Techfest™
INDIAN INSTITUTE OF TECHNOLOGY, BOMBAY

VYORIUS

Internet of things PROJECTS/INTERNSHIP

College Name

→SRM Institute of Science and Technology.

Company Name

→Vyorius.

Domain

→Internet of things.

Project topics

1. IoT based on smart Agriculture System.
2. World map covid-19 dashboard.
3. Obstacle Avoidance Robot using Ultrasonic Sensor.
4. RGB Pattern.

1.IoT based on smart Agriculture System.

Objective

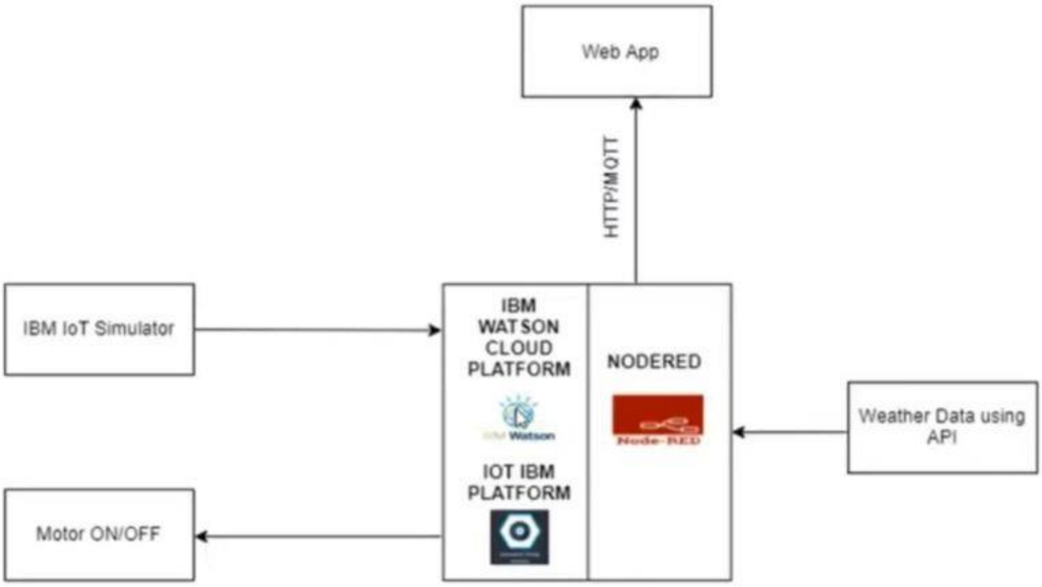
- To update farmers with the new technology and to avoid manual labour.
- To measures accurate soil moistures, temperature, humidity value from sensor.

- To meet the difficulties such as severe weather conditions and advancing climate change, and environmental consequences resulting from intensive farming practices.

Excelled efficiency Monitor their product and conditions in real-time. They get insights fast, can predict issues before they happen and make informed decisions on how to avoid them. Reduced resources

Plenty of loT solutions are focused on optimizing the use of resources-water, energy, land. Improved product Quality Data-driven agriculture helps both grow more and better products.

Block-diagram



Node red is an open-source software, flow base tools to use the wiring of the pallet, libraries, in this we can connect the hardware and the software, it is helping us to connect with everything, the hardware, the software, everything basically which deals with hardware.

IBM Watson cloud platform, cloud platform taking the input from the sensor, on the left Ibm Iot simulator, what is happening is that we have node red to make the flow, a cloud platform to store, but we need sensor for that we are using IBM iot simulator, it will act as a sensor it will simulate, to help to see whether our project working perfectly or not.

Weather data suing API, looking at how we can take a data using API, if we want the temperature, wind speed, cloud, we can have access to all those data, we have everything in the internet, google it is, why don’t we make use of it or any such browsing software to provide information with.

Whatever thing is developing we are making one web application, as in a dashboard format to visualise. Motor on and off, where we use python programming which is going to control on and off the motor movement in the dashboard at the web page.

Software's In detail

Node-RED

→Node.js@ is a JavaScript runtime built on Node-RED is a flow-based development tool for visual programming developed originally by IBM for wiring together hardware devices, APIs and online services as part of the Internet of Things. Chrome's V8 JavaScript engine.

IBM Cloud

→IBM Cloud® offers the most open and secure public cloud for business, a next-generation hybrid multi-cloud platform, advanced data and AI capabilities, and deep enterprise expertise across 20 industries.

→IBM Watson Iot device simulator

→**Open weather API**

Steps to follow

1-Download node js

2-Run with commands in the command prompt

3-Install node-red

4-run, use commands in the command prompt to run it

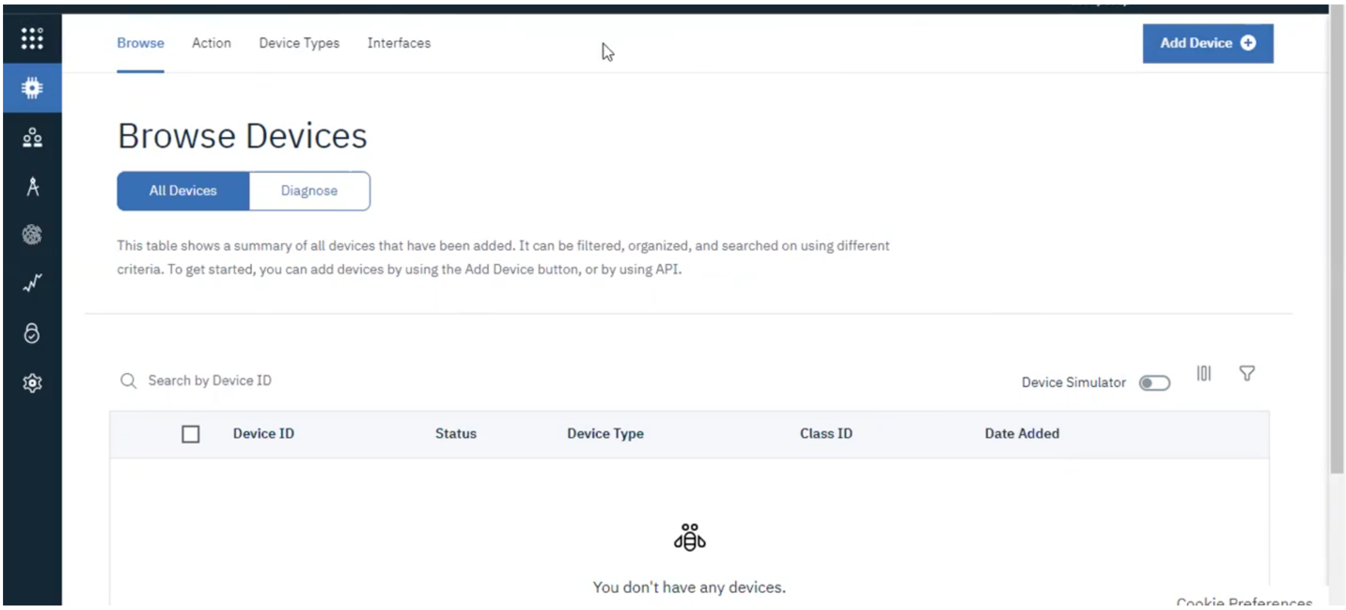
5-Download palette, IBM node red and dashboard

Point to remember-We can export or import our project anytime, all we need to do is save it as a JSON file and we can share our project file through node red with anyone, and in case for import all we need to do is the vice versa we can import and code and upload it in the import menu, thereby we can get all the nodes present in that file.

6-Go to IBM cloud, login and create an account.

7-Select a service, click on internet of things and choose the very first one.

8-Active the service and launch it in IBM Watson platform



Browse Devices

This table shows a summary of all devices that have been added. It can be filtered, organized, and searched on using different criteria. To get started, you can add devices by using the Add Device button, or by using API.

□	Device ID	Status	Device Type	Class ID	Date Added

You don't have any devices.

<u>9-</u> Click on add device type at the top

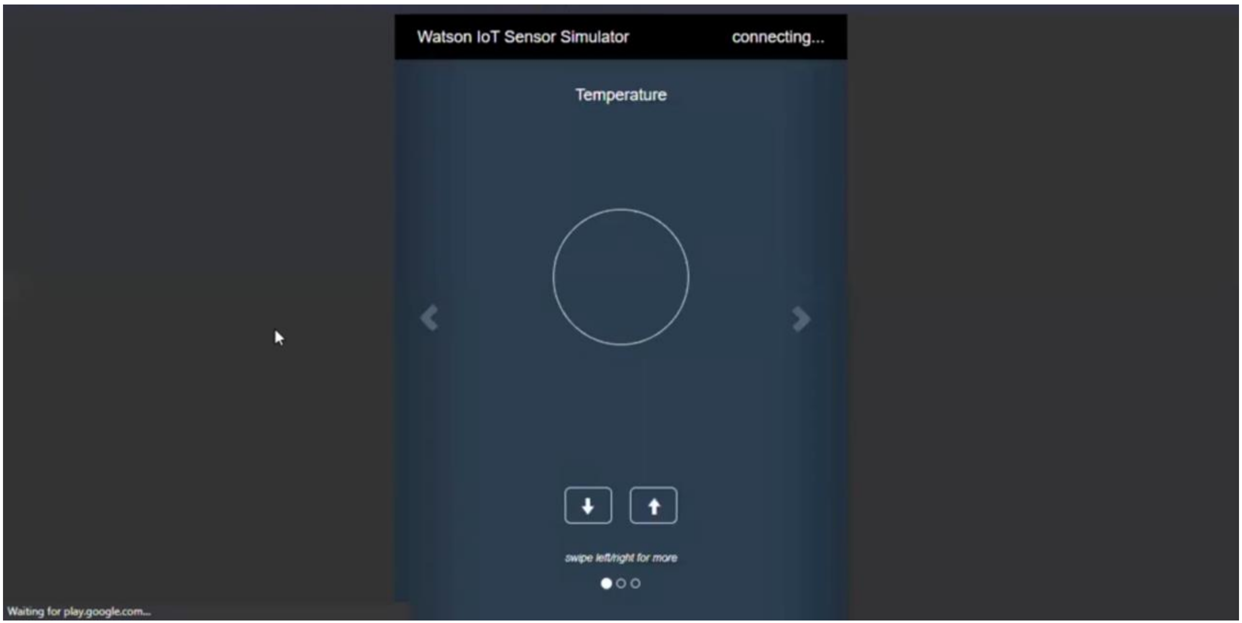
In case when we work in a big company, we have to give a name to the device, and then provide serial number, model, description, hardware version, manufacture, device class, firmware version, description location. But in this project, we don’t need this, we are going to skip it.

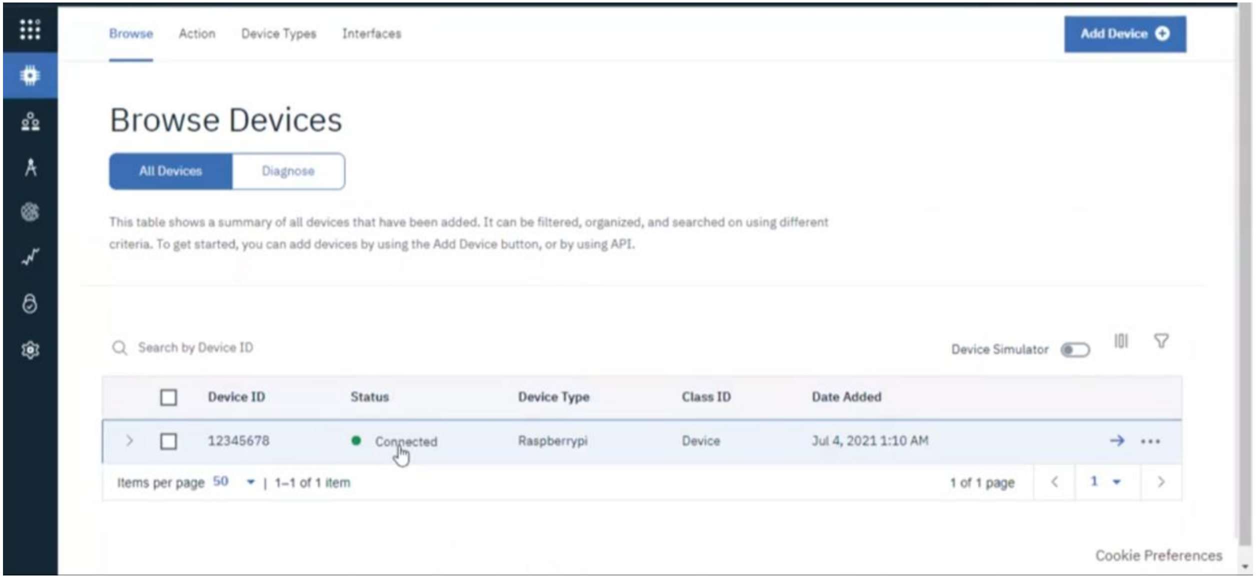
<u>10-</u> After creating a device type, register a device, type the device type and the id, again we came by the same information which we need to control in a very large scale as I said before.

<u>11-</u> One of the most important parts, enter an authentication token, after that view the summary and finish.

Now if we summarize the whole thing, all we did is create an account in IBM cloud, used IBM Watson cloud service, we added a device and registered with detail like the organization ID, device type, device id, authentication model, authentication token and what we need to do is copy and paste all this information in a notepad for further simulation. We have three things till now we have a platform to make the dashboard, we have a cloud platform for our registered device, let's say a microcontroller, now what is missing is a sensor type.

12-Use Watson iot sensor simulator, provide with every necessary detail organization ID, device type, device id, authentication model, authentication token and what we need to do is copy and paste all this information


Watson IoT Sensor Simulator
connecting...
Temperature
swipe left/right for more
Waiting for play.google.com...



Browse Devices

This table shows a summary of all devices that have been added. It can be filtered, organized, and searched on using different criteria. To get started, you can add devices by using the Add Device button, or by using API.

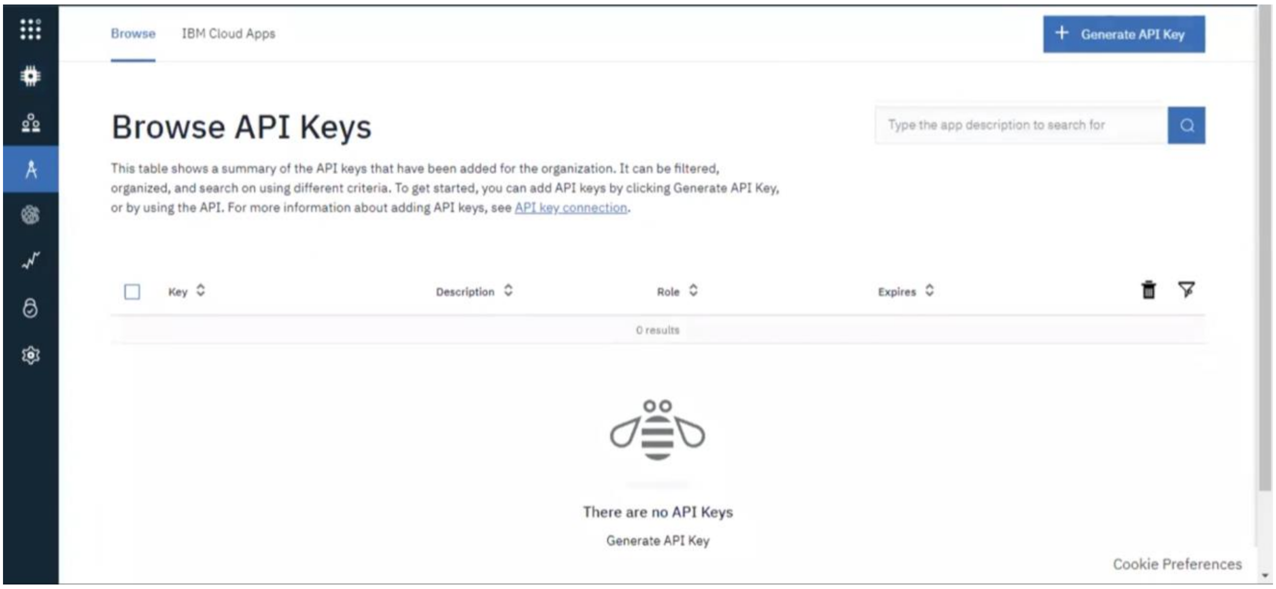
Device ID	Status	Device Type	Class ID	Date Added
12345678	● Connected	Raspberrypi	Device	Jul 4, 2021 1:10 AM

Items per page 50 | 1–1 of 1 item   1 of 1 page

13-After getting connected click on it and view the recent events, there by whatever changes we make in the sensor simulator is reflected back into the cloud, we can easily see the change in data, basically the program in it. We are done with the connection between sensor simulator and IBM cloud.

14-We have to export the data to node-red, so we are going to use or generate API, it will have its own id which will be connected to node red so that whatever changes in the cloud platform will be reflected to node red through API. Description we don’t need to give anything, select your role, standard application, thereby we get our API key and authentication token, again we have to keep this information hands on so we need to copy paste on a notepad, copy the API key and the authentication token.

Now we have configured our cloud, we have the data, we have generated our API key all we need is node red, a dashboard



Browse API Keys

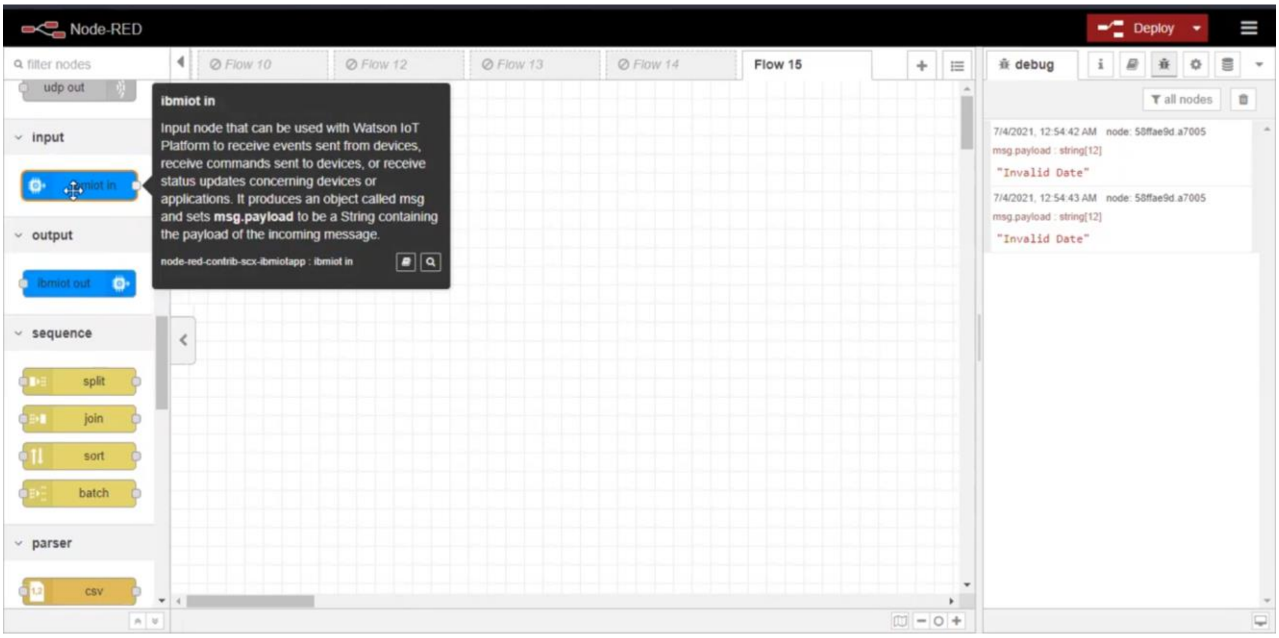
This table shows a summary of the API keys that have been added for the organization. It can be filtered, organized, and search on using different criteria. To get started, you can add API keys by clicking Generate API Key, or by using the API. For more information about adding API keys, see API key connection.

□	Key	Description	Role	Expires

0 results

There are no API Keys

15-Open node red and give the input, ibmiot in.



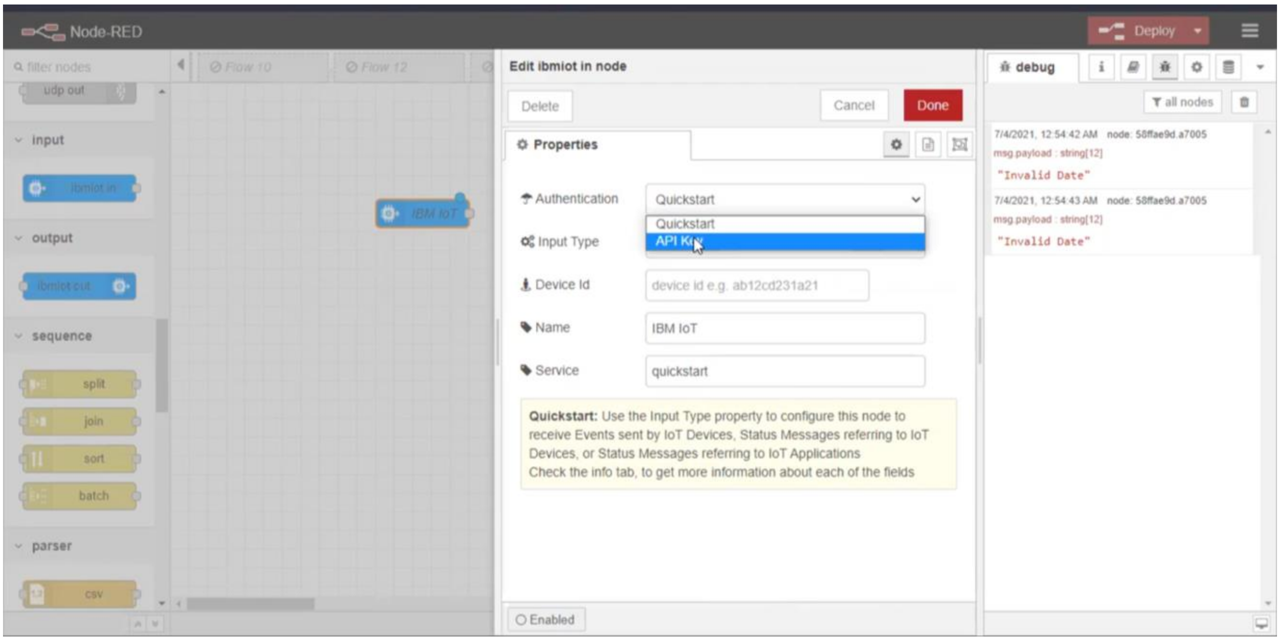
ibmiot in

Input node that can be used with Watson IoT Platform to receive events sent from devices, receive commands sent to devices, or receive status updates concerning devices or applications. It produces an object called msg and sets **msg.payload** to be a String containing the payload of the incoming message.

node-red-contrib-scx-ibmiotapp : ibmiot in

```
7/4/2021, 12:54:42 AM  node: 58ffae9d.a7005
msg.payload : string[12]
"Invalid Date"
7/4/2021, 12:54:43 AM  node: 58ffae9d.a7005
msg.payload : string[12]
"Invalid Date"
```

Change the authentication to API key by double clicking on it



Authentication: Quickstart

Input Type

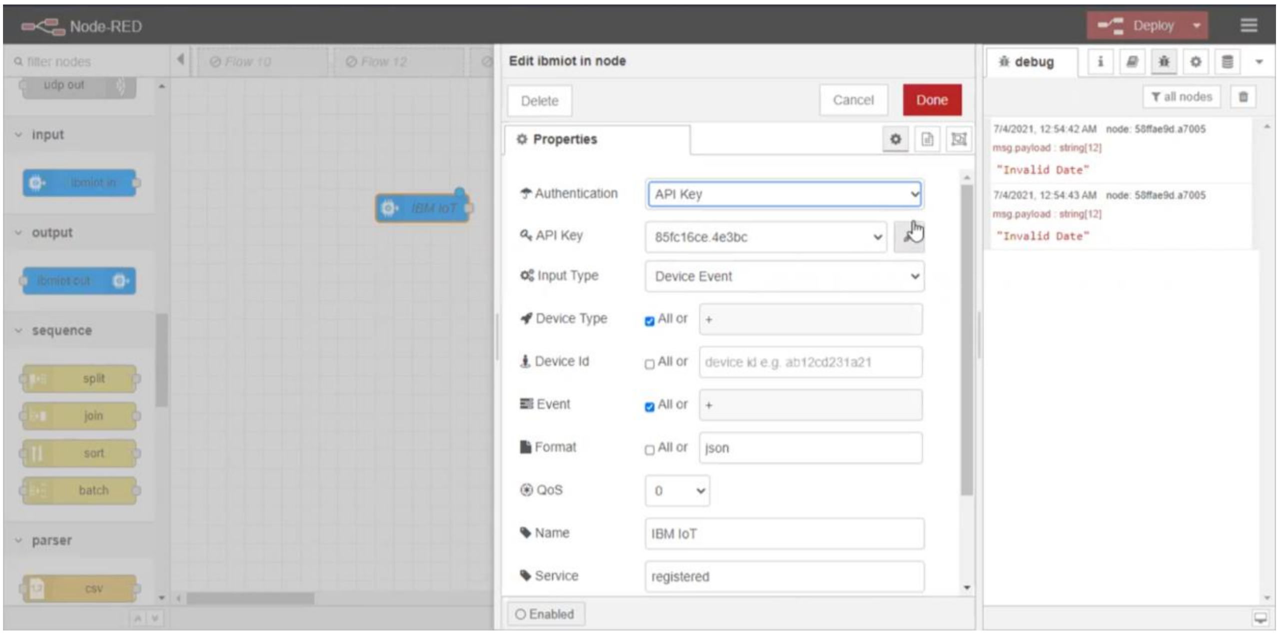
Device Id: device id e.g. ab12cd231a21

Name: IBM IoT

Service: quickstart

Quickstart: Use the Input Type property to configure this node to receive Events sent by IoT Devices, Status Messages referring to IoT Devices, or Status Messages referring to IoT Applications
Check the info tab, to get more information about each of the fields

```
7/4/2021, 12:54:42 AM  node: 58ffae9d.a7005
msg.payload : string[12]
"Invalid Date"
7/4/2021, 12:54:43 AM  node: 58ffae9d.a7005
msg.payload : string[12]
"Invalid Date"
```

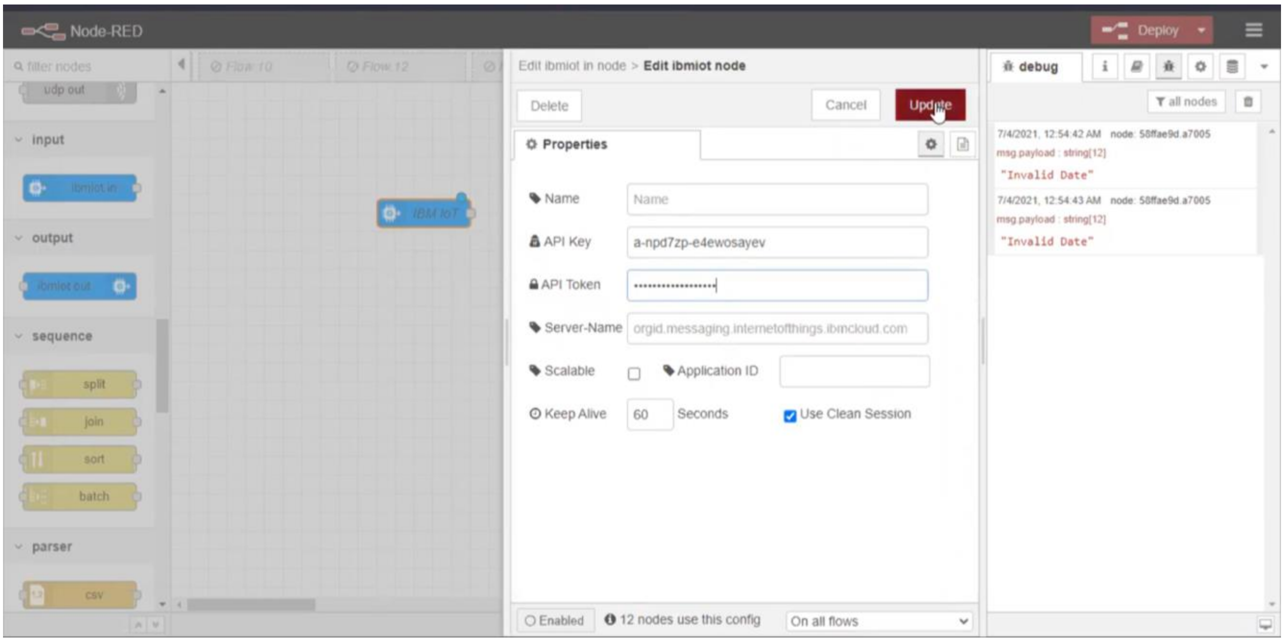


Edit ibmiot in node

- Authentication: API Key
- API Key: 85fc16ce.4e3bc
- Input Type: Device Event
- Device Type: ☑ All or +
- Device Id: ☐ All or device id e.g. ab12cd231a21
- Event: ☑ All or +
- Format: ☐ All or json
- QoS: 0
- Name: IBM IoT
- Service: registered

```
7/4/2021, 12:54:42 AM  node: 58ffae9d.a7005
msg.payload : string[12]
"Invalid Date"
7/4/2021, 12:54:43 AM  node: 58ffae9d.a7005
msg.payload : string[12]
"Invalid Date"
```

Click on update after giving the API key and the token.



Edit ibmiot in node > Edit ibmiot node

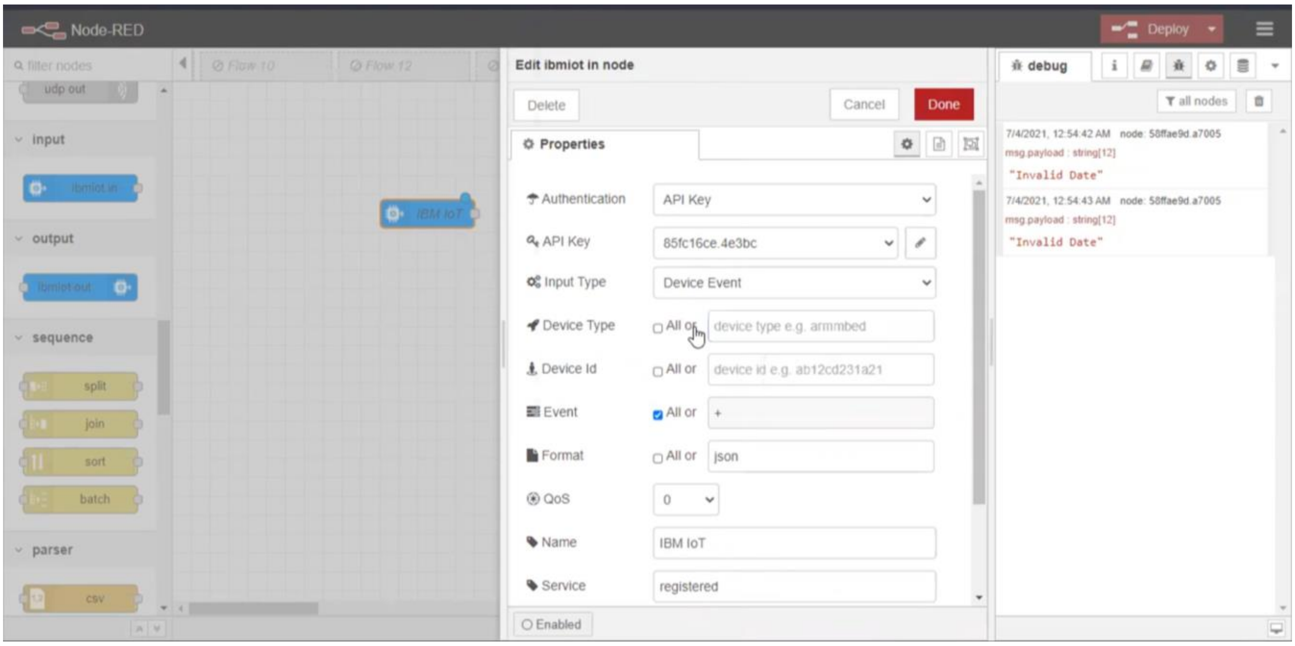
- Name: Name
- API Key: a-npd7zp-e4ewosayev
- API Token: ••••••••••••••••••
- Server-Name: orgid.messaging.internetofthings.ibmcloud.com
- Scalable: ☐
- Application ID:
- Keep Alive: 60 Seconds
- ☑ Use Clean Session

12 nodes use this config

On all flows

```
7/4/2021, 12:54:42 AM  node: 58ffae9d.a7005
msg.payload : string[12]
"Invalid Date"
7/4/2021, 12:54:43 AM  node: 58ffae9d.a7005
msg.payload : string[12]
"Invalid Date"
```

Fill with the necessary detail, the reason I mentioned to note it down in your notepad for easy access.

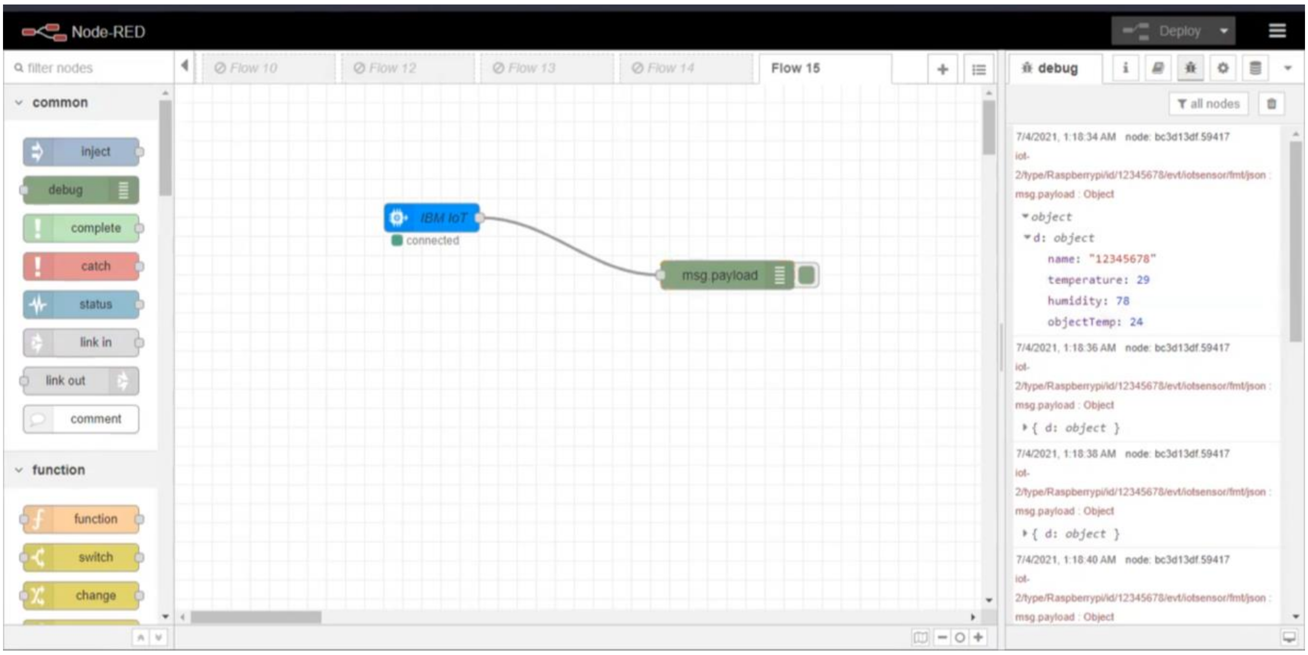


Edit ibmiot in node

Authentication: API Key
API Key: 85fc16ce.4e3bc
Input Type: Device Event
Device Type: □ All or device type e.g. armmbed
Device Id: □ All or device id e.g. ab12cd231a21
Event: ☑ All or +
Format: □ All or json
QoS: 0
Name: IBM IoT
Service: registered
○ Enabled

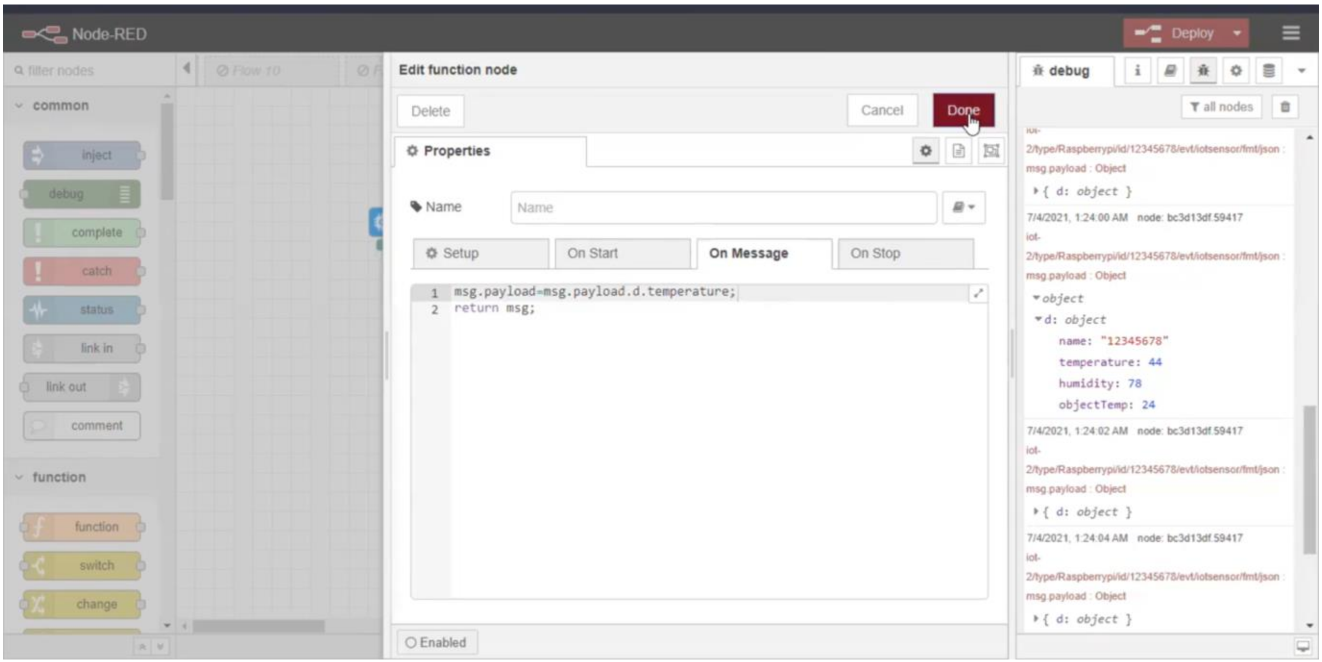
```
7/4/2021, 12:54:42 AM   node: 58ffae9d.a7005
msg.payload : string[12]
 "Invalid Date"
7/4/2021, 12:54:43 AM   node: 58ffae9d.a7005
msg.payload : string[12]
 "Invalid Date"
```

Now for real time check connect the node in the dashboard as we can see that in the right-hand side debug, we are getting the output the same as we are putting in as input in the Watson IBM sensor simulator, that means if we increase the temperature or humidity, we can get the output reading in the very debug section at node red.



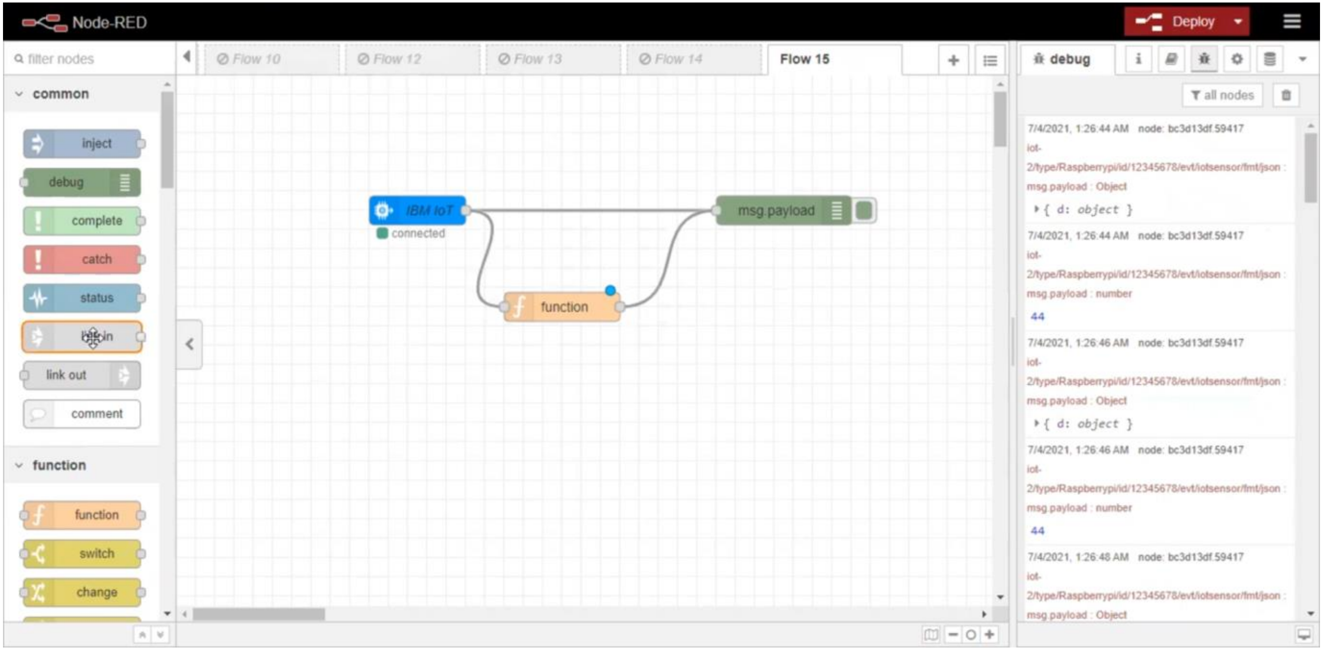
```
7/4/2021, 1:18:34 AM   node: bc3d13df.59417
iot-2/type/Raspberrypi/id/12345678/evt/iotsensor/fmt/json :
msg.payload : Object
 ▾object
  ▾d: object
     name: "12345678"
     temperature: 29
     humidity: 78
     objectTemp: 24
7/4/2021, 1:18:36 AM   node: bc3d13df.59417
iot-2/type/Raspberrypi/id/12345678/evt/iotsensor/fmt/json :
msg.payload : Object
 ▸{ d: object }
7/4/2021, 1:18:38 AM   node: bc3d13df.59417
iot-2/type/Raspberrypi/id/12345678/evt/iotsensor/fmt/json :
msg.payload : Object
 ▸{ d: object }
7/4/2021, 1:18:40 AM   node: bc3d13df.59417
iot-2/type/Raspberrypi/id/12345678/evt/iotsensor/fmt/json :
msg.payload : Object
```

15-We cannot give these data to any farmers for use, so what we need to do is arranging the nodes in the dashboard in a simple format so that everyone can get access to the data easily. We need to add a function node and edit the function node.



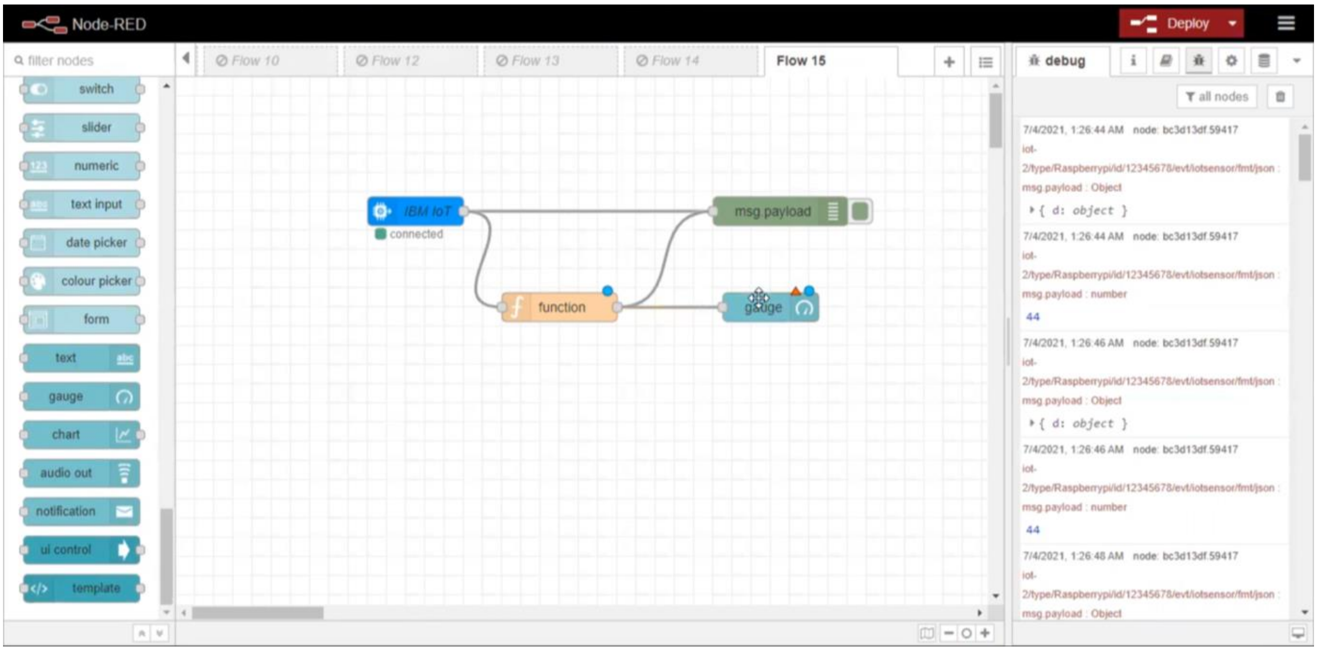
```
msg.payload=msg.payload.d.temperature;
return msg;
```

```
iot-2/type/Raspberrypi/id/12345678/evt/iotsensor/fmt/json : msg.payload : Object
▸{ d: object }
7/4/2021, 1:24:00 AM   node: bc3d13df.59417
iot-2/type/Raspberrypi/id/12345678/evt/iotsensor/fmt/json : msg.payload : Object
▾object
 ▾d: object
    name: "12345678"
    temperature: 44
    humidity: 78
    objectTemp: 24
7/4/2021, 1:24:02 AM   node: bc3d13df.59417
iot-2/type/Raspberrypi/id/12345678/evt/iotsensor/fmt/json : msg.payload : Object
▸{ d: object }
7/4/2021, 1:24:04 AM   node: bc3d13df.59417
iot-2/type/Raspberrypi/id/12345678/evt/iotsensor/fmt/json : msg.payload : Object
▸{ d: object }
```

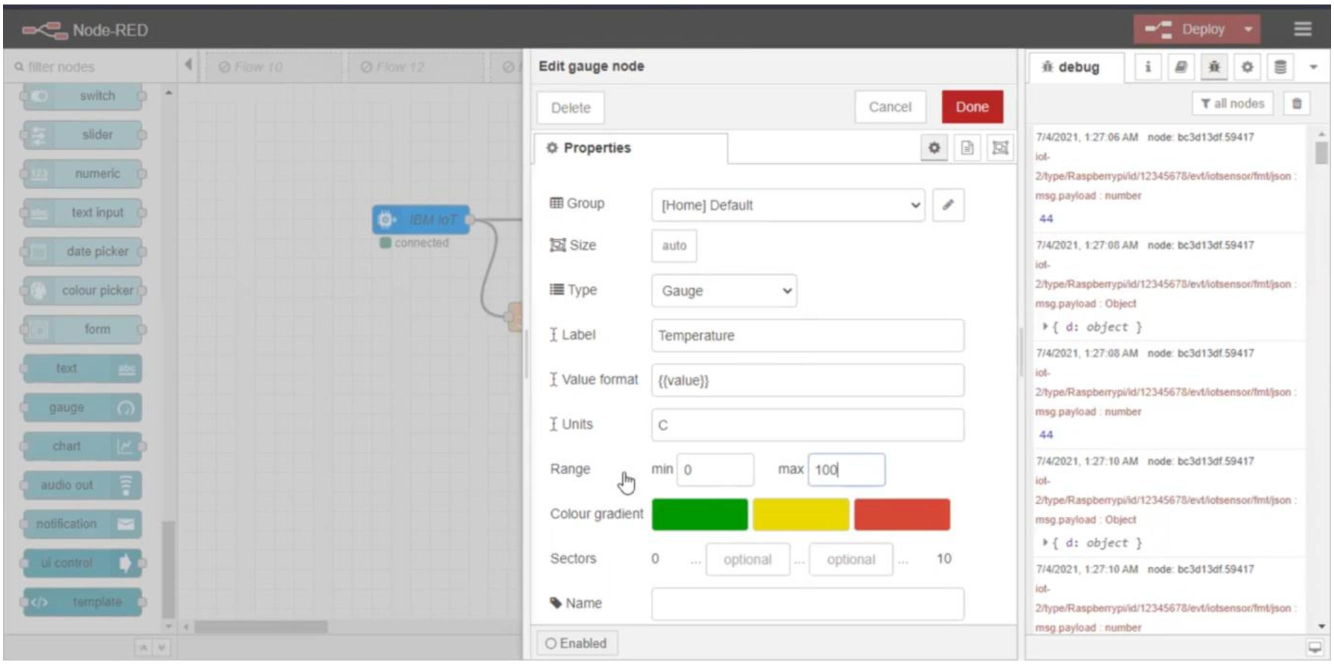


```
7/4/2021, 1:26:44 AM   node: bc3d13df.59417
iot-2/type/Raspberrypi/id/12345678/evt/iotsensor/fmt/json : msg.payload : Object
▸{ d: object }
7/4/2021, 1:26:44 AM   node: bc3d13df.59417
iot-2/type/Raspberrypi/id/12345678/evt/iotsensor/fmt/json : msg.payload : number
44
7/4/2021, 1:26:46 AM   node: bc3d13df.59417
iot-2/type/Raspberrypi/id/12345678/evt/iotsensor/fmt/json : msg.payload : Object
▸{ d: object }
7/4/2021, 1:26:46 AM   node: bc3d13df.59417
iot-2/type/Raspberrypi/id/12345678/evt/iotsensor/fmt/json : msg.payload : number
44
7/4/2021, 1:26:48 AM   node: bc3d13df.59417
iot-2/type/Raspberrypi/id/12345678/evt/iotsensor/fmt/json : msg.payload : Object
```

As we can see the temperature is easily be seen, now we will add another node as gauge so that we can see the user interface, the ui where the temperature can easily be seen with the parameters set



```
7/4/2021, 1:26:44 AM  node: bc3d13df.59417
iot-2/type/Raspberrypi/id/12345678/evt/iotsensor/fmt/json : msg.payload : Object
▸{ d: object }
7/4/2021, 1:26:44 AM  node: bc3d13df.59417
iot-2/type/Raspberrypi/id/12345678/evt/iotsensor/fmt/json : msg.payload : number
44
7/4/2021, 1:26:46 AM  node: bc3d13df.59417
iot-2/type/Raspberrypi/id/12345678/evt/iotsensor/fmt/json : msg.payload : Object
▸{ d: object }
7/4/2021, 1:26:46 AM  node: bc3d13df.59417
iot-2/type/Raspberrypi/id/12345678/evt/iotsensor/fmt/json : msg.payload : number
44
7/4/2021, 1:26:48 AM  node: bc3d13df.59417
iot-2/type/Raspberrypi/id/12345678/evt/iotsensor/fmt/json : msg.payload : Object
```



Edit gauge node

Properties

Group	[Home] Default
Size	auto
Type	Gauge
Label	Temperature
Value format	{{value}}
Units	C
Range	min 0 max 100
Colour gradient	🟩 🟨 🟥
Sectors	0 ... optional ... optional ... 10
Name	

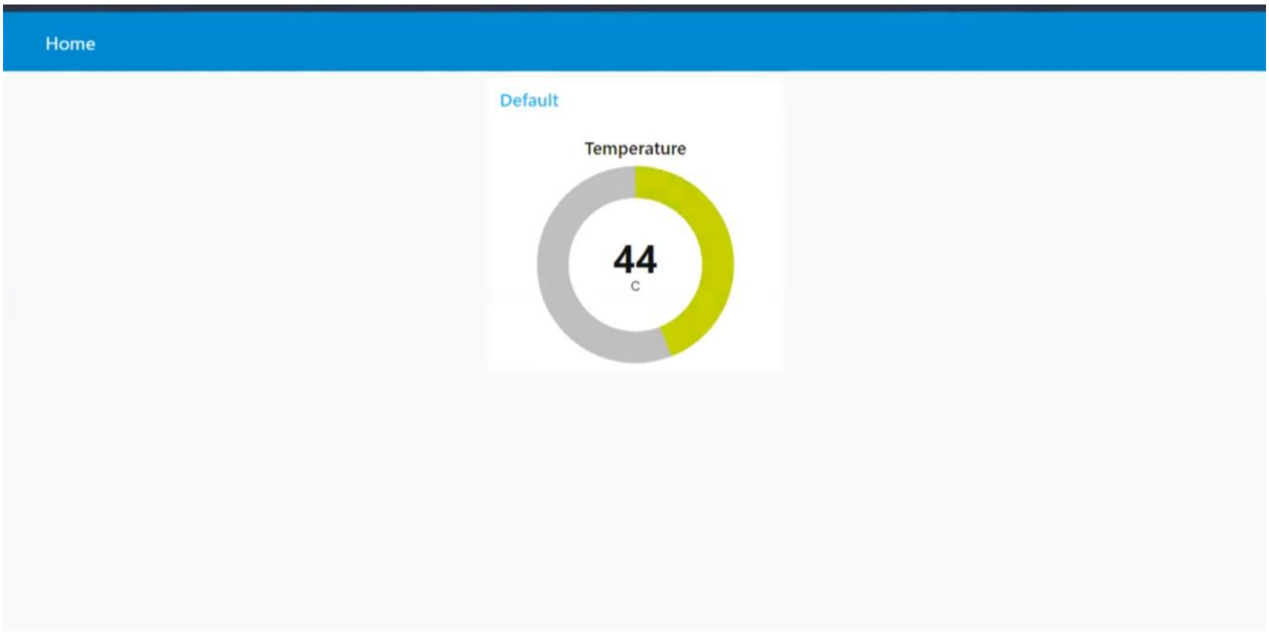
○ Enabled

```
7/4/2021, 1:27:06 AM  node: bc3d13df.59417
iot-2/type/Raspberrypi/id/12345678/evt/iotsensor/fmt/json : msg.payload : number
44
7/4/2021, 1:27:08 AM  node: bc3d13df.59417
iot-2/type/Raspberrypi/id/12345678/evt/iotsensor/fmt/json : msg.payload : Object
▸{ d: object }
7/4/2021, 1:27:08 AM  node: bc3d13df.59417
iot-2/type/Raspberrypi/id/12345678/evt/iotsensor/fmt/json : msg.payload : number
44
7/4/2021, 1:27:10 AM  node: bc3d13df.59417
iot-2/type/Raspberrypi/id/12345678/evt/iotsensor/fmt/json : msg.payload : Object
▸{ d: object }
7/4/2021, 1:27:10 AM  node: bc3d13df.59417
iot-2/type/Raspberrypi/id/12345678/evt/iotsensor/fmt/json : msg.payload : number
```

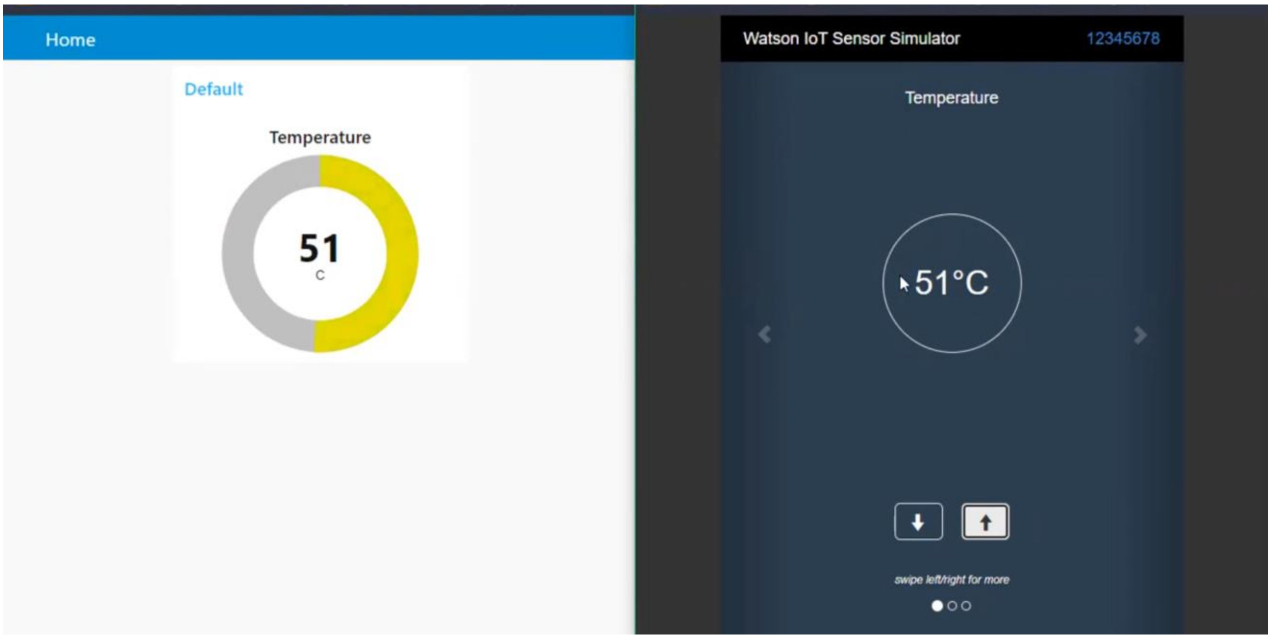
16-Now it's time to visualise the date by just copying the link of our node red,

127.0.0.1:1880/#flow/7e89ed2bdeb7861 till

number:1800/ and add ui into a new tab like number:1800/ui

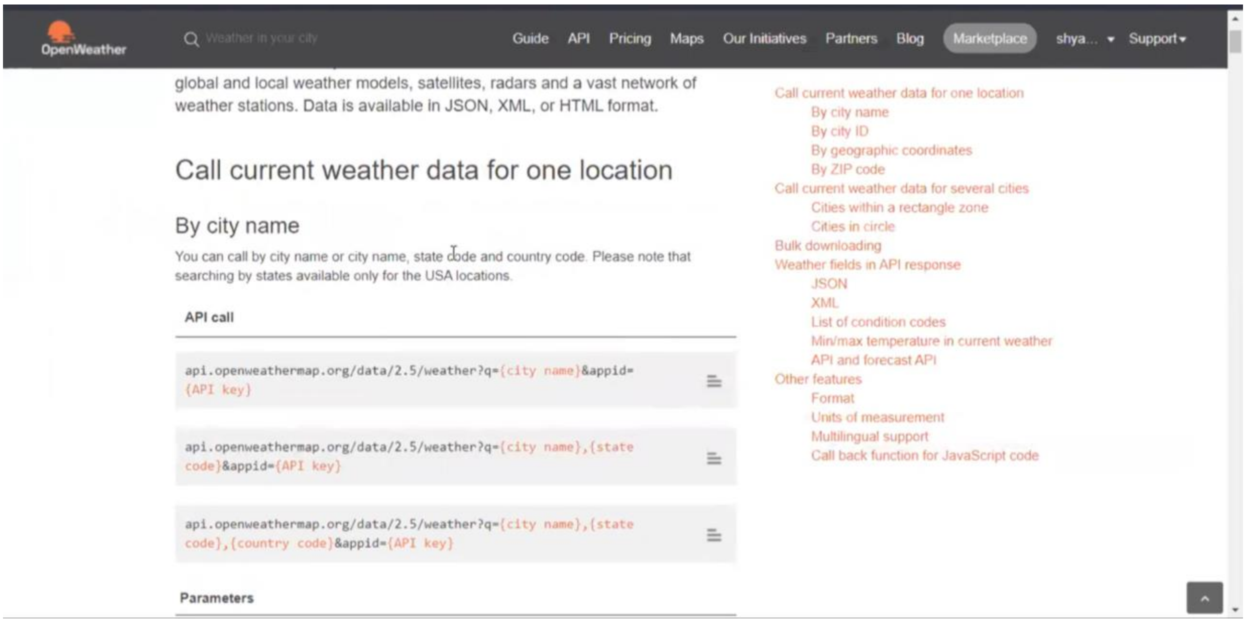


And there by we successfully interfaced node red and Watson iot sensor simulator parallelly.



<u>**17-**</u> Now it's time to connect it with the real time weather data which we get easily access from internet, and for that we need an API, and thereby we have to connect with an application known as open weather API. After creating account and finishing off, all we need to do is to go to the API doc and select an API call.

After that we have to copy the API call in notepad and put the city name and API key accordingly.


OpenWeather
Weather in your city
Guide
API
Pricing
Maps
Our Initiatives
Partners
Blog
Marketplace
shya...
Support
Call current weather data for one location
By city name
By city ID
By geographic coordinates
By ZIP code
Call current weather data for several cities
Cities within a rectangle zone
Cities in circle
Bulk downloading
Weather fields in API response
JSON
XML
List of condition codes
Min/max temperature in current weather API and forecast API
Other features
Format
Units of measurement
Multilingual support
Call back function for JavaScript code

global and local weather models, satellites, radars and a vast network of weather stations. Data is available in JSON, XML, or HTML format.

Call current weather data for one location

By city name

You can call by city name or city name, state code and country code. Please note that searching by states available only for the USA locations.

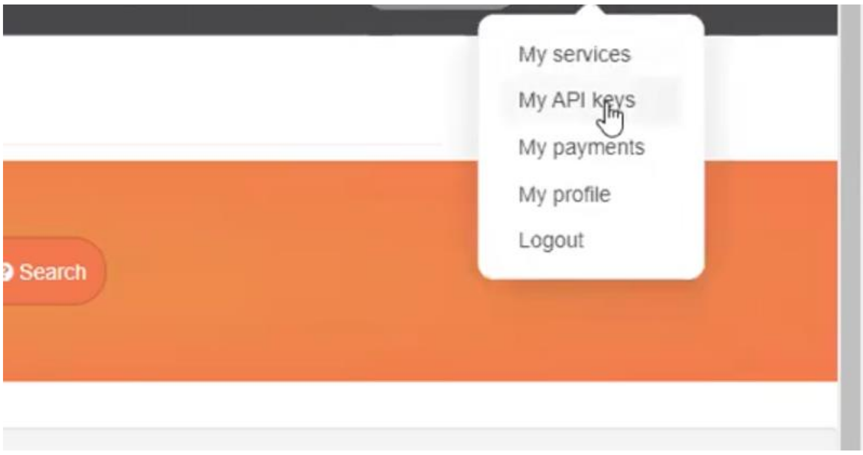
API call

```
api.openweathermap.org/data/2.5/weather?q={city name}&appid=
{API key}
```

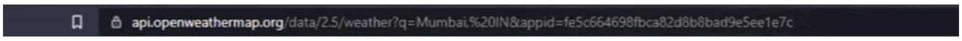
```
api.openweathermap.org/data/2.5/weather?q={city name},{state
code}&appid={API key}
```

```
api.openweathermap.org/data/2.5/weather?q={city name},{state
code},{country code}&appid={API key}
```

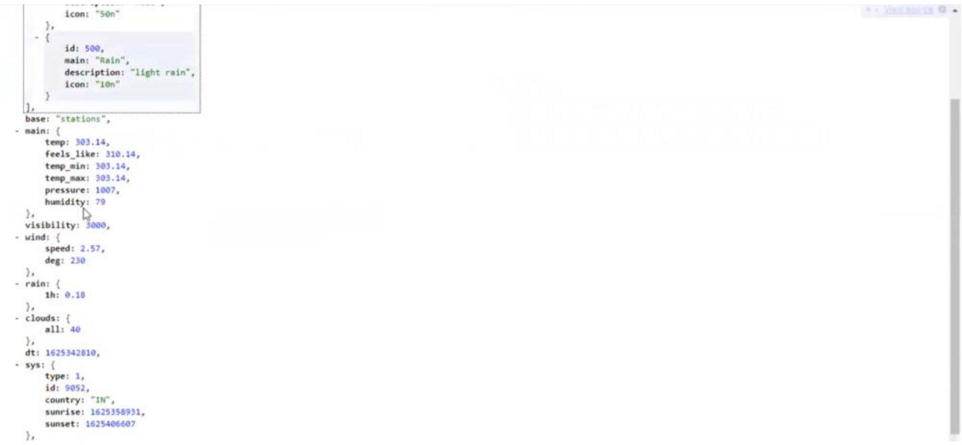
Parameters



Copy the API call in a new tab, it might take some time, and here we are all set to interface the live data with node red.

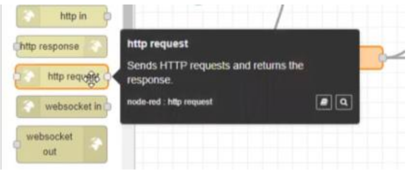


```
api.openweathermap.org/data/2.5/weather?q=Mumbai,%20IN&appid=fe5c664698fbca82d8b8bad9e5ee1e7c
```



```
            icon: "50n"
        },
    - {
            id: 500,
            main: "Rain",
            description: "light rain",
            icon: "10n"
        }
    ],
    base: "stations",
  - main: {
        temp: 303.14,
        feels_like: 310.14,
        temp_min: 303.14,
        temp_max: 303.14,
        pressure: 1007,
        humidity: 79
    },
    visibility: 3000,
  - wind: {
        speed: 2.57,
        deg: 230
    },
  - rain: {
        1h: 0.18
    },
  - clouds: {
        all: 40
    },
    dt: 1625342810,
  - sys: {
        type: 1,
        id: 9052,
        country: "IN",
        sunrise: 1625358931,
        sunset: 1625406607
    },
```

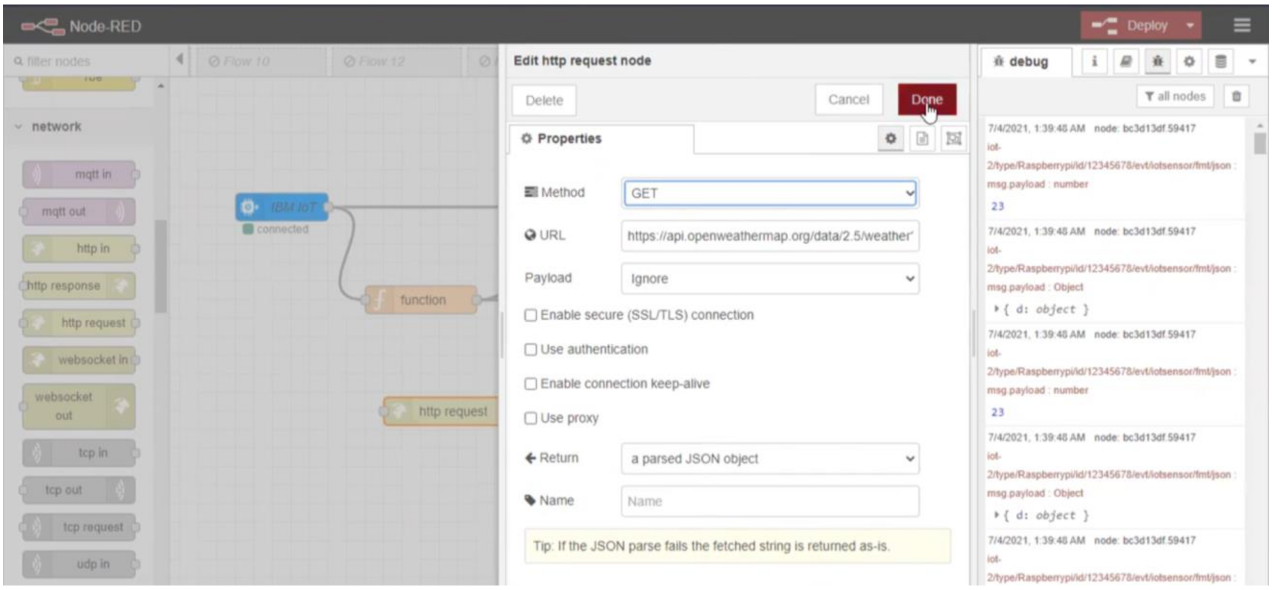
18-Adding nodes to node red copying the URL extension and making changes as given below,



http request

Sends HTTP requests and returns the response.

node-red : http request



Edit http request node

Delete Cancel Done

Properties

Method: GET

URL: https://api.openweathermap.org/data/2.5/weather'

Payload: Ignore

☐ Enable secure (SSL/TLS) connection

☐ Use authentication

☐ Enable connection keep-alive

☐ Use proxy

Return: a parsed JSON object

Name: Name

Tip: If the JSON parse fails the fetched string is returned as-is.

7/4/2021, 1:39:48 AM node: bc3d13df.59417
iot-2/type/Raspberrypi/id/12345678/evt/iotsensor/fmt/json : msg.payload : number
23

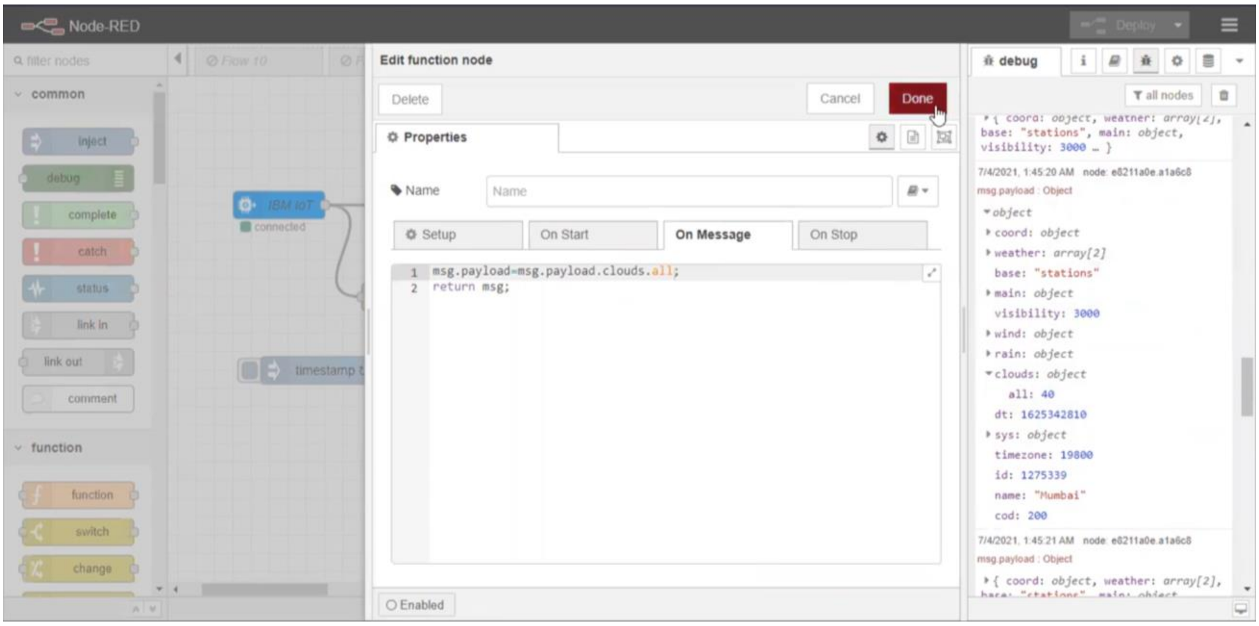
7/4/2021, 1:39:48 AM node: bc3d13df.59417
iot-2/type/Raspberrypi/id/12345678/evt/iotsensor/fmt/json : msg.payload : Object
▸{ d: object }

7/4/2021, 1:39:48 AM node: bc3d13df.59417
iot-2/type/Raspberrypi/id/12345678/evt/iotsensor/fmt/json : msg.payload : number
23

7/4/2021, 1:39:48 AM node: bc3d13df.59417
iot-2/type/Raspberrypi/id/12345678/evt/iotsensor/fmt/json : msg.payload : Object
▸{ d: object }

7/4/2021, 1:39:48 AM node: bc3d13df.59417
iot-2/type/Raspberrypi/id/12345678/evt/iotsensor/fmt/json :

And again, we have to add a function node and make the changes as following,



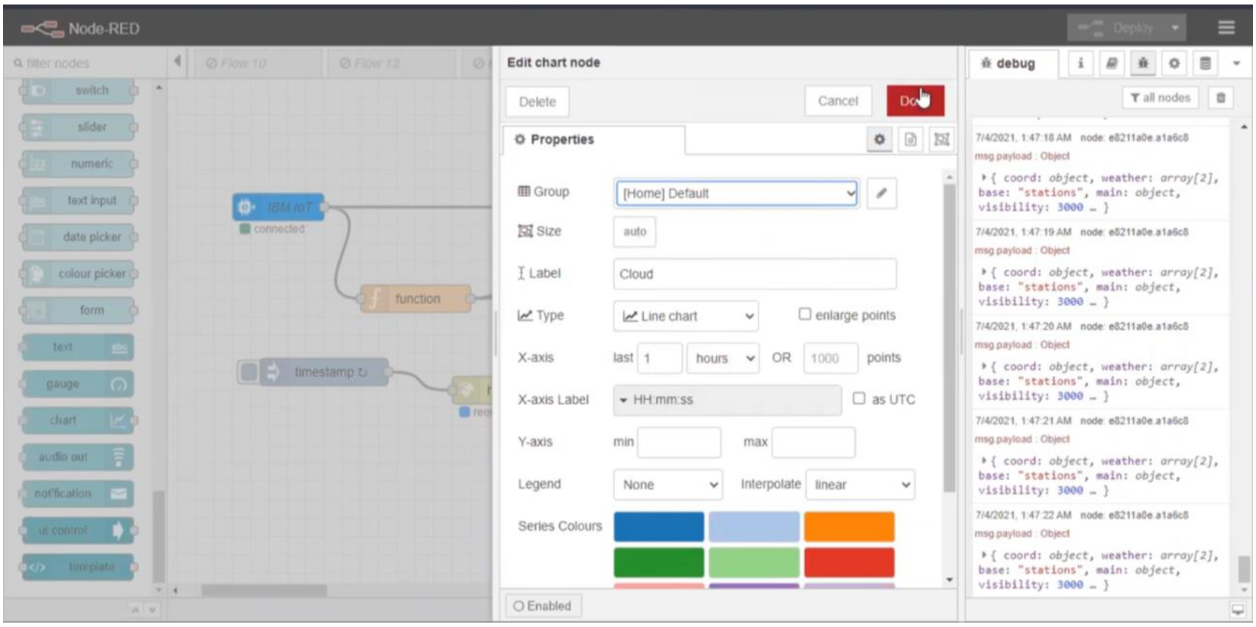
Edit function node

Name: Name

```
msg.payload=msg.payload.clouds.all;
return msg;
```

```
▸{ coord: object, weather: array[2],
base: "stations", main: object,
visibility: 3000 … }
7/4/2021, 1:45:20 AM  node: e8211a0e.a1a6c8
msg.payload : Object
▾object
 ▸coord: object
 ▸weather: array[2]
  base: "stations"
 ▸main: object
  visibility: 3000
 ▸wind: object
 ▸rain: object
 ▾clouds: object
    all: 40
  dt: 1625342810
 ▸sys: object
  timezone: 19800
  id: 1275339
  name: "Mumbai"
  cod: 200
7/4/2021, 1:45:21 AM  node: e8211a0e.a1a6c8
msg.payload : Object
▸{ coord: object, weather: array[2],
base: "stations", main: object,
```

Add another node as chart



Edit chart node

Group: [Home] Default

Size: auto

Label: Cloud

Type: Line chart ☐ enlarge points

X-axis: last 1 hours OR 1000 points

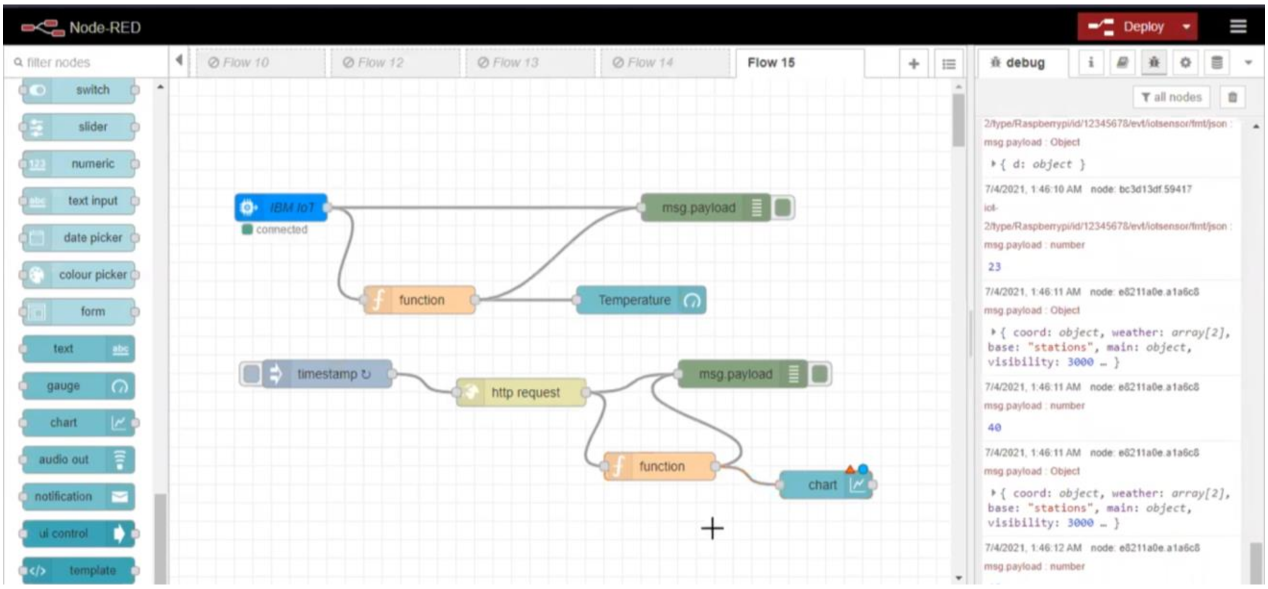
X-axis Label: HH:mm:ss ☐ as UTC

Y-axis: min max

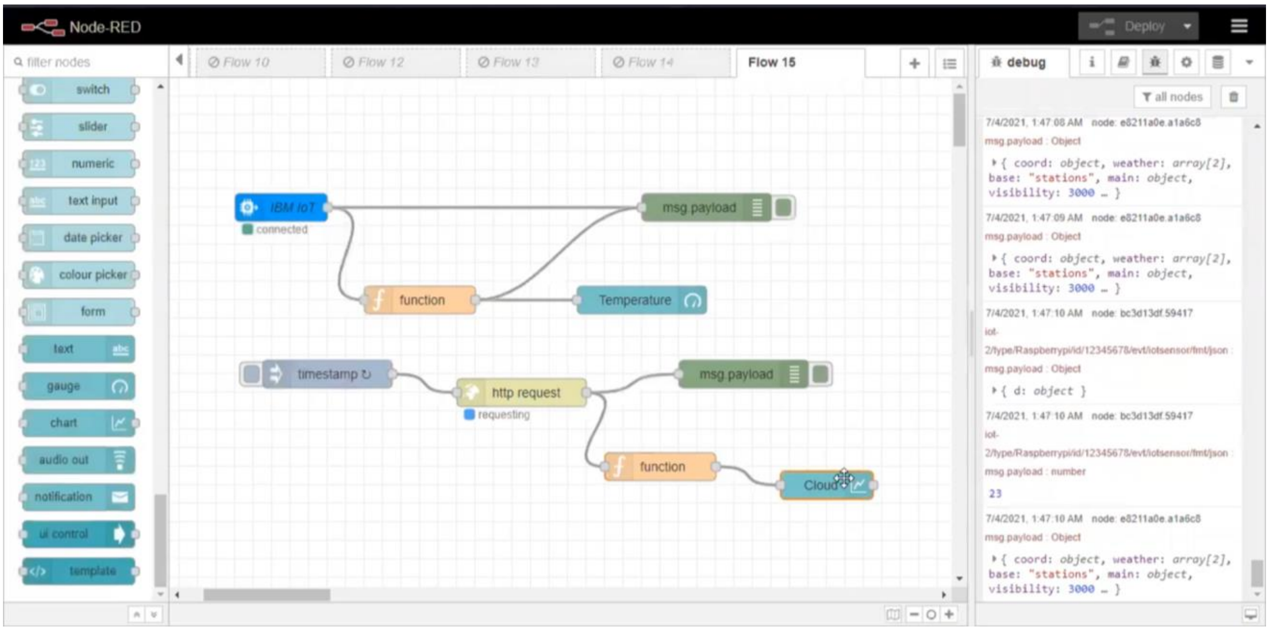
Legend: None Interpolate: linear

Series Colours

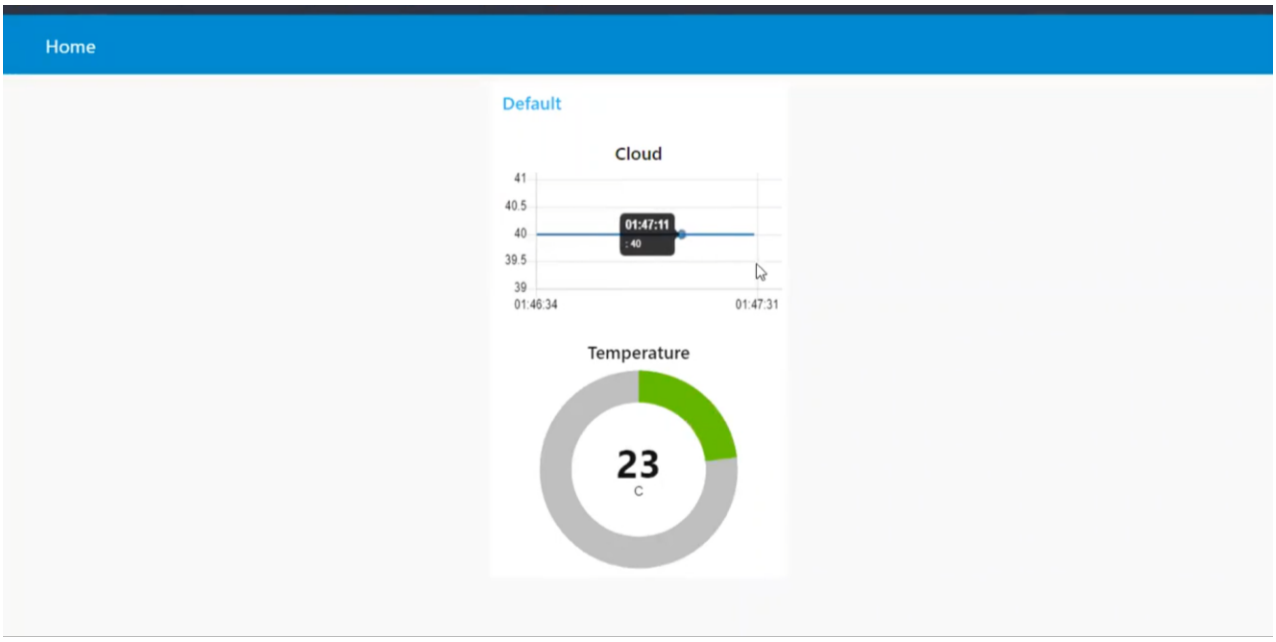
```
7/4/2021, 1:47:18 AM  node: e8211a0e.a1a6c8
msg.payload : Object
▸{ coord: object, weather: array[2],
base: "stations", main: object,
visibility: 3000 … }
7/4/2021, 1:47:19 AM  node: e8211a0e.a1a6c8
msg.payload : Object
▸{ coord: object, weather: array[2],
base: "stations", main: object,
visibility: 3000 … }
7/4/2021, 1:47:20 AM  node: e8211a0e.a1a6c8
msg.payload : Object
▸{ coord: object, weather: array[2],
base: "stations", main: object,
visibility: 3000 … }
7/4/2021, 1:47:21 AM  node: e8211a0e.a1a6c8
msg.payload : Object
▸{ coord: object, weather: array[2],
base: "stations", main: object,
visibility: 3000 … }
7/4/2021, 1:47:22 AM  node: e8211a0e.a1a6c8
msg.payload : Object
▸{ coord: object, weather: array[2],
base: "stations", main: object,
visibility: 3000 … }
```



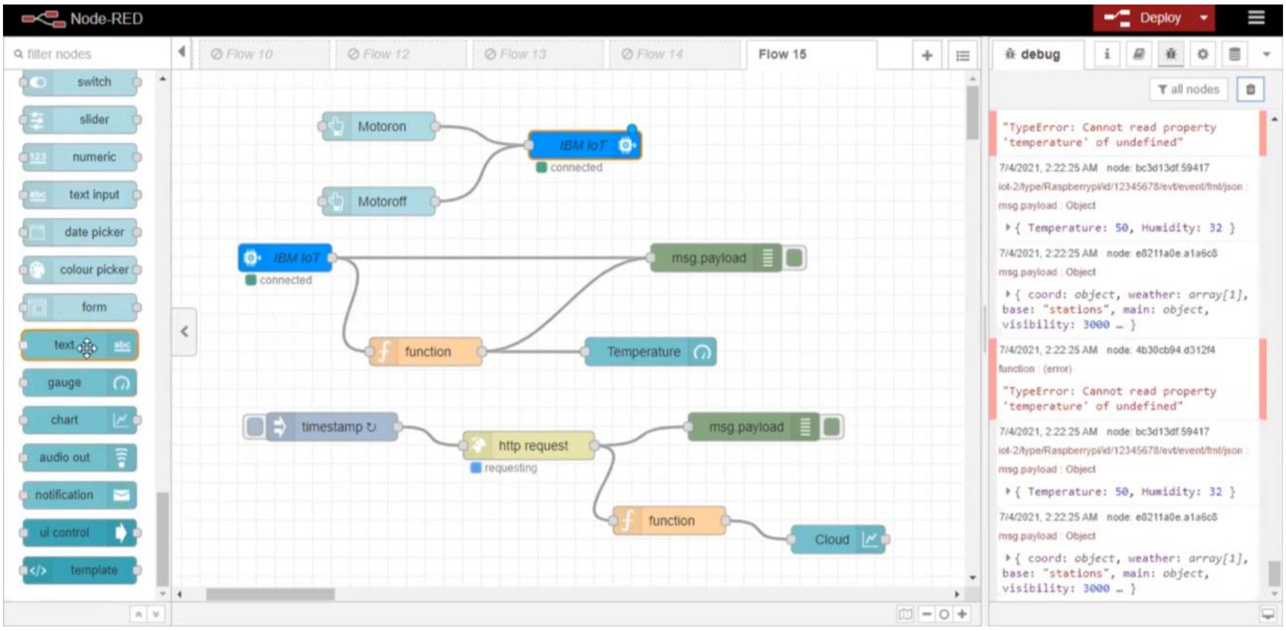
```
2/type/Raspberrypi/id/12345678/evt/iotsensor/fmt/json :
msg.payload : Object
▸{ d: object }
7/4/2021, 1:46:10 AM   node: bc3d13df.59417
iot-
2/type/Raspberrypi/id/12345678/evt/iotsensor/fmt/json :
msg.payload : number
23
7/4/2021, 1:46:11 AM   node: e8211a0e.a1a6c8
msg.payload : Object
▸{ coord: object, weather: array[2],
base: "stations", main: object,
visibility: 3000 … }
7/4/2021, 1:46:11 AM   node: e8211a0e.a1a6c8
msg.payload : number
40
7/4/2021, 1:46:11 AM   node: e8211a0e.a1a6c8
msg.payload : Object
▸{ coord: object, weather: array[2],
base: "stations", main: object,
visibility: 3000 … }
7/4/2021, 1:46:12 AM   node: e8211a0e.a1a6c8
msg.payload : number
```



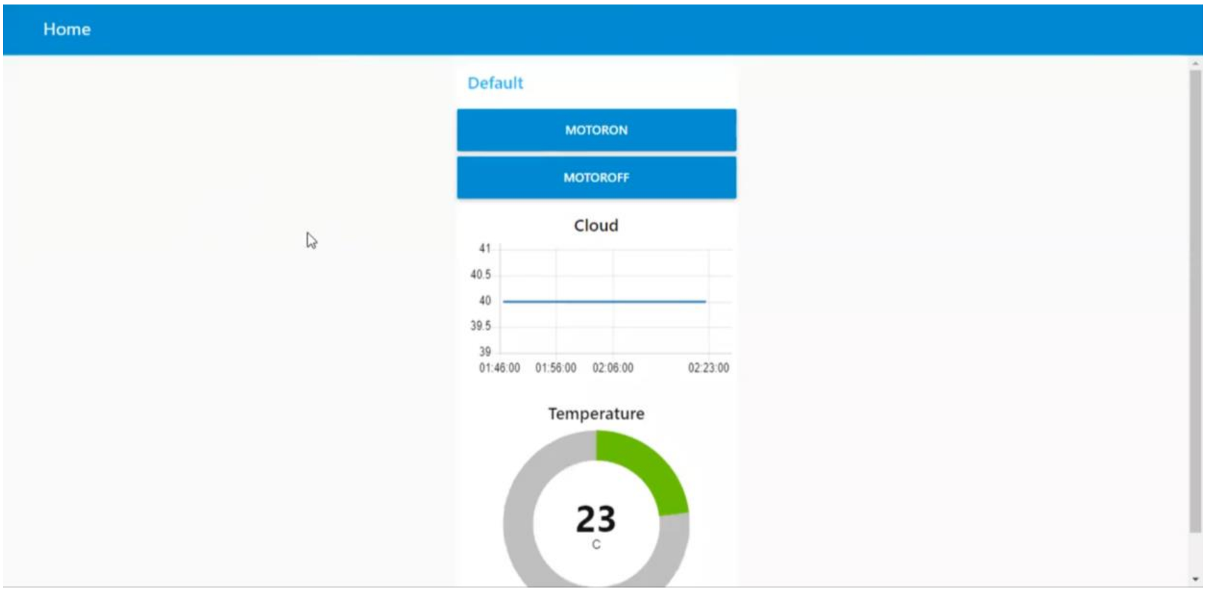
```
7/4/2021, 1:47:08 AM   node: e8211a0e.a1a6c8
msg.payload : Object
▸{ coord: object, weather: array[2],
base: "stations", main: object,
visibility: 3000 … }
7/4/2021, 1:47:09 AM   node: e8211a0e.a1a6c8
msg.payload : Object
▸{ coord: object, weather: array[2],
base: "stations", main: object,
visibility: 3000 … }
7/4/2021, 1:47:10 AM   node: bc3d13df.59417
iot-
2/type/Raspberrypi/id/12345678/evt/iotsensor/fmt/json :
msg.payload : Object
▸{ d: object }
7/4/2021, 1:47:10 AM   node: bc3d13df.59417
iot-
2/type/Raspberrypi/id/12345678/evt/iotsensor/fmt/json :
msg.payload : number
23
7/4/2021, 1:47:10 AM   node: e8211a0e.a1a6c8
msg.payload : Object
▸{ coord: object, weather: array[2],
base: "stations", main: object,
visibility: 3000 … }
```

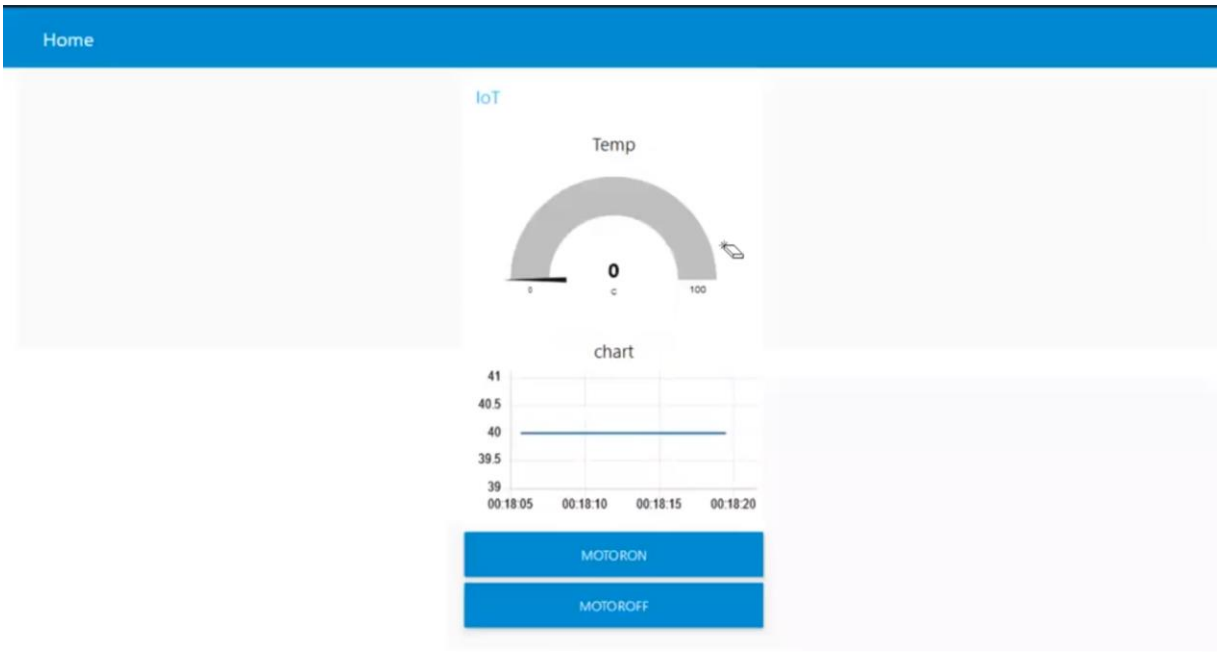


All we are left with motors and interfacing it with IBM iot .

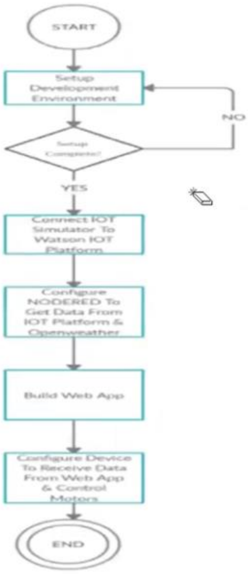


```
"TypeError: Cannot read property
'temperature' of undefined"
7/4/2021, 2:22:25 AM  node: bc3d13df.59417
iot-2/type/Raspberrypi/id/12345678/evt/event/fmt/json :
msg.payload : Object
▸{ Temperature: 50, Humidity: 32 }
7/4/2021, 2:22:25 AM  node: e8211a0e.a1a6c8
msg.payload : Object
▸{ coord: object, weather: array[1],
base: "stations", main: object,
visibility: 3000 … }
7/4/2021, 2:22:25 AM  node: 4b30cb94.d312f4
function : (error)
"TypeError: Cannot read property
'temperature' of undefined"
7/4/2021, 2:22:25 AM  node: bc3d13df.59417
iot-2/type/Raspberrypi/id/12345678/evt/event/fmt/json :
msg.payload : Object
▸{ Temperature: 50, Humidity: 32 }
7/4/2021, 2:22:25 AM  node: e8211a0e.a1a6c8
msg.payload : Object
▸{ coord: object, weather: array[1],
base: "stations", main: object,
visibility: 3000 … }
```




Home
IoT
Temp
0
0
c
100
chart
41
40.5
40
39.5
39
00:18:05
00:18:10
00:18:15
00:18:20
MOTORON
MOTOROFF

Flow chart


START
Setup Development Environment
Setup Complete?
NO
YES
Connect IOT Simulator To Watson IOT Platform
Configure NODERED To Get Data From IOT Platform & Openweather
Build Web App
Configure Device To Receive Data From Web App & Control Motors
END