecture:

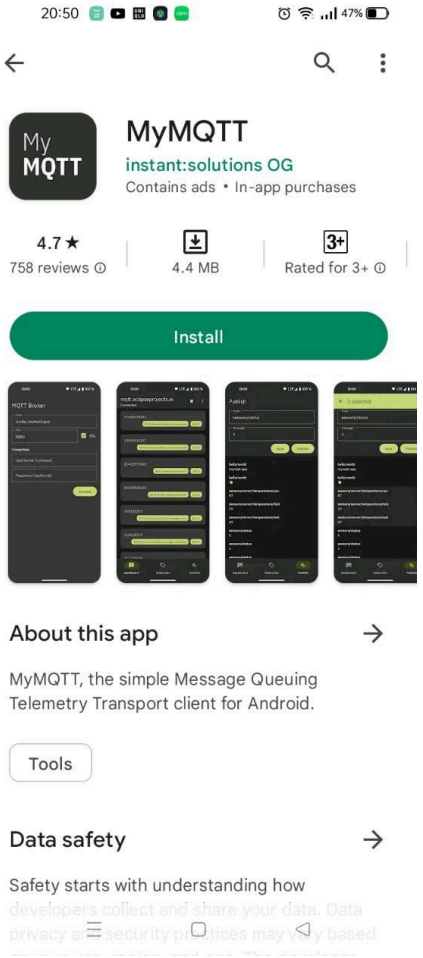


Mobile Apps *subscribe the topic that had been published by the NodeMCU. The apps will read the data every time it is published.*

Mobile Apps subscribe MQTT from Broker

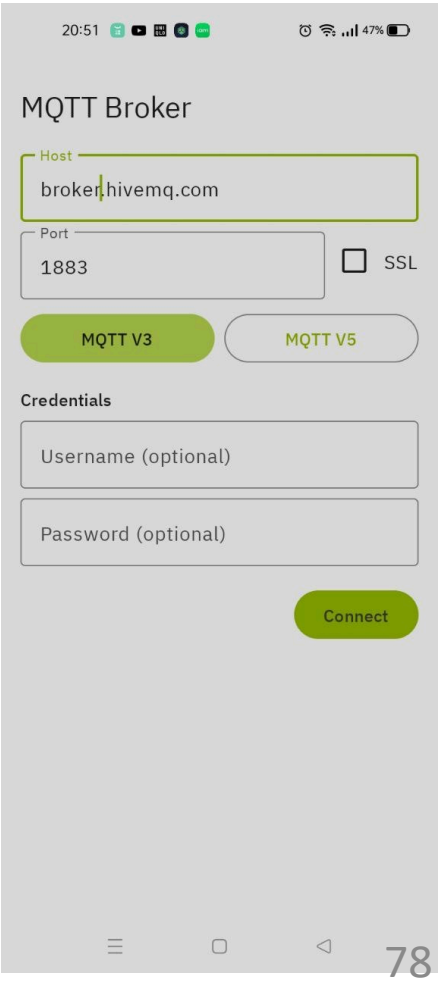
Setup MQTT Client Apps on Mobile Phone (6 steps)

1. Download **MyMQTT** Apps from Google Play or Apple Store



2. Setup the Broker connection:
   - **Host**: broker.hivemq.com
   - **Port**: 1883
   - **Protocol**: MQTT V3

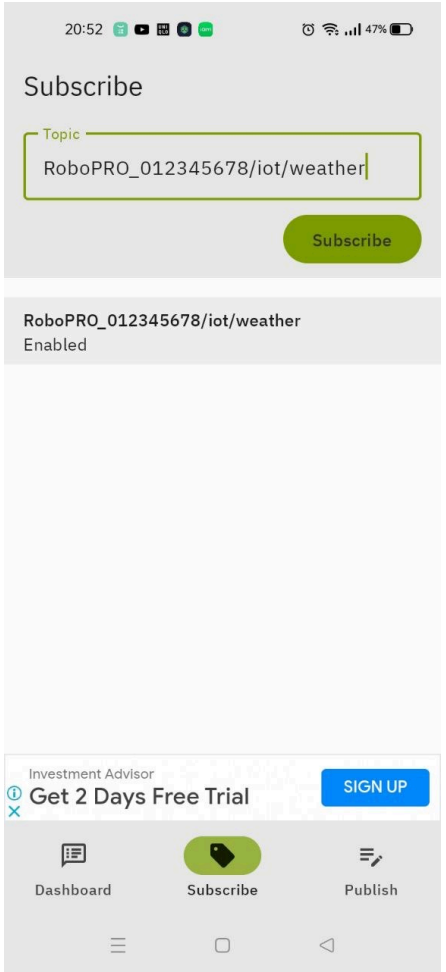
3. Click **Connect** to connect to the broker server



Mobile Apps subscribe MQTT from Broker

4. Enter your **subscribe topic** as stated in your Arduino Sketch.

5. Click **Subscribe**.

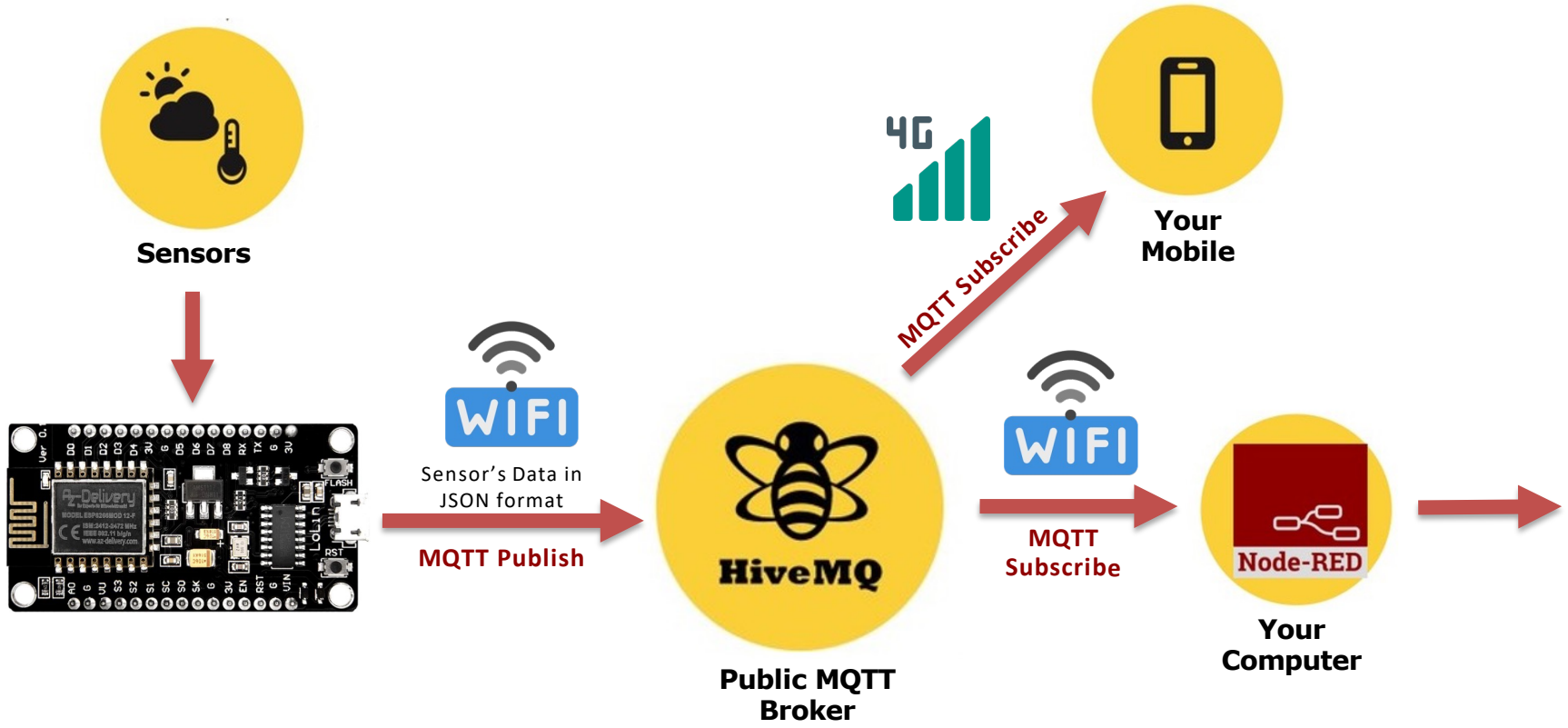


6. Click **Dashboard**. You should receive sensors data every time it is published by the NodeMCU.

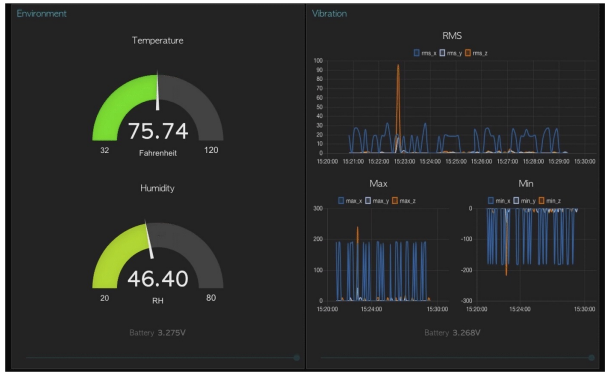
Data Visualisation through NodeRED Dashboard

The connectivity of NodeMCU and Node-RED is via HiveMQ public MQTT broker.

The architecture:



***Data visualization** by Node-RED will view sensors data in real time according to user design*



Node-RED Dashboard

Integrate the JSON Payload Data to the Node-RED Dashboard Nodes.

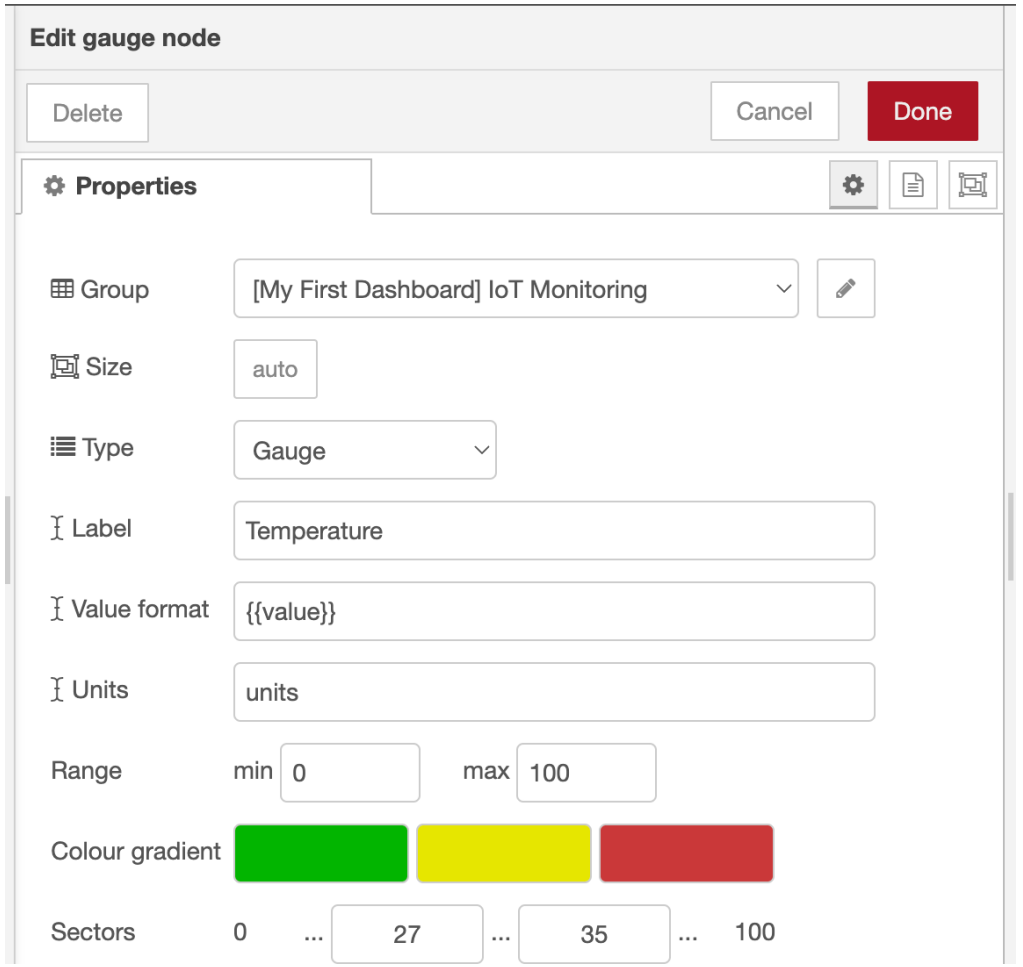
- Below is the example of incoming JSON payload into Node-RED **mqtt in** node.

```
{
  "temperature":29.33,
  "led1":0
}
```

- It is an object, with 2 JSON keys, **temperature** and **led1**.
- In Node-RED the right way to get the value of **temperature** is to called the variable with complete message object, for example **msg.payload.temperature** will return 28.17.
- While **msg.payload.led1** will return 0

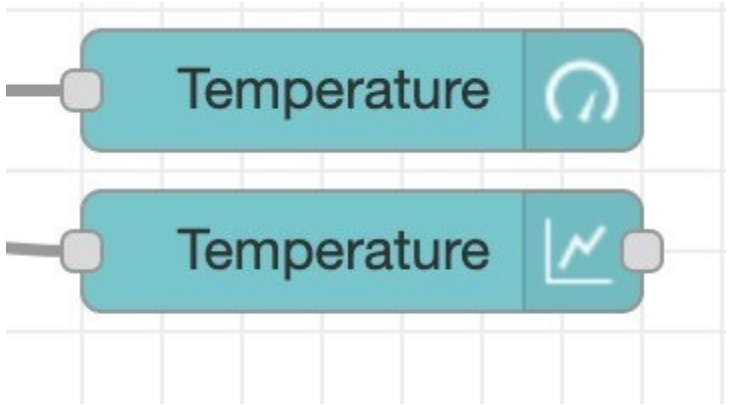
Data Visualisation through NodeRED Dashboard

- Insert **gauge** and **chart** node into the workspace.
- Double click the **gauge** node to edit the properties.
- The Label is "**Temperature**". Leave other values as default.
- Select the existing group (that you had created before) – eg. *[My First Dashboard] IoT Monitoring*
- Click Done.



Data Visualisation through NodeRED Dashboard

- Double click the **chart** node to edit the properties.
- The Label is "**Temperature**".
- Select the existing group (that you had created before) – eg. *[My First Dashboard] IoT Monitoring*
- Click Done.



- The **gauge** and **chart** node require value from **msg.payload** object **only**, therefore it is not able to read the value from **msg.payload.temperature** object.
- So, we need to set the value from **msg.payload.temperature** object to **msg.payload** object, then the node are able to read the value correctly.

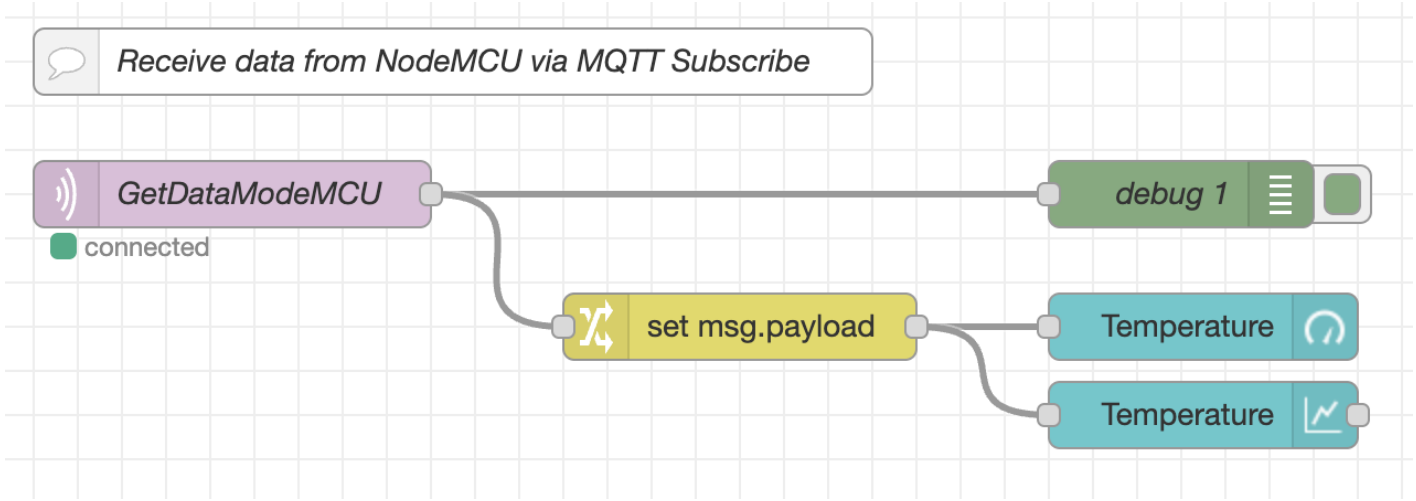
- To do so, we can use **change** node to do the job.

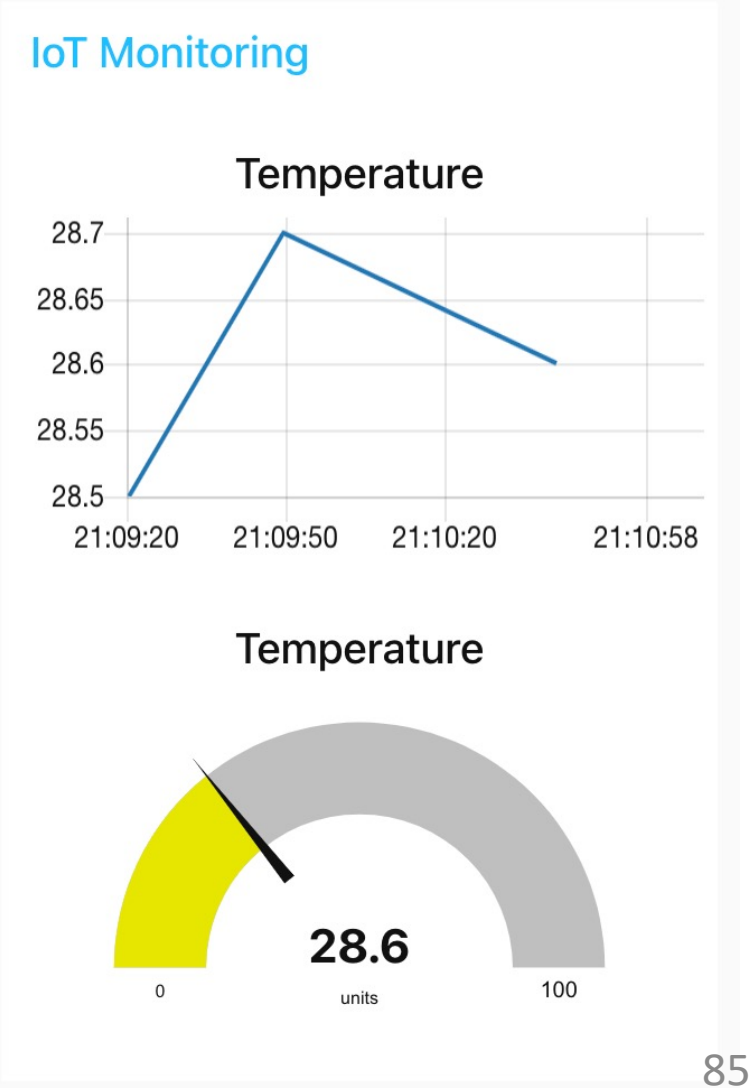
Set msg.payload Value from the msg.payload.temperature (5 Steps)

1. Insert **change** node into the workspace.
2. Double click the node to setup the conversion properties.
3. Set the **msg.payload** to the value of **msg.payload.temperature** and click the **Done** button.

Data Visualisation through NodeRED Dashboard

4. Wire them together and redeployed the flow.
5. View the dashboard to see real-time widget changing value depending on the interval of MQTT publish from the NodeMCU.

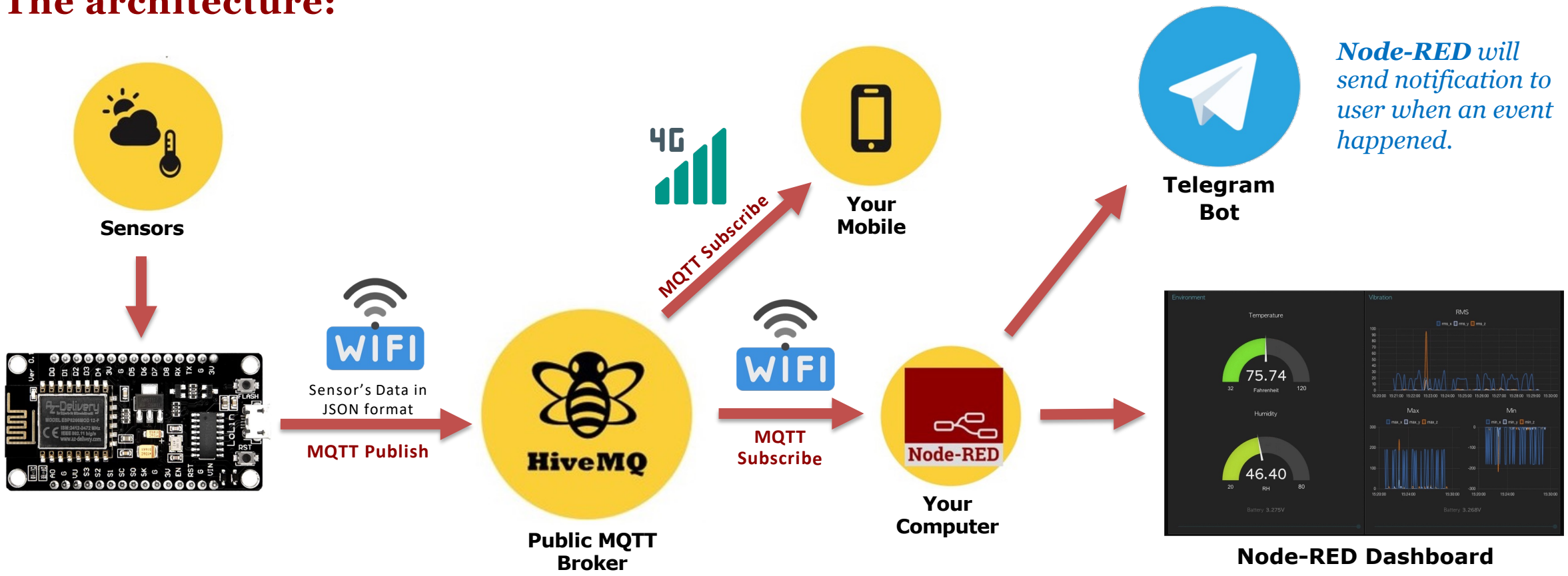




NodeRED Send Notification to Telegram

The connectivity of NodeMCU and Node-RED is via HiveMQ public MQTT broker.

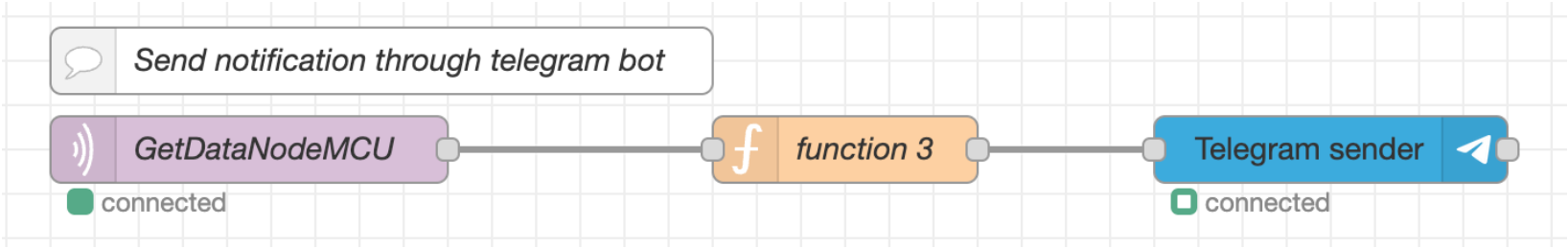
The architecture:



NodeRED Send Notification to Telegram

Real-Time Data Alert Notification to Telegram (8 Steps)

1. Copy the **mqtt in** node **GetDataNodeMCU** from the previous lesson and paste the node in the workspace .
2. Insert a **function** node and a **Telegram sender** node and wire them together.



3. Double click the **function** node to setup to the Telegram notification based on specific threshold value. For example, trigger an alert when temperature's value is above 30.

NodeRED Send Notification to Telegram

4. Create a variable named **suhu** and its value equivalent to JSON object from **mqtt in** node **msg.payload.temperature**.

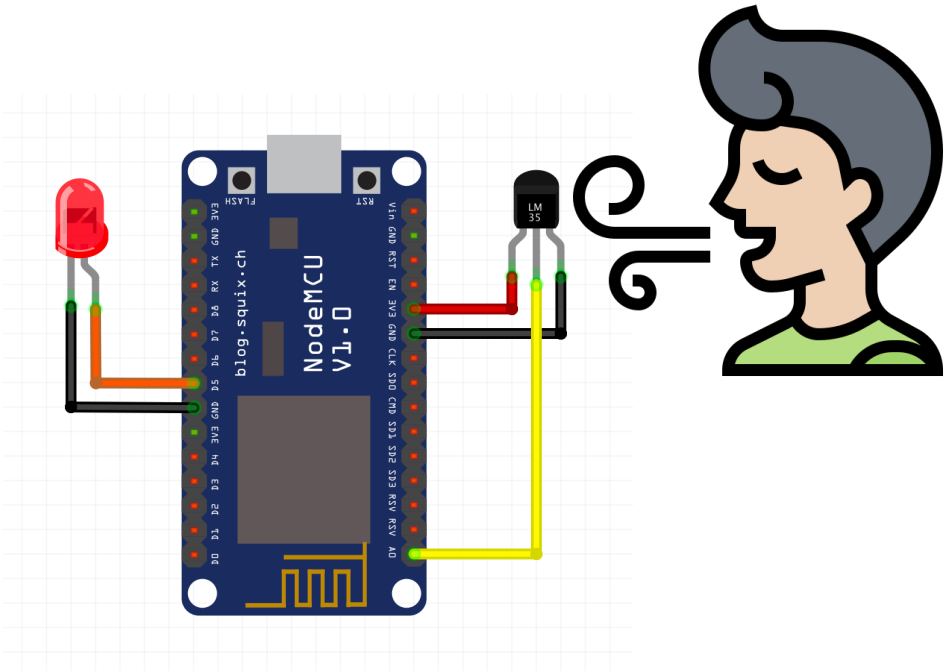
5. The previous Telegram configurations will only be executed if the suhu value is more than 30, with message of **Awas!**

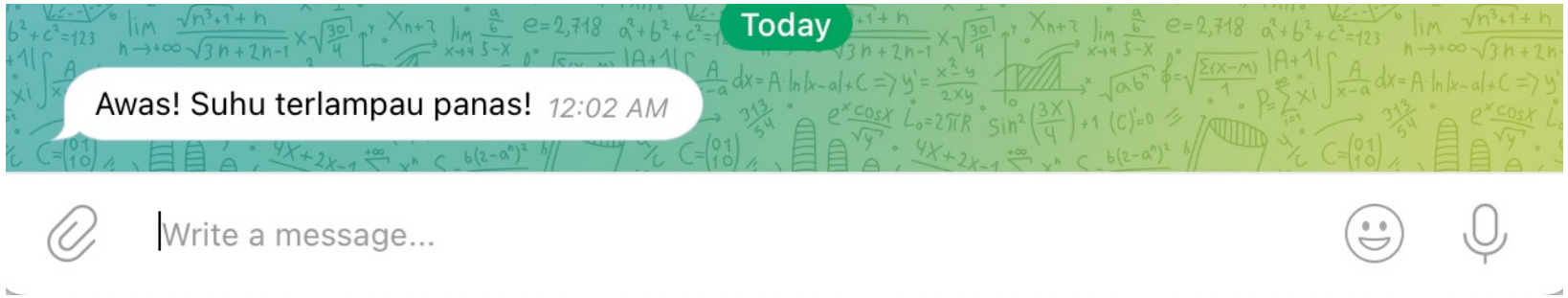
Setup | On Start | **On Message** | On Stop

```
1  var suhu = msg.payload.temperature;
2
3  if(suhu>30){
4      msg.payload = {
5          "chatId": 61722174,
6          "type": "message",
7          "content": "Awas! Suhu terlampau panas!",
8          "options": {
9              "parse_mode": "Markdown"
10         }
11     }
12 }
13
14 return msg;
15
```

NodeRED Send Notification to Telegram

6. Click the **Done** button and redeploy the flow.
7. Provide high value of temperature by using your breathe pointing on the LM35 sensor on the NodeMCU.
8. Check Telegram alert notification receive when the temperature's value above 30.

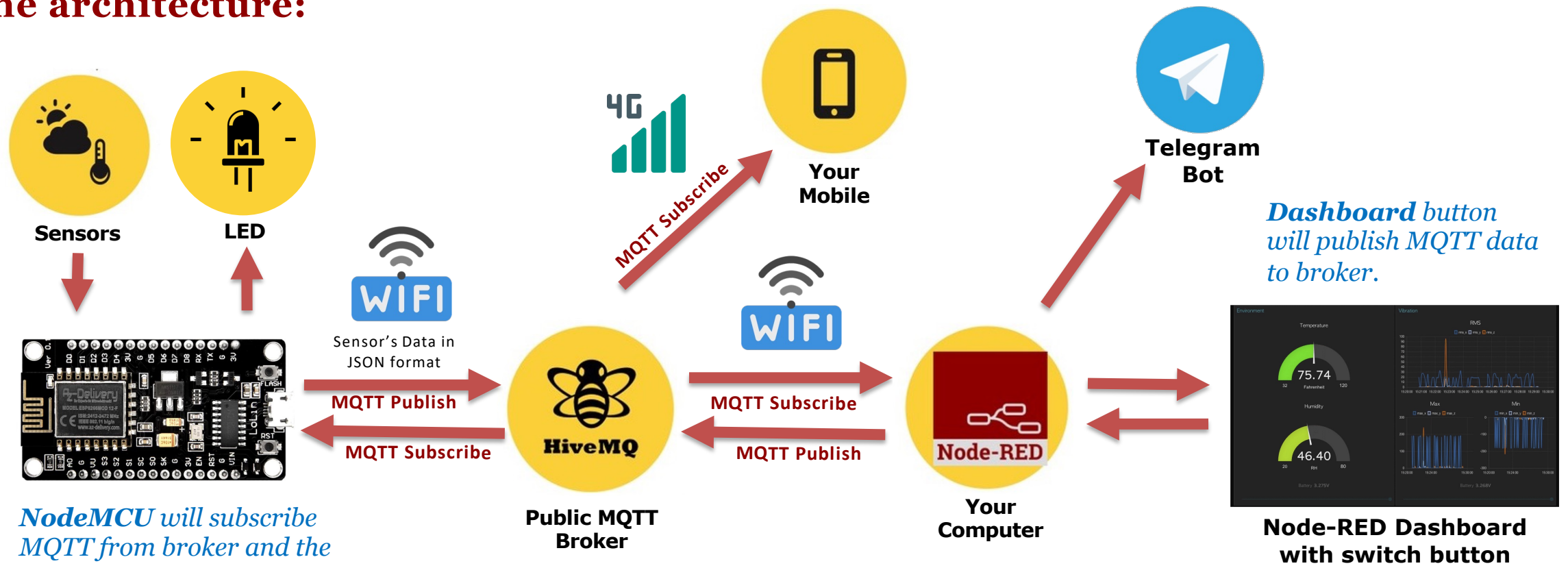




Controlling NodeMCU from Dashboard

The connectivity of NodeMCU and Node-RED is via HiveMQ public MQTT broker.

The architecture:



***NodeMCU** will subscribe MQTT from broker and the LED will behave accordingly every time **dashboard** publish data.*

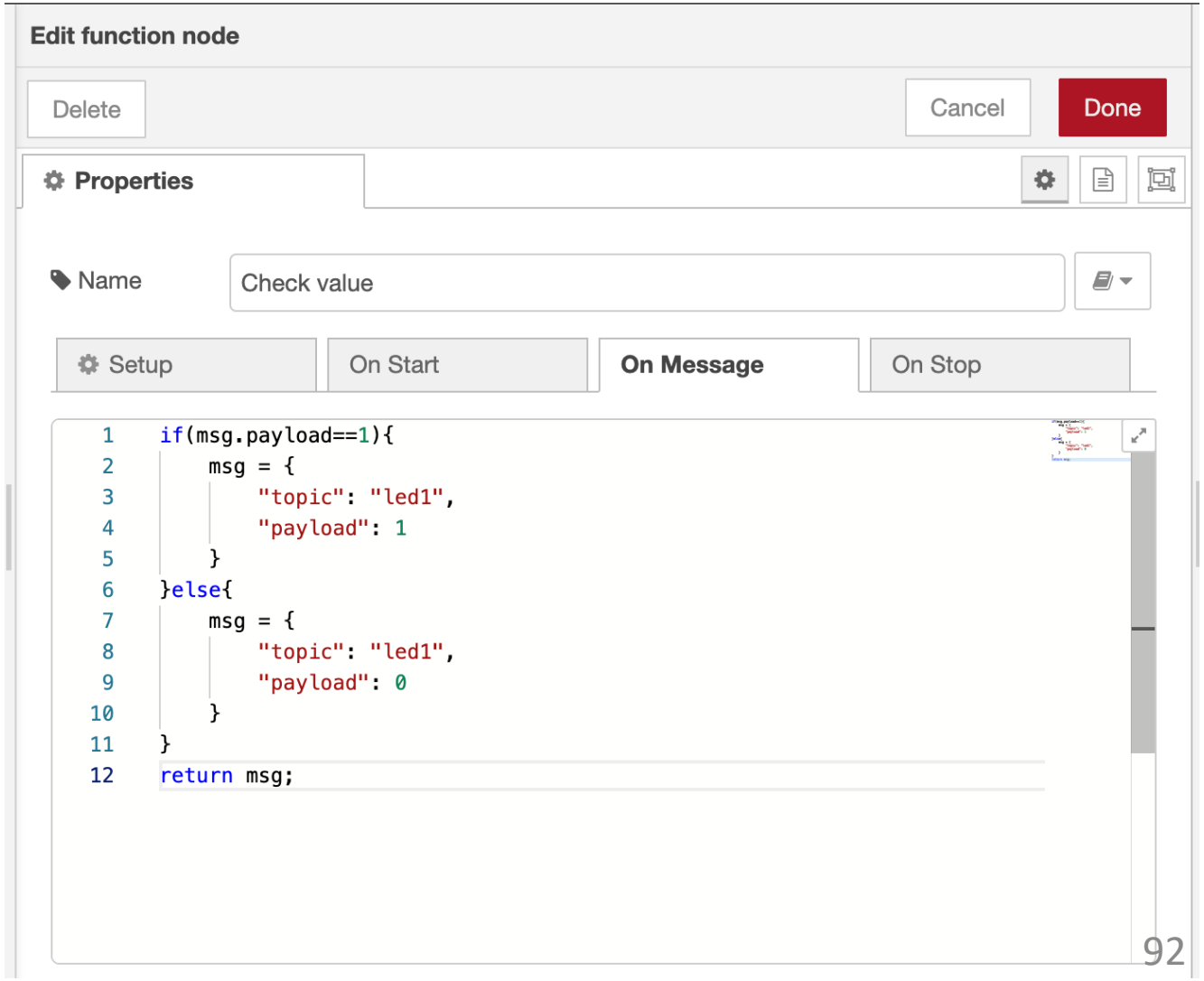
Controlling NodeMCU from Dashboard

Integrate Switch/Button to the Node-RED Dashboard Nodes.

- Insert Dashboard **Switch**, **function**, and **mqtt out** nodes.
- Double click the **switch** node to edit the properties.
- The Label is "**LED 1**".
- Select the existing group (that you had created before) – eg. *[My First Dashboard] IoT Monitoring*
- Click Done.

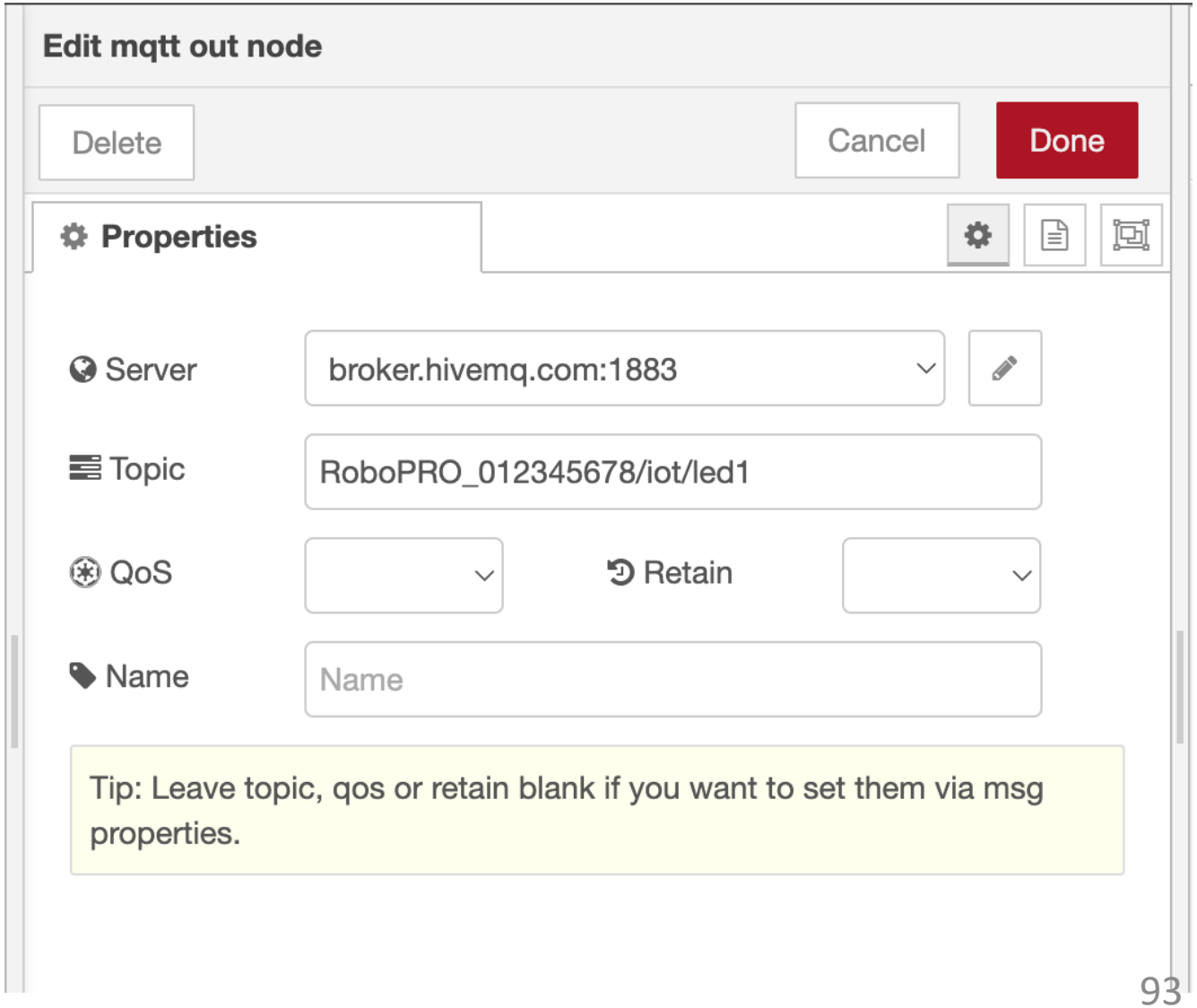
Controlling NodeMCU from Dashboard

- Double click the **function** node to edit the properties.
- Insert the given code in the **On Message** tab.
- Click Done.

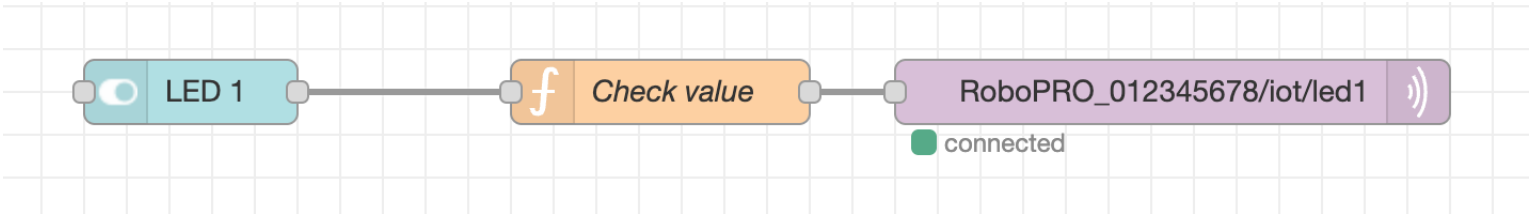


Controlling NodeMCU from Dashboard

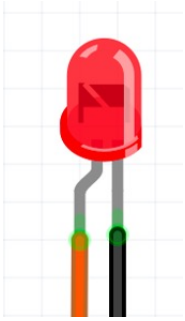
- Double click the **mqtt out** node to edit the properties.
- Select the existing Server (broker.hivemq.com:1883).
- Insert your LED1 topic, eg.: **RoboPRO_012345678/iot/led1**
- Click **Done**.

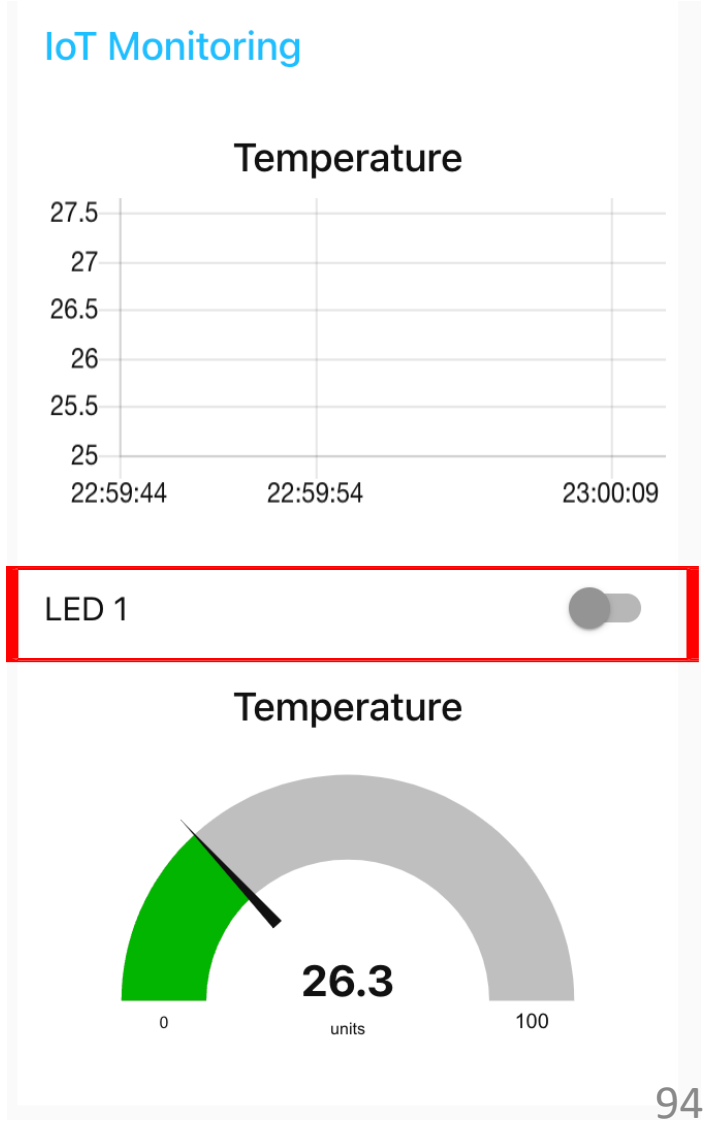


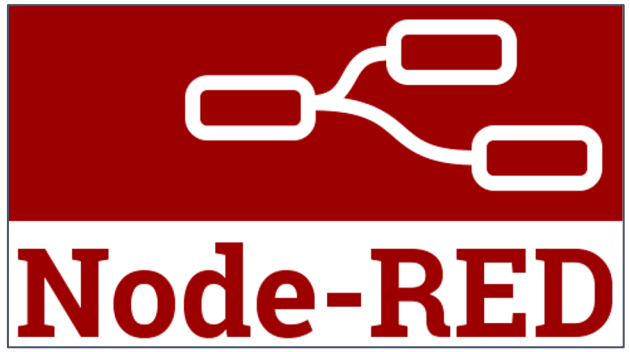
Controlling NodeMCU from Dashboard



- Connect the nodes.
- Open your dashboard, you will see a switch button labelled "LED 1" will appear.
- Click the **switch** on/off, it will turn your LED on NodeMCU on/off accordingly.
- Check your LED!







Node-RED with MySQL

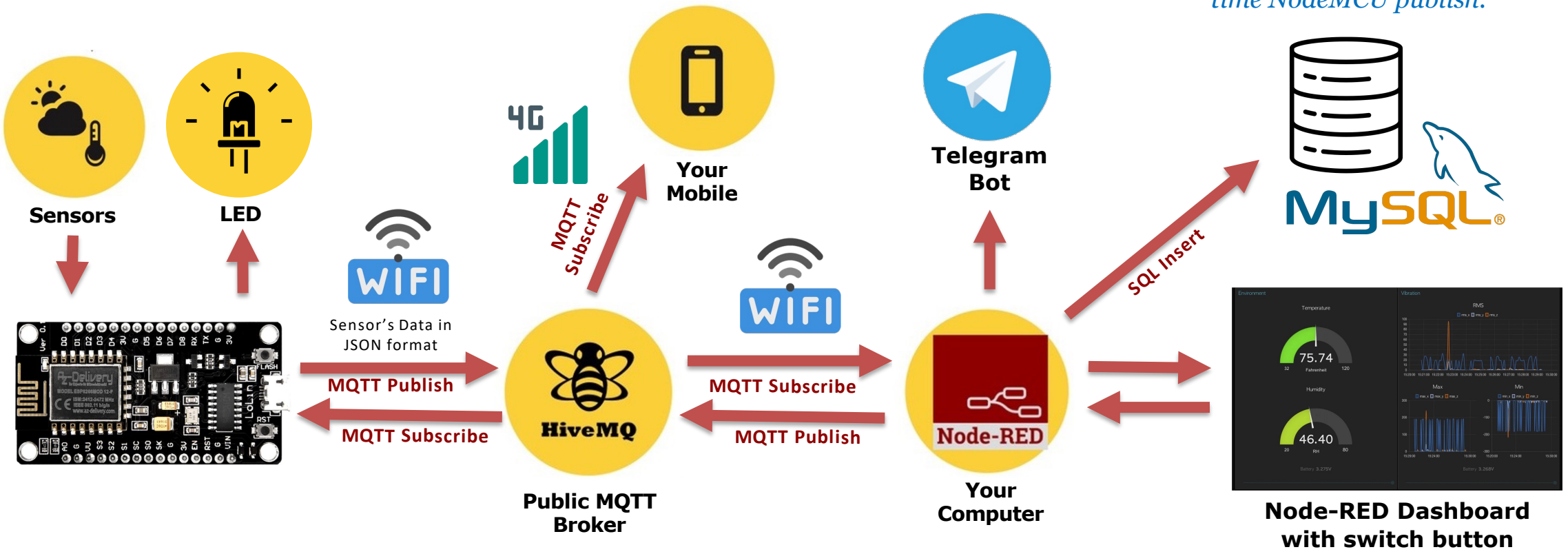
Using Node-RED to read and write data to a MySQL Database.

Saving Data into MySQL via Node-RED

The connectivity of NodeMCU and Node-RED is via HiveMQ public MQTT broker.

The architecture:

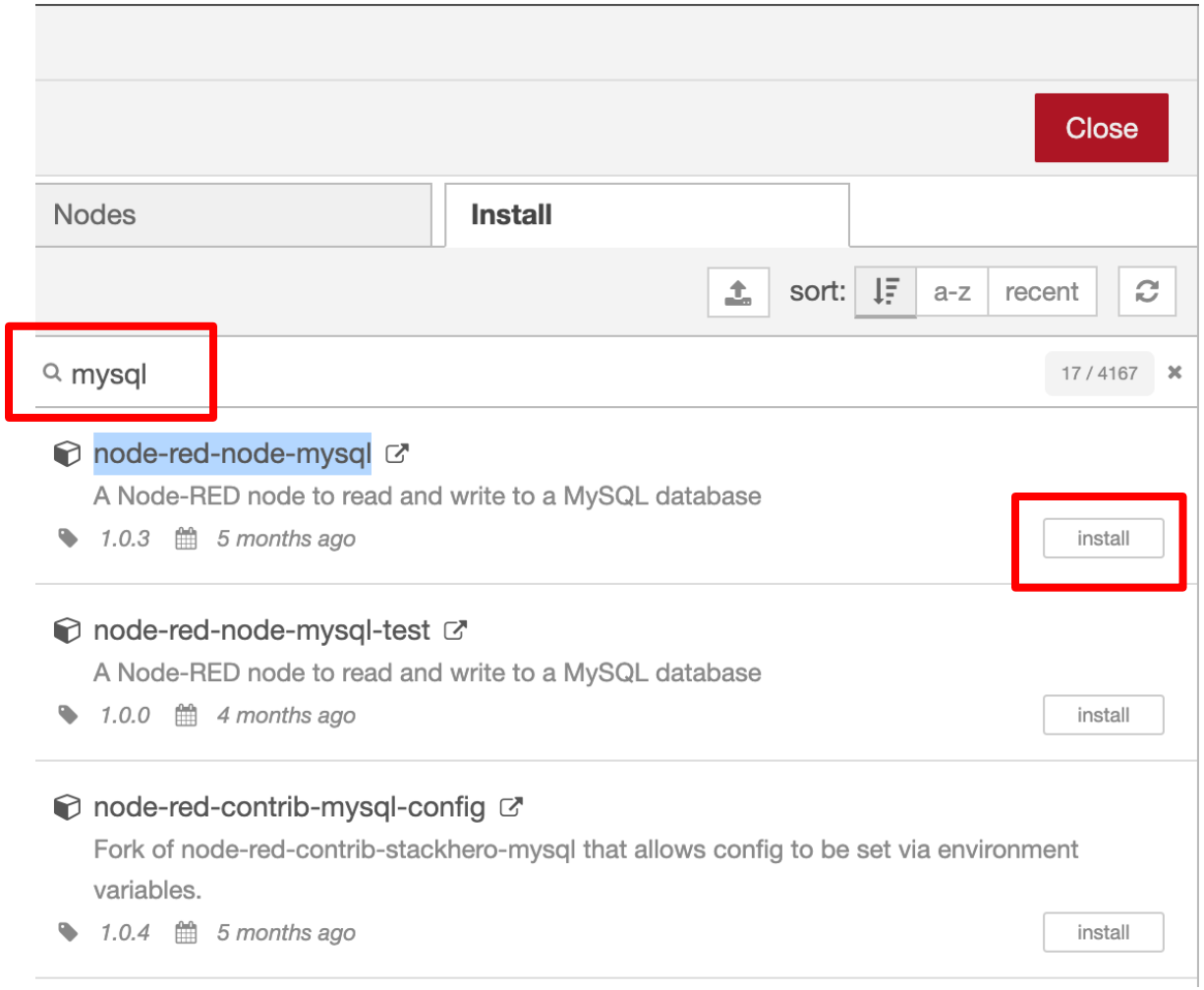
***Node-RED** will insert data into MySQL Database every time NodeMCU publish.*



Saving Data into MySQL via Node-RED

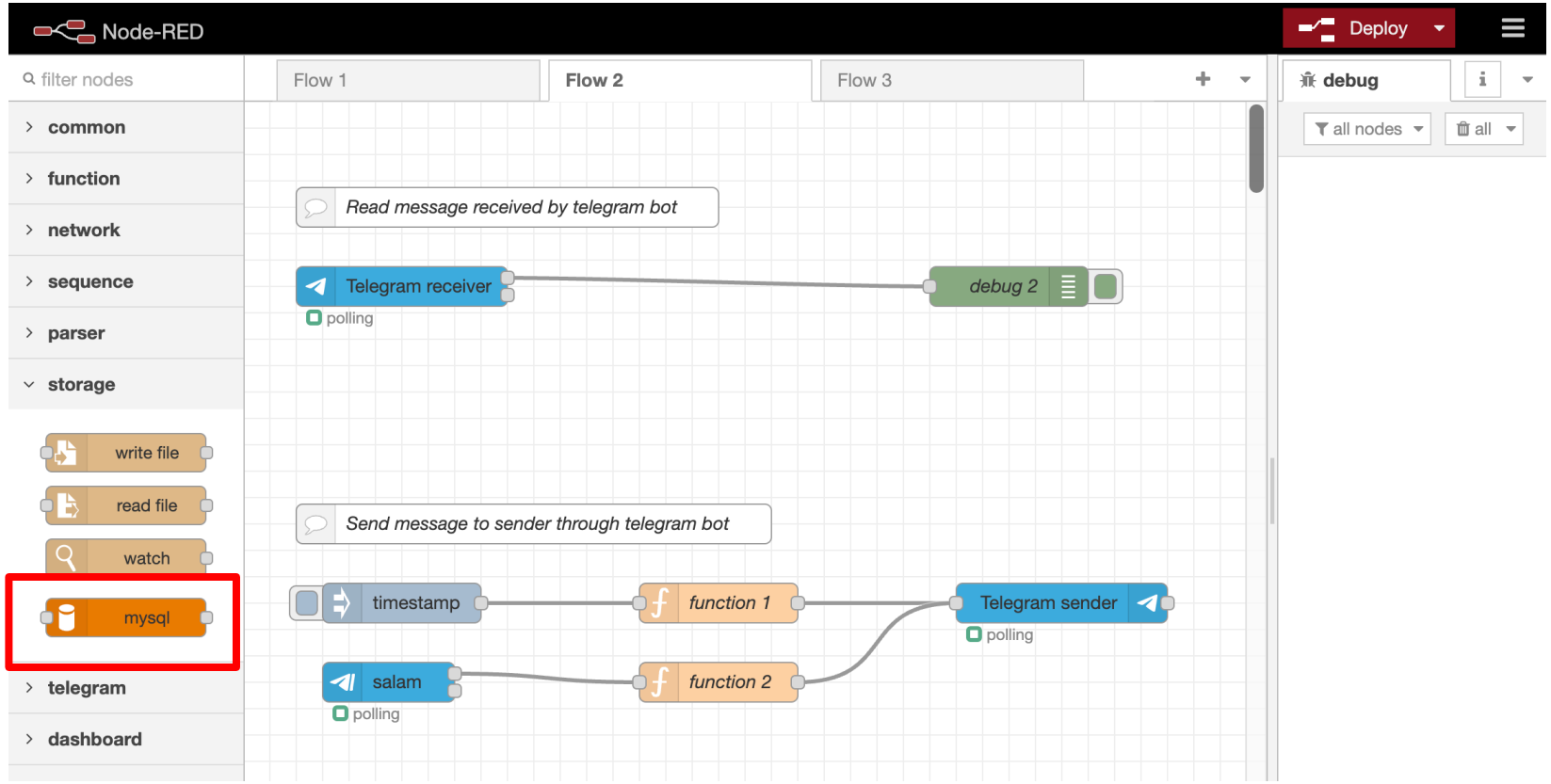
Install Node-RED Dashboard

- Open Node-RED Palette Manager.
- Click on the **Install** tab and on the **search modules** fields type "**mysql**" to filter the list of the available modules.
- Click the **install** button to install the **node-red-node-mysql** module and wait until the installation process is successful.



Saving Data into MySQL via Node-RED

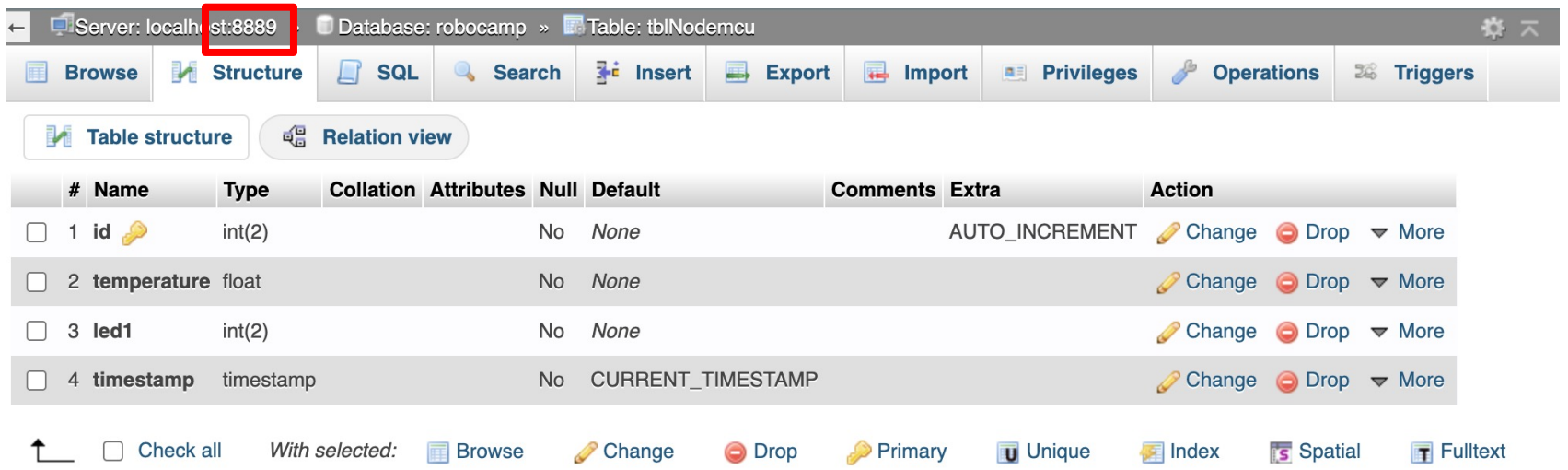
- Once the installation is successful, the mysql node will be available on the palette under an existing category: **storage**.



Saving Data into MySQL via Node-RED

Node-RED write data into MySQL table (10 Steps)

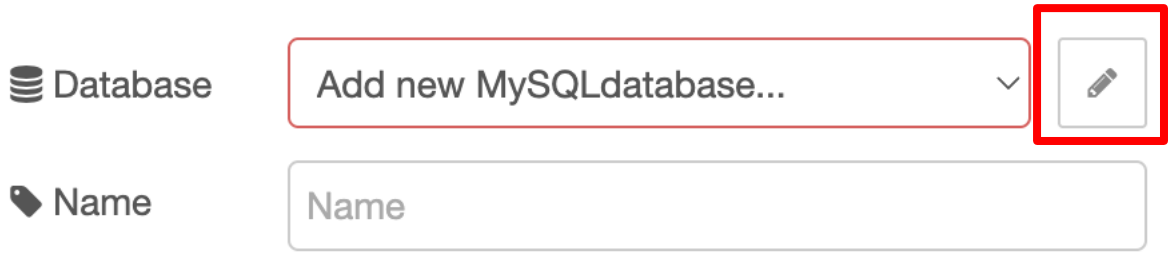
1. Make sure you already have MySQL or MariaDB installed on your localhost.
2. Create a new database name "**robocamp**" and a new table name "**tblNodemcu**" with the same structure as below:



3. Keep note on the **port number** of your database. You are going to use it later.

Saving Data into MySQL via Node-RED

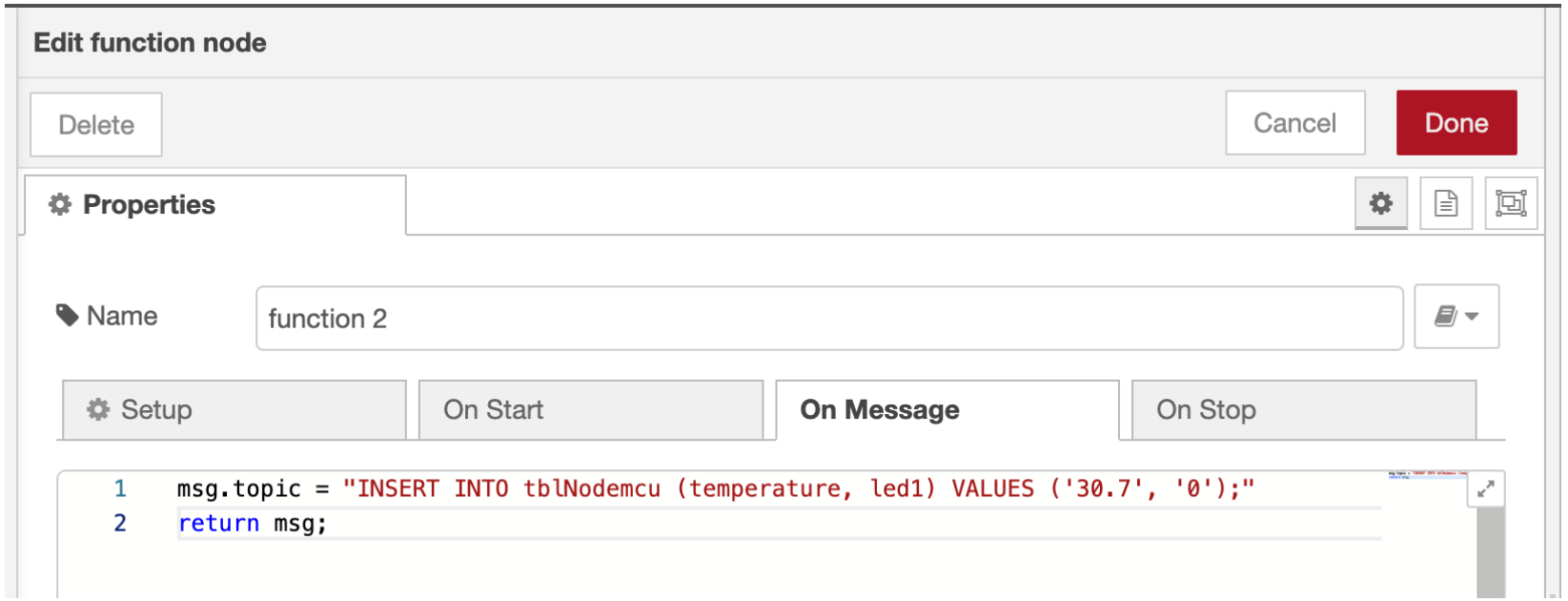
4. Insert **inject**, **function**, **mysql** and **debug** nodes into the workspace.
5. Double click **mysql** node and add *a new MySQL database* by clicking the pencil icon.



6. Enter the following details:
   - **Host**: 127.0.0.1
   - **Port**: enter the port number from step 3.
   - **User**: enter your MySQL username
   - **Password**: enter your MySQL password
   - **Database**: robocamp
7. Leave other values as default. Click **Add** when finished.

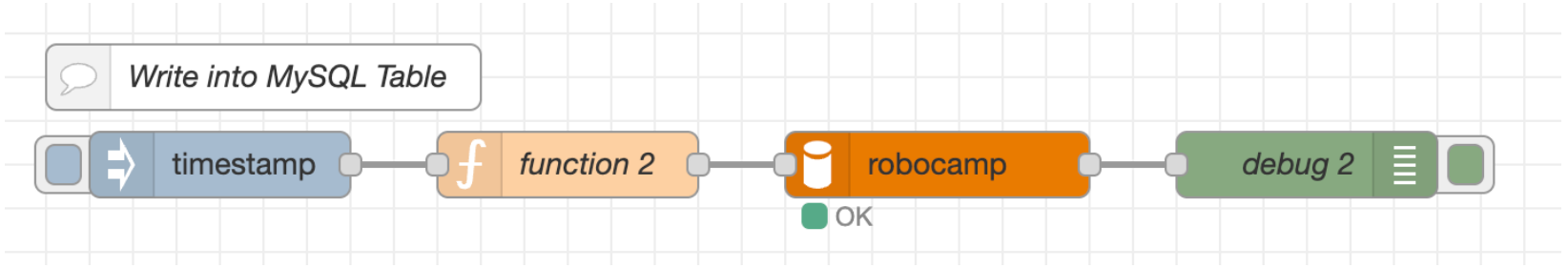
Saving Data into MySQL via Node-RED

8. Double click the **function** node to edit the properties. Insert the given code in the On Message tab. 30.7 and 0 are just dummy value for temperature and LED1. You can use any value that you want.



9. Click **Done**.

Saving Data into MySQL via Node-RED



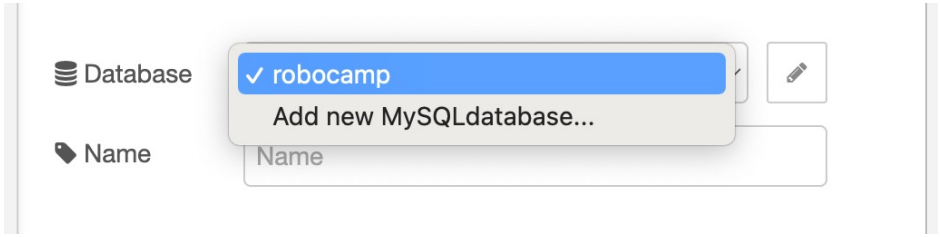
10. Connect all the nodes. Click **Deploy**. Check your database, a new value should had been inserted in the **tblNodemcu** table.

Note:
This is how you can store your sensor data into a database. You can also read date from your table by changing the SQL statement in the **function** node.

Saving Data into MySQL via Node-RED

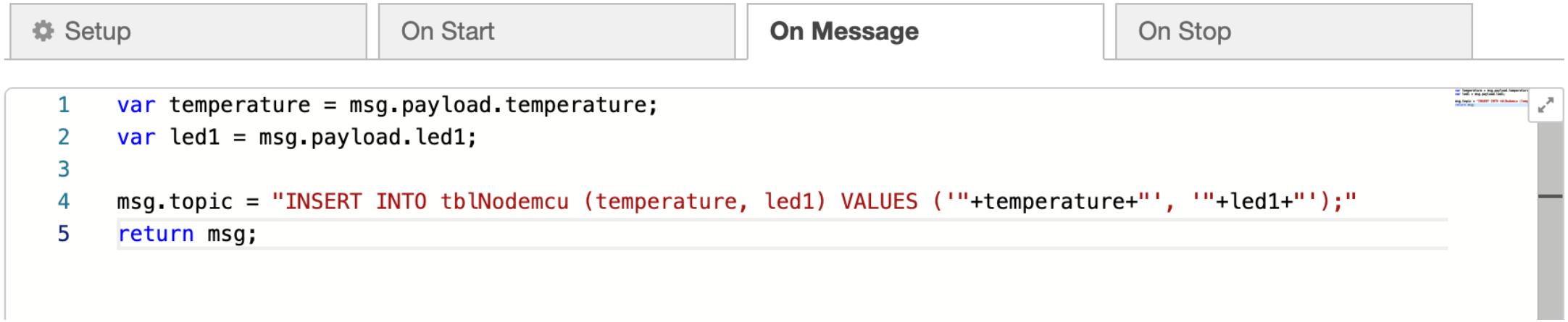
Node-RED get MQTT data and write into MySQL table (4 Steps)

1. Insert **function** and **mysql** nodes into the workspace.
2. Double click **mysql** node and select the existing MySQL database *robocamp*.



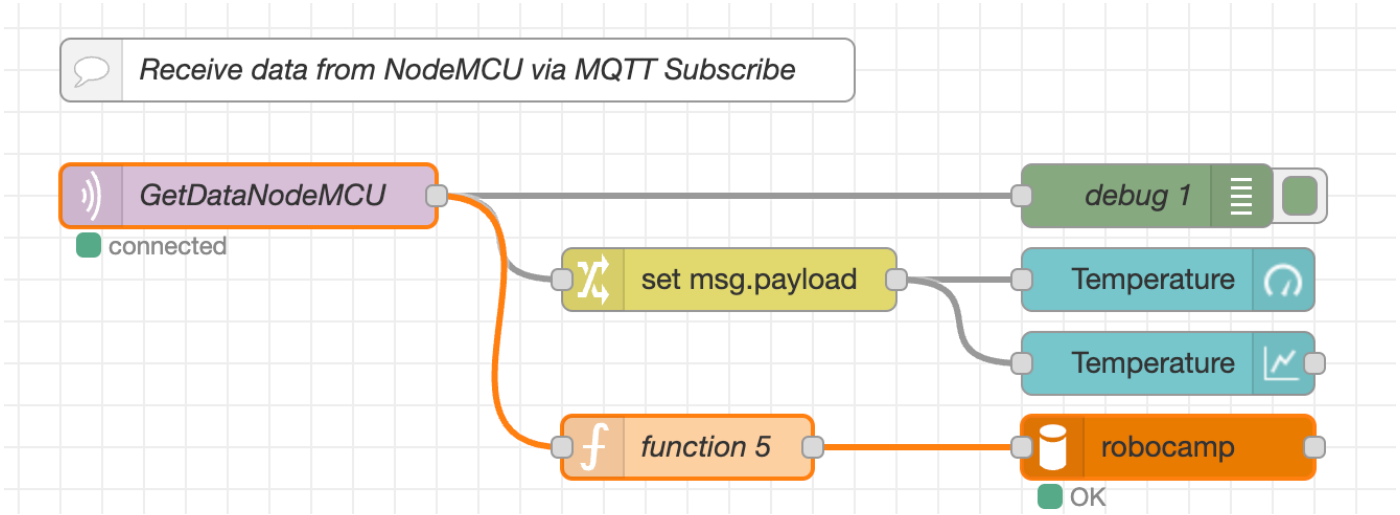
Saving Data into MySQL via Node-RED

3. Double click the **function** node to edit the properties. Insert the given code in the On Message tab. Click **Done**.



Saving Data into MySQL via Node-RED

4. Connect the function node from the existing **mqtt in** node *GetDataNodeMCU*, and to the new **mysql** node.



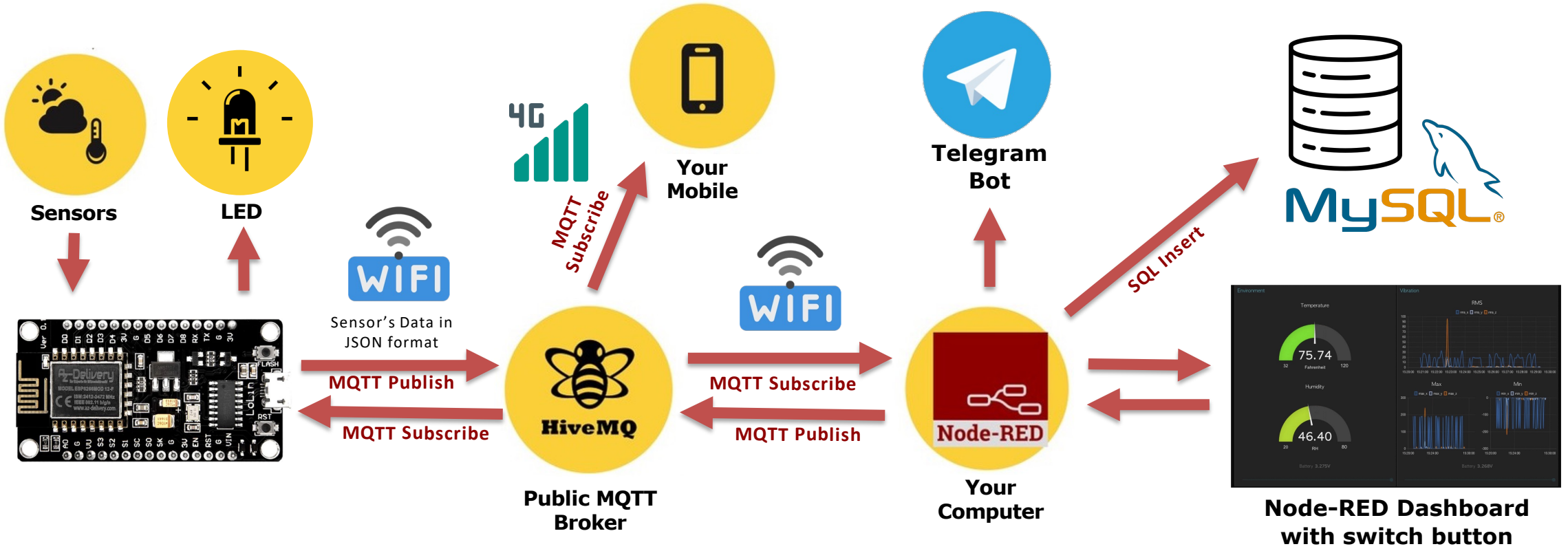
Note:

Check your database. A new entry should be inserted every time *GetDataNodeMCU* read new data from your sensors. You can develop a mobile or web application to read the data from the MySQL database and visualize them in a graph/figures, or perhaps produce a meaningful report.

Wrapping Up!

What have we learned today?

The final architecture:



Got it? You can do so much from here. Be creative!

That's it!

There is only so much I can teach.

Knowledge is a great treasure that has no limits.

The only limitation is your imagination.

Thank you.

- Cikgu Fadzli -