

SPRINT DELIVERY – 2

Team ID	PNT2022TMID28757
Project	IoT Enabled Smart Farming Application
Date	14-11-2022

5. Building Project

Connecting IOT Simulator to IBM Watson

IOTPlatform

Open link provided in above section 4.3

Give the credentials of your device in IBM Watson

IOTPlatformClick on connect

My credentials given to simulator are:

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

f5rg4qxp3 Device type:NodeMCU

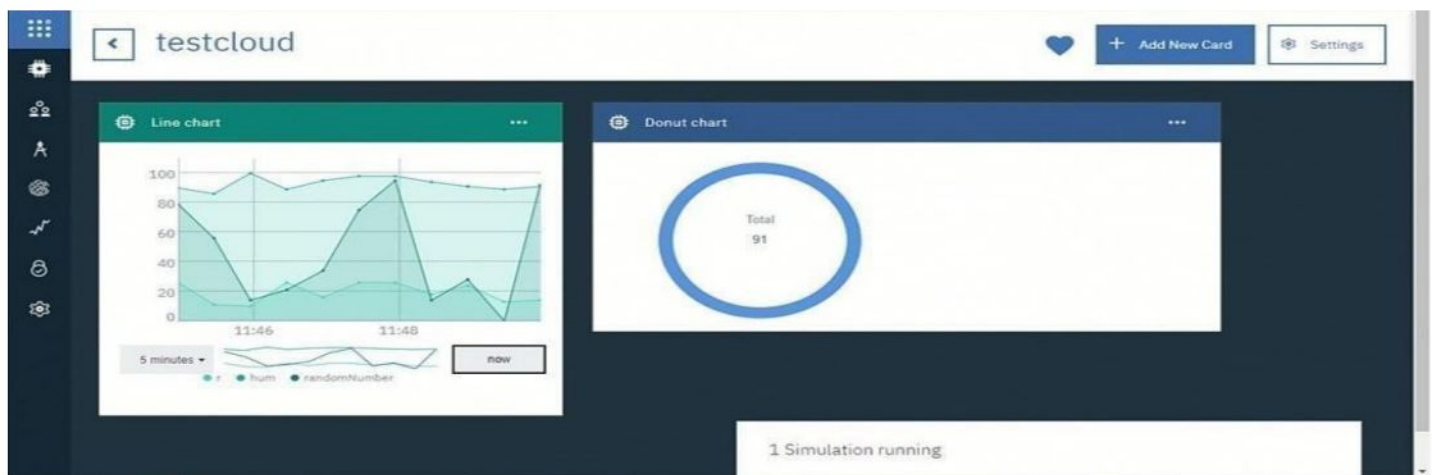
token:**6ogMaaQHNWFEgOD8R?**

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)

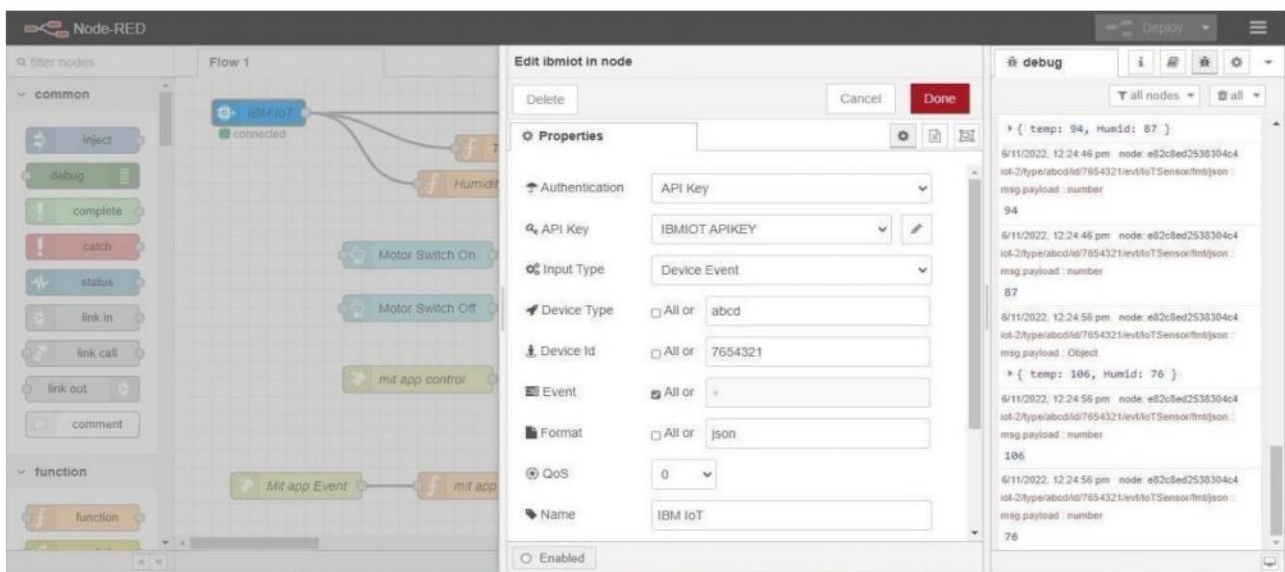
```
{  
  "d": {  
    "name": "NodeMCU",  
    "temperature": 17,  
    "humidity": 76,  
    "Moisture ": 25  
  }  
}
```

The screenshot shows a web dashboard with a dark sidebar on the left containing icons for home, users, a search icon, a globe, a line graph, a lock, and a gear. The main content area has a top navigation bar with tabs: 'Browse', 'Action', 'Device Types', and 'Interfaces'. On the right of this bar is a blue 'Add Device' button with a plus icon. Below the tabs is a modal window with a title bar containing 'Identity', 'Device Information', 'Recent Events' (which is active), 'State', and 'Logs', along with a close button 'X'. Inside the modal, a text message reads: 'The recent events listed show the live stream of data that is coming and going from this device.' Below this is a table with four columns: 'Event', 'Value', 'Format', and 'Last Received'. The table contains three rows of data for 'IoT Sensor' events. At the bottom of the modal, there is a pagination bar showing 'Items per page: 50', a dropdown arrow, and '1-2 of 2 items'. To the right of this are '1 of 1 page', a left arrow, a dropdown showing '1', and a right arrow.

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

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.



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:

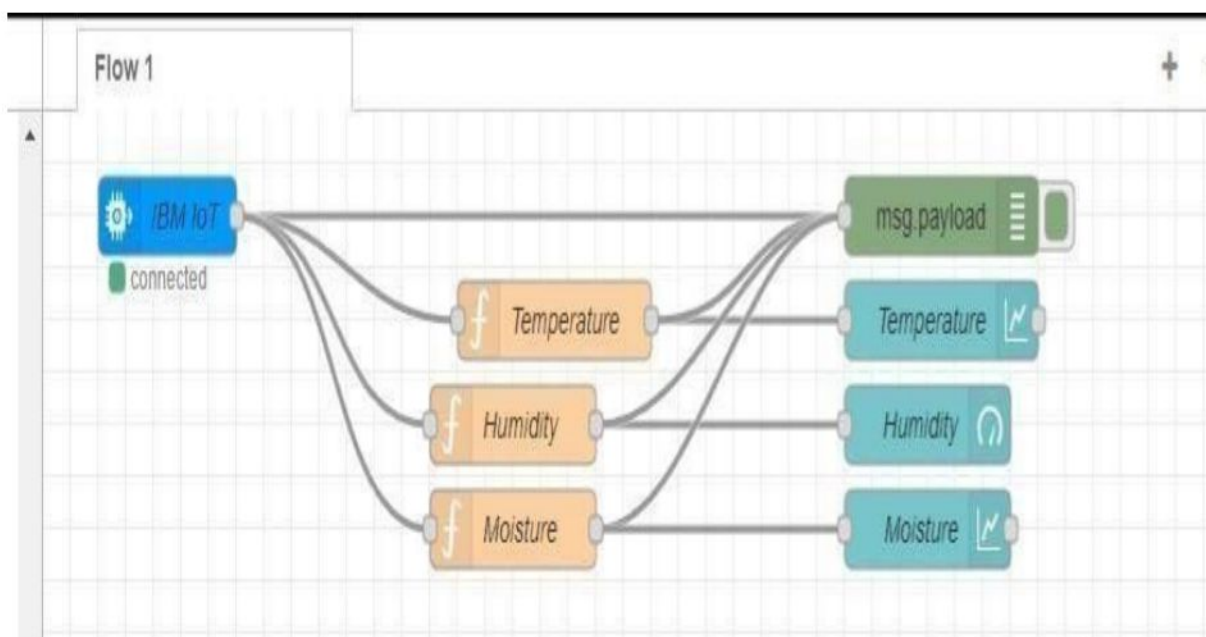
```
msg.payload = msg.payload.d.temperature return
```

```
msg;
```

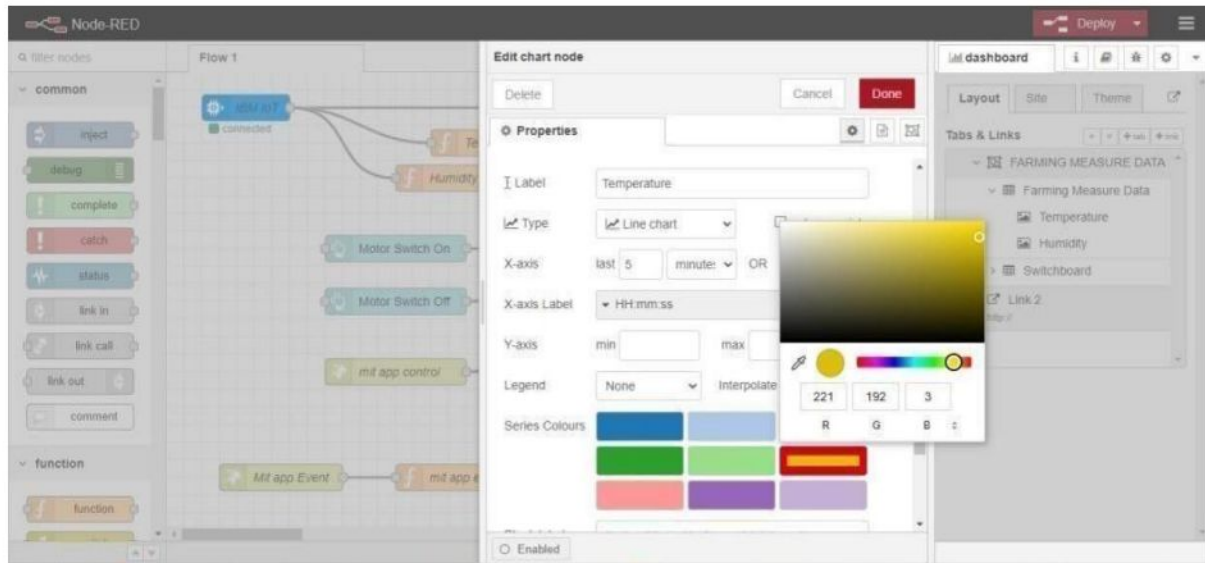
Finally connect Gauge nodes from dashboard to see the data in UI

```
C:\WINDOWS\py.exe
Published Temperature = 109 C Humidity = 64 % to IBM Watson
Published Temperature = 105 C Humidity = 86 % to IBM Watson
Published Temperature = 105 C Humidity = 83 % to IBM Watson
Published Temperature = 102 C Humidity = 86 % to IBM Watson
Published Temperature = 103 C Humidity = 60 % to IBM Watson
Published Temperature = 106 C Humidity = 83 % to IBM Watson
Published Temperature = 101 C Humidity = 85 % to IBM Watson
Published Temperature = 106 C Humidity = 84 % to IBM Watson
Published Temperature = 95 C Humidity = 74 % to IBM Watson
Published Temperature = 107 C Humidity = 73 % to IBM Watson
Published Temperature = 92 C Humidity = 96 % to IBM Watson
Published Temperature = 93 C Humidity = 82 % to IBM Watson
Published Temperature = 98 C Humidity = 80 % to IBM Watson
Published Temperature = 107 C Humidity = 71 % to IBM Watson
Published Temperature = 94 C Humidity = 87 % to IBM Watson
Published Temperature = 106 C Humidity = 76 % to IBM Watson
Published Temperature = 98 C Humidity = 81 % to IBM Watson
Published Temperature = 103 C Humidity = 95 % to IBM Watson
Published Temperature = 92 C Humidity = 66 % to IBM Watson
Published Temperature = 99 C Humidity = 76 % to IBM Watson
Published Temperature = 93 C Humidity = 68 % to IBM Watson
```

Data received from the cloud in Node-Red console



Nodes connected in following manner to get each reading separately



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

