

SPRINT DELIVERY – 2

| | |
|---------|------------------------------------------|
| Team ID | PNT2022TMID43969 |
| Project | IoT Enabled Smart Farming Application |
| Date | 15 November 2022 |

5. Building Project

Connecting IOT Simulator to IBM Watson

IOT Platform

Open link provided in above section 4.3

Give the credentials of your device in IBM Watson

IOT Platform Click on connect

My credentials given to simulator are: OrgID:

4clor3 api: **a-157uf3-**

f5rg4qxp3 Device type: **NodeMCU**

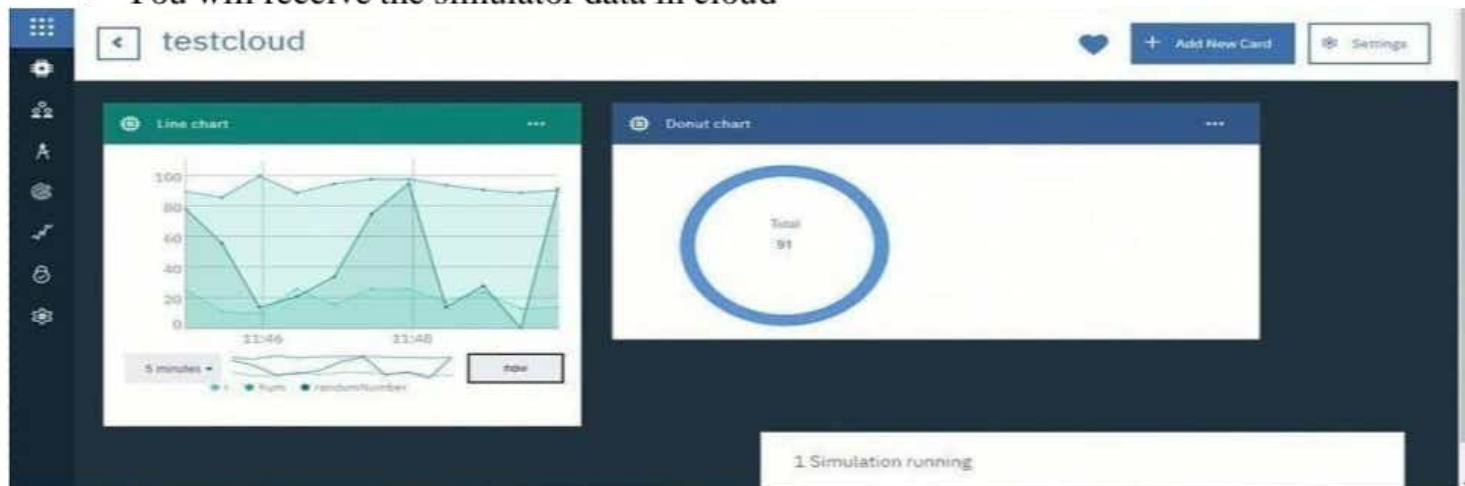
token: **6ogMaaQHNWFEgOD8R?**

Device ID : New

Device Token : 7575

You can see the received data in graphs by creating cards in Boards tab

➤ You will receive the simulator data in cloud



- You can see the received data in Recent Events under your device ➤ Data received in this format(json)

```
{  
  "d": {  
  
    ▪ "name": "NodeMCU",  
    ▪ "temperature": 17,  
    ▪ "humidity": 76,  
    ▪ "Moisture ": 25  
  }  
}
```

Identity Device Information **Recent Events** State Logs

The recent events listed show the live stream of data that is coming and going from this device.

| Event | Value | Format | Last Received |
|------------|-------------------------|--------|-------------------|
| IoT Sensor | {"temp":108,"Humid":84} | json | a few seconds ago |
| IoT Sensor | {"temp":91,"Humid":93} | json | a few seconds ago |
| IoT Sensor | {"temp":106,"Humid":83} | json | a few seconds ago |

items per page: 50 | 1-2 of 2 items

1 of 1 page

Configuration of Node-Red to collect IBM cloud data

The node IBM IOT App In is added to Node-Red workflow. Then the appropriate device credentials obtained earlier are entered into the node to connect and fetch device telemetry to Node-Red.

Node-RED

Flow 1

common

function

IBM IOT App In

Edit IBM IOT in node

Authentication: API Key

API Key: IBM-IOT-APIKEY

Input Type: Device Event

Device Type: All or abcd

Device ID: All or 7654321

Event: All or -

Format: All or json

GoS: 0

Name: IBM IoT

Enabled

debug

```
[{"temp": 94, "Humid": 87}]
6/11/2022, 12:24:48 pm node: e82c5ed2538304e4
v8.25.0 (arm64) 7654321 node: IoT Sensor: IoT app in :
msg.payload: number
94
6/11/2022, 12:24:48 pm node: e82c5ed2538304e4
v8.25.0 (arm64) 7654321 node: IoT Sensor: IoT app in :
msg.payload: number
87
6/11/2022, 12:24:50 pm node: e82c5ed2538304e4
v8.25.0 (arm64) 7654321 node: IoT Sensor: IoT app in :
msg.payload: Object
{"temp": 106, "Humid": 78}
6/11/2022, 12:24:50 pm node: e82c5ed2538304e4
v8.25.0 (arm64) 7654321 node: IoT Sensor: IoT app in :
msg.payload: number
106
6/11/2022, 12:24:50 pm node: e82c5ed2538304e4
v8.25.0 (arm64) 7654321 node: IoT Sensor: IoT app in :
msg.payload: number
78
```

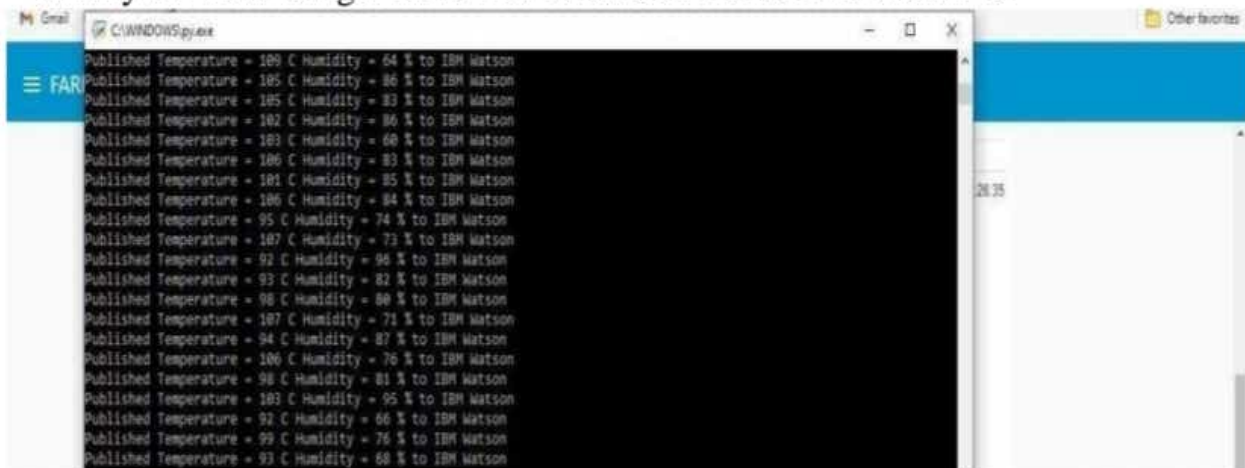
Once it is connected Node-Red receives data from the device Display the data using debug node for verification

Connect function node and write the Java script code to get each reading separately.

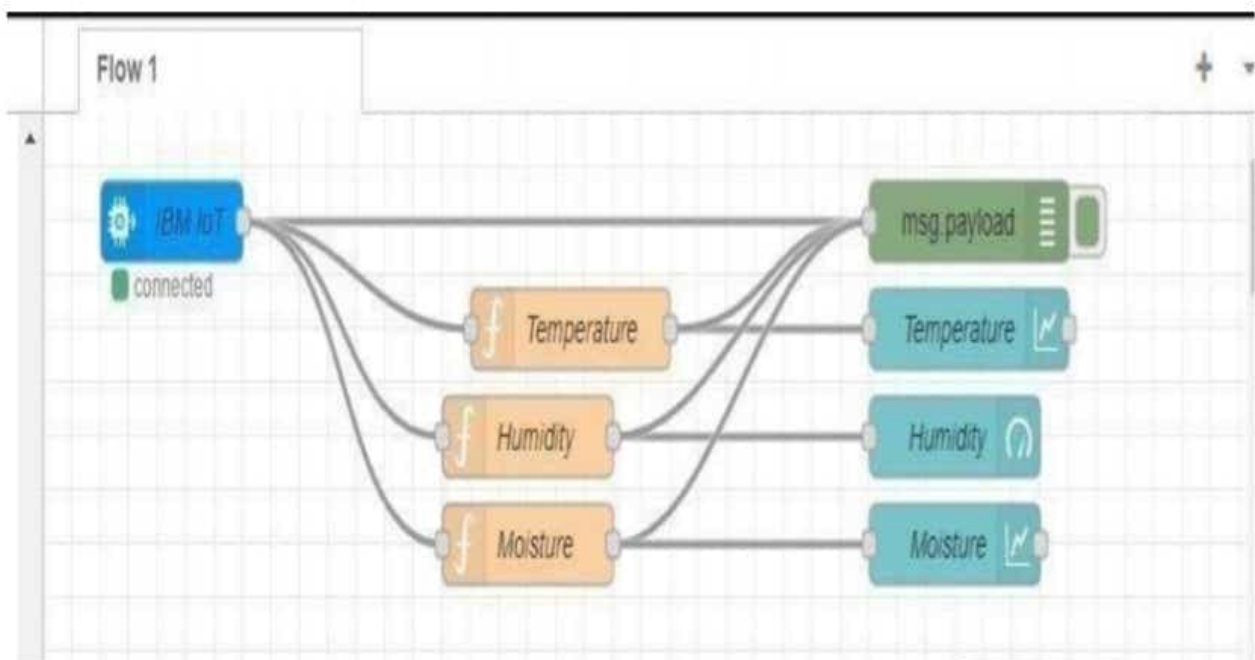
The Java script code for the function node is:

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

Finally connect Gauge nodes from dashboard to see the data in UI



Data received from the cloud in Node-Red console



Nodes connected in following manner to get each reading separately

The screenshot displays the Node-RED interface with the 'Edit chart node' panel open for a 'Temperature' chart. The configuration includes:

- Label:** Temperature
- Type:** Line chart
- X-axis:** last 5 minute
- X-axis Label:** Ht mms/s
- Y-axis:** min (empty) max (empty)
- Legend:** None
- Series Colours:** A grid of color options is shown, with a yellow color selected.

The background shows the main Node-RED workspace with various nodes and a dashboard panel on the right.

This is the Java script code I written for the function node to get Temperature separately.

Configuration of Node-Red to collect data from Open Weather

The Node-Red also receive data from the Open Weather API by HTTP GET request. An inject trigger is added to perform HTTP request for every certain interval. HTTP request node is configured with URL we saved before in section 4.4
The

data we receive from Open Weather after request is in below JSON

```
format:{"coord":{"lon":79.85,"lat":14.13},"weather":[{"id":803,"main":"Clouds","description":"brokenclouds","icon":"04n"}],"base":"stations","main":{"temp":307.59,"feels_like":305.5,"temp_min":307.59,"temp_max":307.59,"pressure":1002,"humidity":35,"sea_level":1002,"grnd_level":1000},"wind":{"speed":6.23,"deg":170},"clouds":{"all":68},"dt":1589991979,"sys":{"country":"IN","sunrise":1589933553,"sunset":1589979720},"timezone":19800,"id":1270791,"name":"Gūdūr","cod":200}
```

In order to parse the JSON string we use Java script functions and get each parameters

```
var temperature = msg.payload.main.temp;  
temperature = temperature-273.15; return  
{payload : temperature.toFixed(2)};
```

In the above Java script code we take temperature parameter into a new variable and convert it from kelvin to Celsius

Then we add Gauge and text nodes to represent data visually in UI

