

SPRINT DELIVERY – 2

Team ID	PNT2022TMID14879
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-**

f5rg4qxd3 Device type: **NodeMCU**

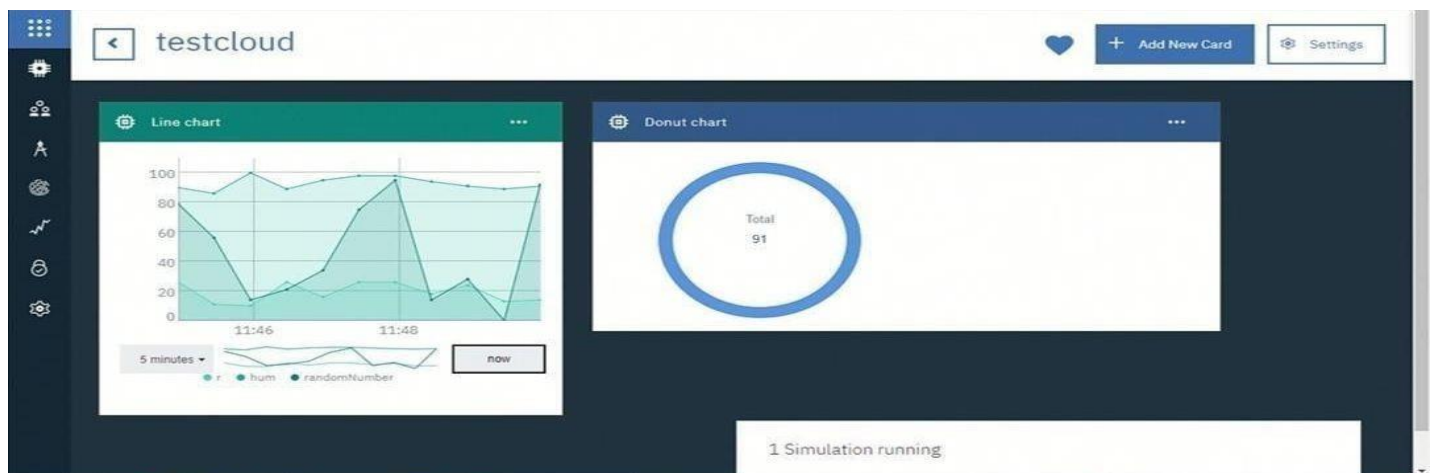
token: **6ogMaaQHNWFEgOD8R?**

Device ID : **1234**

Device Token : **87654321**

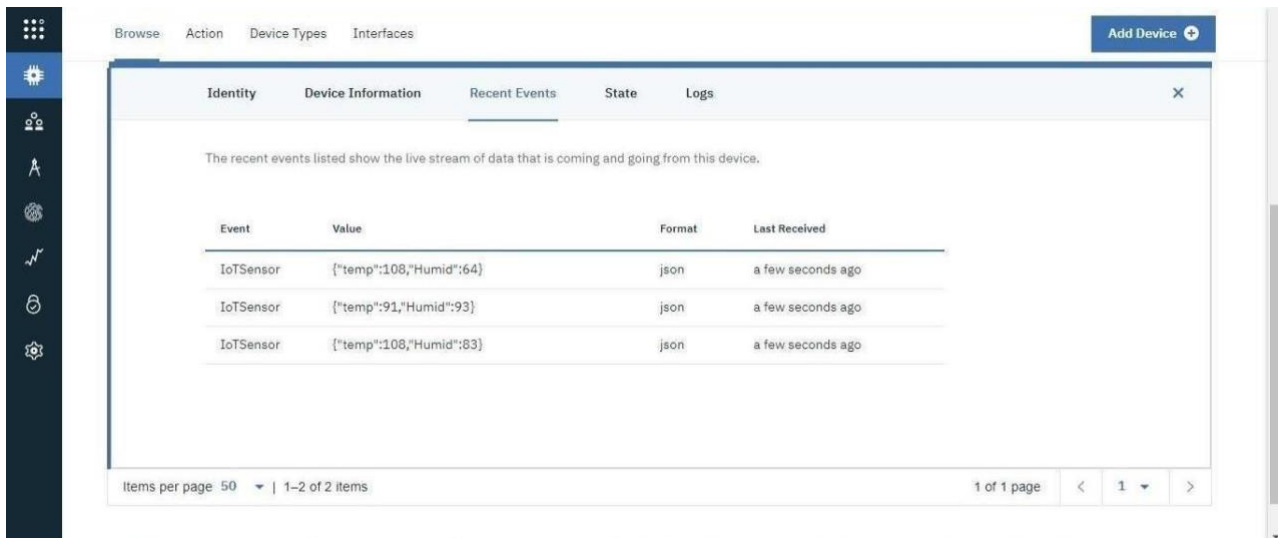
You can see the received data in graphs by creating cards in Boards tab

➤ 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  
  }  
}
```



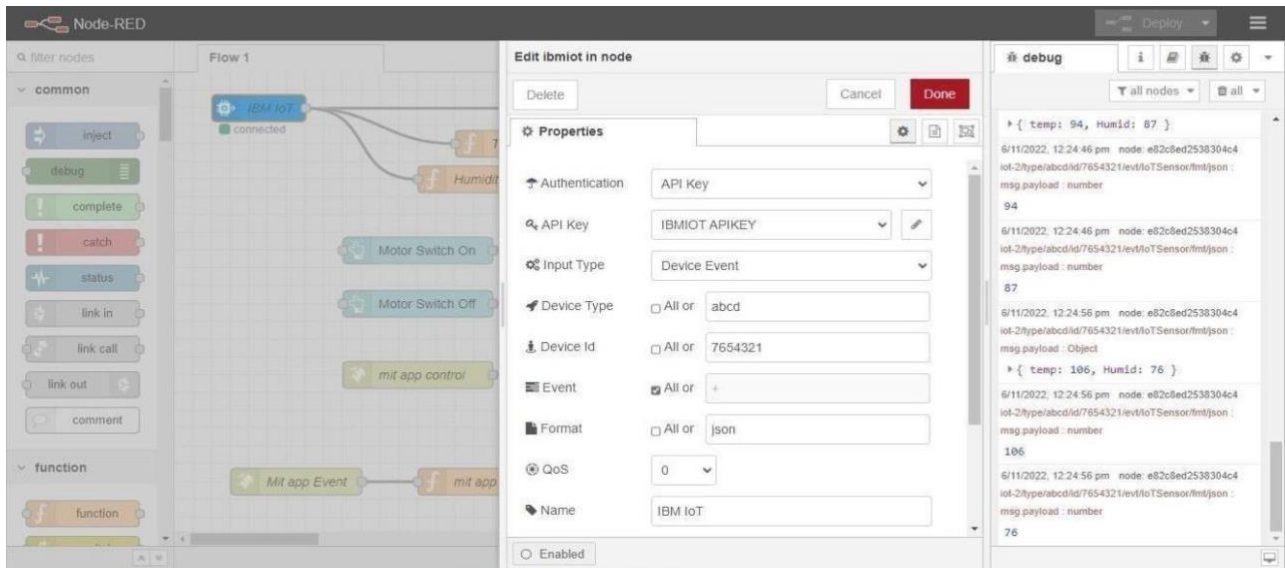
The screenshot shows the 'Recent Events' tab in the IBM IoT Dashboard. It displays a table of recent events received from a device. The table has four columns: Event, Value, Format, and Last Received. There are three rows of data, all from an 'IoT Sensor' device, showing temperature and humidity readings in JSON format. The events were received a few seconds ago.

Event	Value	Format	Last Received
IoT Sensor	{"temp":108,"Humid":64}	json	a few seconds ago
IoT Sensor	{"temp":91,"Humid":93}	json	a few seconds ago
IoT Sensor	{"temp":108,"Humid":83}	json	a few seconds ago

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

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.



The screenshot shows the Node-RED interface with the 'Edit ibmiot in node' dialog box open. The dialog box contains the following configuration details:

- Authentication:** API Key
- API Key:** IBMIOT APIKEY
- Input Type:** Device Event
- Device Type:** All or abcd
- Device Id:** All or 7654321
- Event:** All or +
- Format:** All or json
- QoS:** 0
- Name:** IBM IoT
- Enabled:** ☒ Enabled

The background shows a Node-RED workflow with an 'IBM IoT' node connected to a 'connected' node, which then triggers a 'Motor Switch On' action.

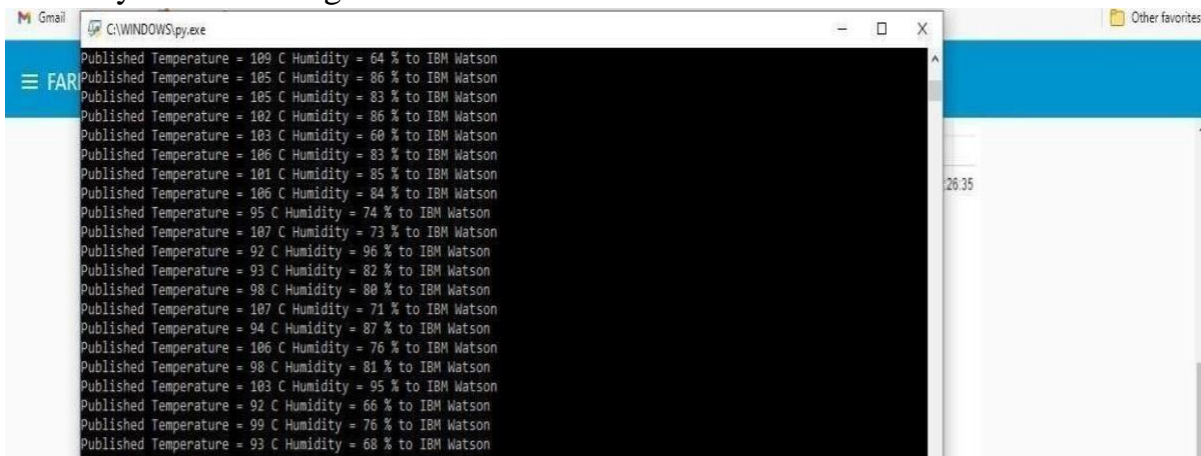
Once it is connected Node-Red receives data from the device Display the data using debug node for verification

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

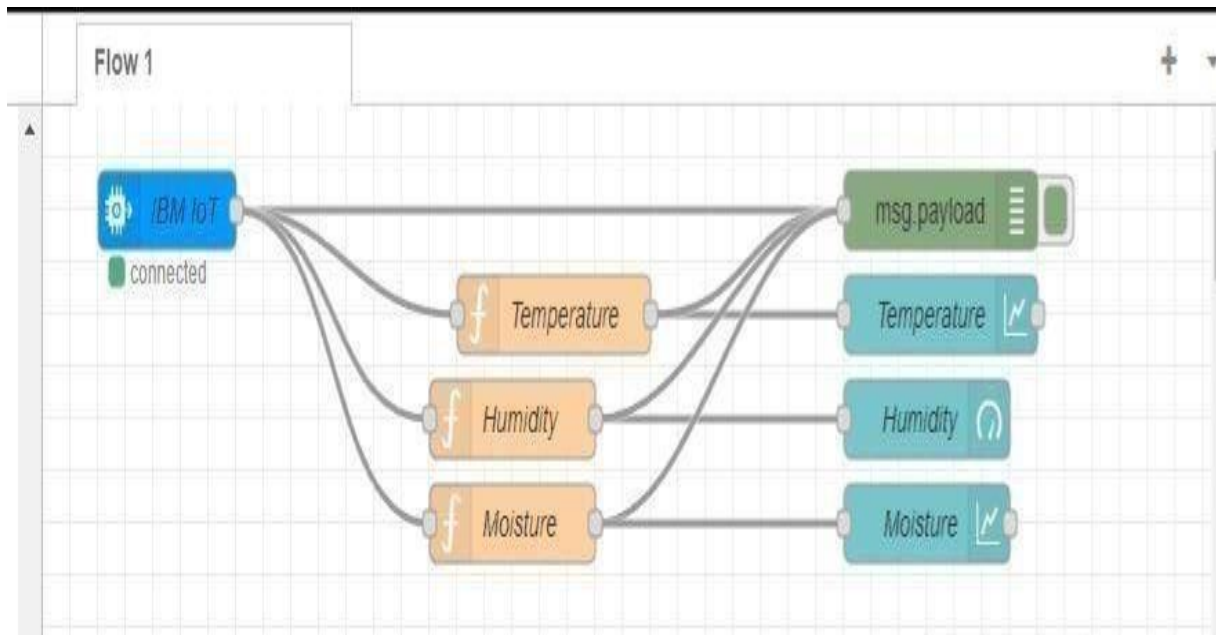
The Java script code for the function node is:

```
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 shows the Node-RED interface with the 'Edit chart node' dialog open for the 'Temperature' chart. The dialog has the following settings:

- I Label:** Temperature
- Type:** Line chart
- X-axis:** last 5 minute
- X-axis Label:** HH:mm:ss
- Y-axis:** min max
- Legend:** None
- Interpolate:** (checked)
- Series Colours:** (various color swatches)

A color picker is also visible, showing the selected color (yellow) and its RGB values (221, 192, 3).

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

Node-RED interface showing a flow editor and a function node configuration.

Flow 1: A flow starting with an **IBM IoT** node (connected) leading to a **function** node.

Edit function node:

- Properties:** Name: Temperature
- On Message:**

```
1 msg.payload=msg.payload.temp
2 global.set("t",msg.payload)
3 return msg;
```

debug console output:

```
{ temp: 107, Humid: 73 }
6/11/2022, 12:23:56 pm node: e82c8ed2538304c4
iot-2/type/abcs/6/7654321evstfTSensor/rmtjson :
msg.payload : number
107
6/11/2022, 12:23:57 pm node: e82c8ed2538304c4
iot-2/type/abcs/6/7654321evstfTSensor/rmtjson :
msg.payload : number
73
6/11/2022, 12:24:06 pm node: e82c8ed2538304c4
iot-2/type/abcs/6/7654321evstfTSensor/rmtjson :
msg.payload : Object
{ temp: 92, Humid: 96 }
6/11/2022, 12:24:06 pm node: e82c8ed2538304c4
iot-2/type/abcs/6/7654321evstfTSensor/rmtjson :
msg.payload : number
92
6/11/2022, 12:24:07 pm node: e82c8ed2538304c4
iot-2/type/abcs/6/7654321evstfTSensor/rmtjson :
msg.payload : number
96
```