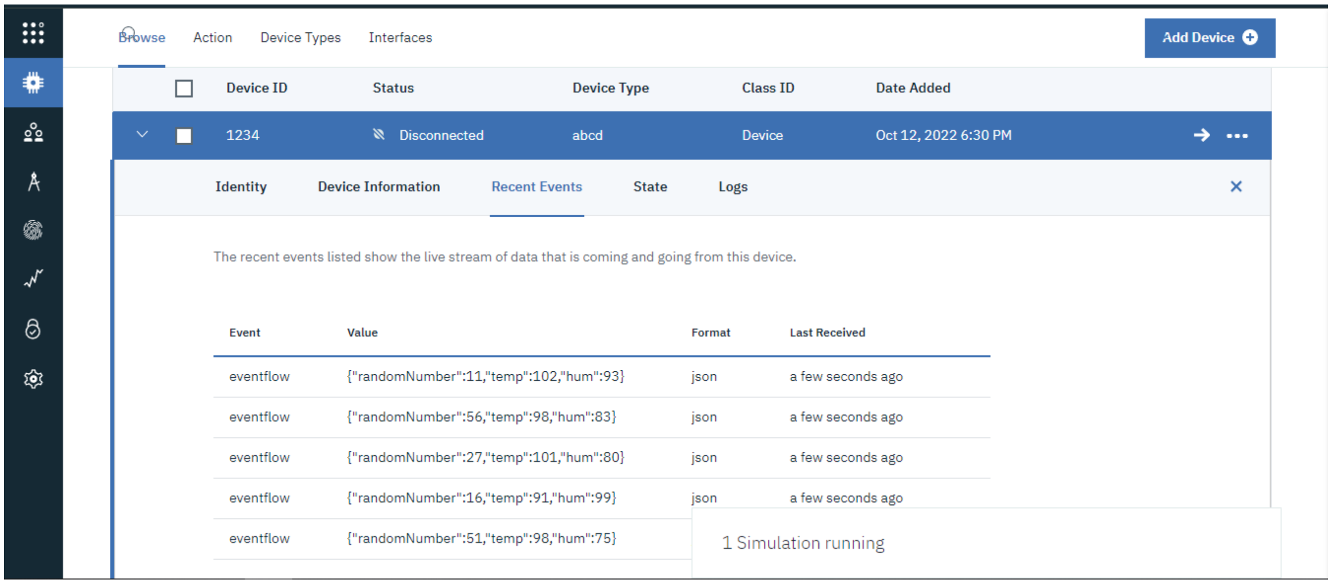
Team ID	**PNT2022TMID35659**
Project Name	**SmartFarmer - IoT Enabled Smart Farming Application**

Sprint Delivery - 2

Building Project

- **Connecting IoT Simulator to IBM Watson IoT Platform**
  - Give the credentials of your device in IBM Watson IoT Platform
  - 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": "abcd",
          “temperature": 17,
          "humidity": 76,
          "Moisture ": 25
     }
}
```



Device ID	Status	Device Type	Class ID	Date Added
1234	Disconnected	abcd	Device	Oct 12, 2022 6:30 PM

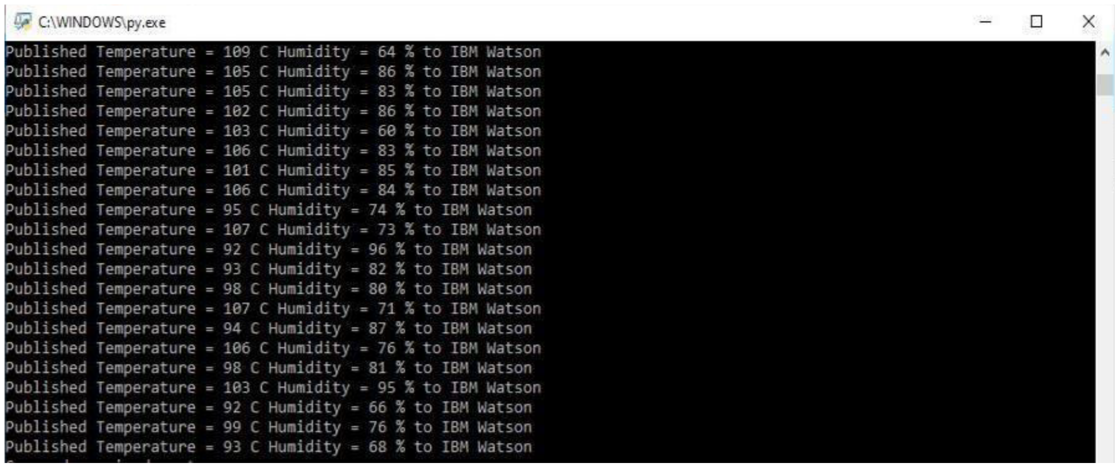
The recent events listed show the live stream of data that is coming and going from this device.

Event	Value	Format	Last Received
eventflow	{"randomNumber":11,"temp":102,"hum":93}	json	a few seconds ago
eventflow	{"randomNumber":56,"temp":98,"hum":83}	json	a few seconds ago
eventflow	{"randomNumber":27,"temp":101,"hum":80}	json	a few seconds ago
eventflow	{"randomNumber":16,"temp":91,"hum":99}	json	a few seconds ago
eventflow	{"randomNumber":51,"temp":98,"hum":75}		

1 Simulation running

- **Configuration of Node-Red to collect IBM cloud data**
  - The node IBM IoT App In is added to the 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 the 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;`

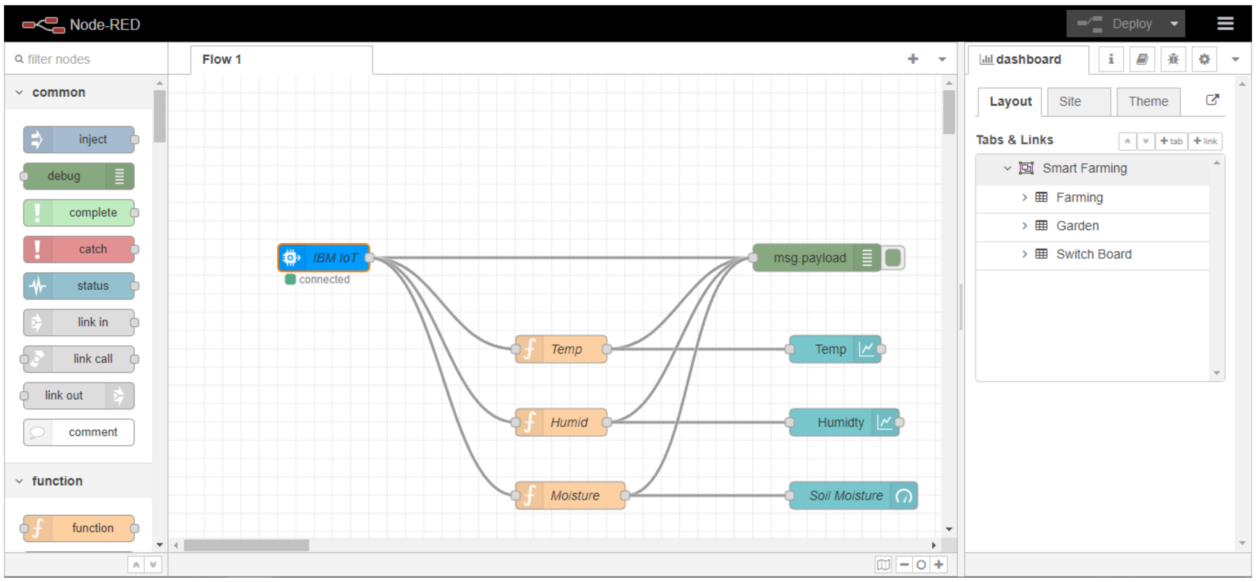
- Finally connect Gauge nodes from dashboard to see the data in UI



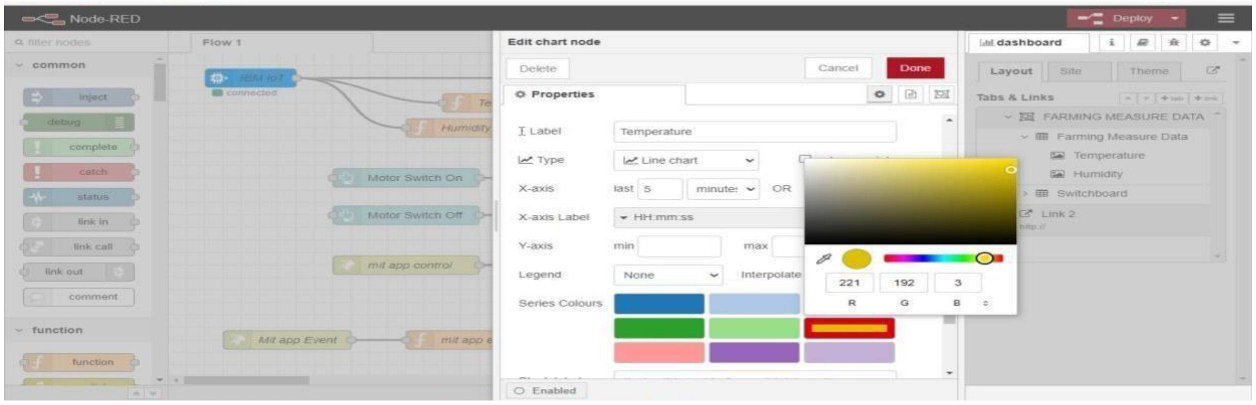
C:\WINDOWS\py.exe

```
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



Label: Temperature
Type: Line chart
X-axis: last 5 minute: OR
X-axis Label: HH:mm:ss
Y-axis: min max
Legend: None Interpolate
Series Colours
R 221 G 192 B 3

- **Configuration of Node-Red to collect data from OpenWeather**
    - The Node-Red also receives data from the OpenWeather API by HTTP GET request.
    - An inject trigger is added to perform HTTP requests for every certain interval.
    - The HTTP request node is configured with the URL we saved before.
    - The data we receive from OpenWeather 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":3
07.59,"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}
```

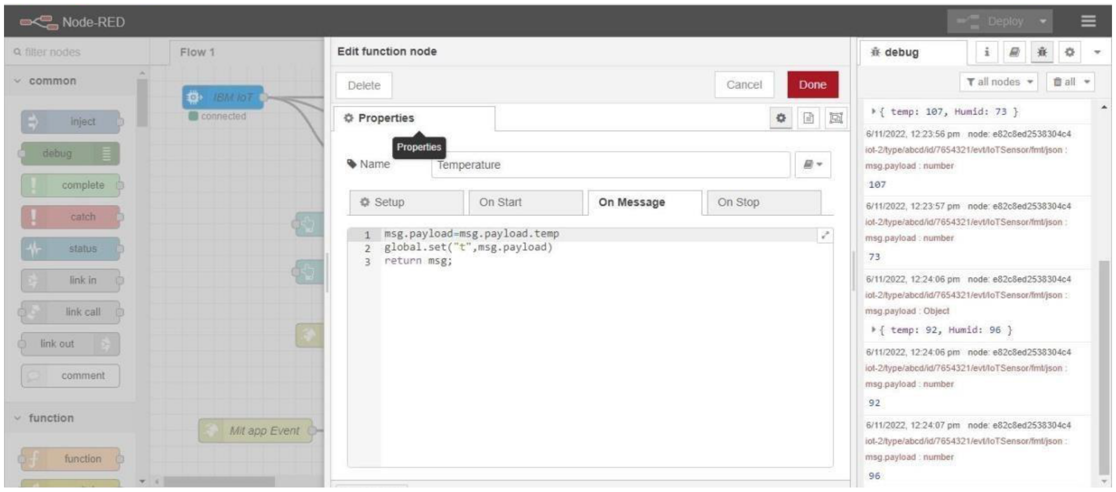
- In order to parse the JSON string we use Java script functions and get each parameters

```
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



Name: Temperature

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

```
▸{ temp: 107, Humid: 73 }
6/11/2022, 12:23:56 pm  node: e82c8ed2538304c4
iot-2/type/abcd/id/7654321/evt/IoTSensor/fmt/json :
msg.payload : number
107
6/11/2022, 12:23:57 pm  node: e82c8ed2538304c4
iot-2/type/abcd/id/7654321/evt/IoTSensor/fmt/json :
msg.payload : number
73
6/11/2022, 12:24:06 pm  node: e82c8ed2538304c4
iot-2/type/abcd/id/7654321/evt/IoTSensor/fmt/json :
msg.payload : Object
▸{ temp: 92, Humid: 96 }
6/11/2022, 12:24:06 pm  node: e82c8ed2538304c4
iot-2/type/abcd/id/7654321/evt/IoTSensor/fmt/json :
msg.payload : number
92
6/11/2022, 12:24:07 pm  node: e82c8ed2538304c4
iot-2/type/abcd/id/7654321/evt/IoTSensor/fmt/json :
msg.payload : number
96
```