

Team ID	PNT2022TMID03184
Project Name	Project – Smart farmer-IoT enabled smart farming application.

Sprint-2

Building Project

Connecting IoT Simulator to IBM Watson IoT Platform

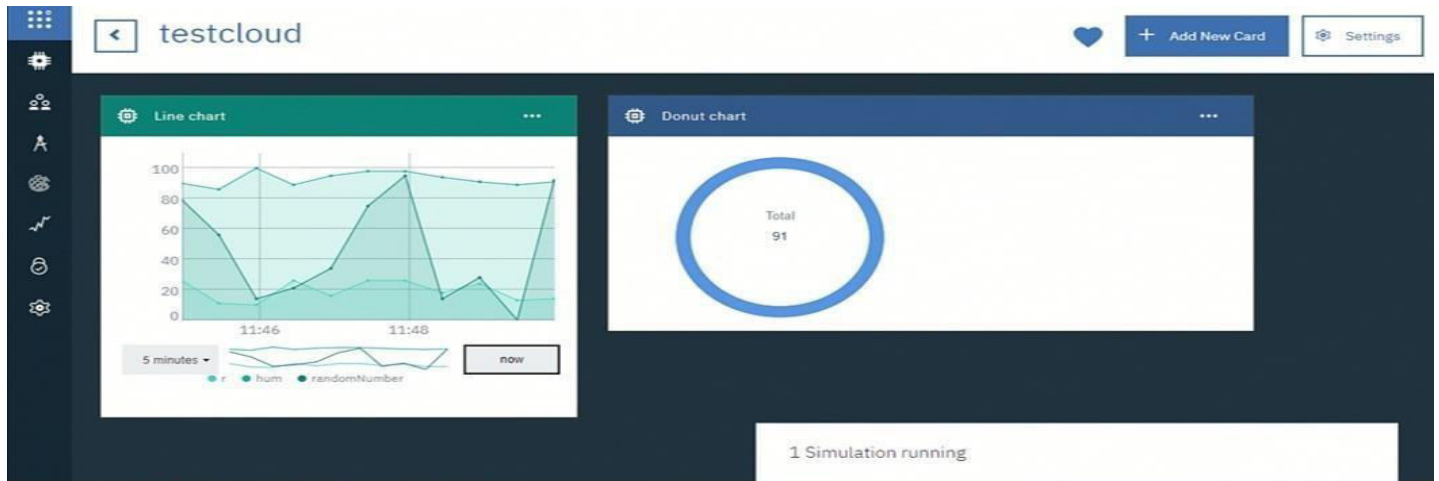
Give the credentials of your device in IBM Watson IoT Platform Click on connect

My credentials given to simulator are:

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
  }
}
```

Browse Action Device Types Interfaces Add Device +

Identity Device Information Recent Events State Logs X

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

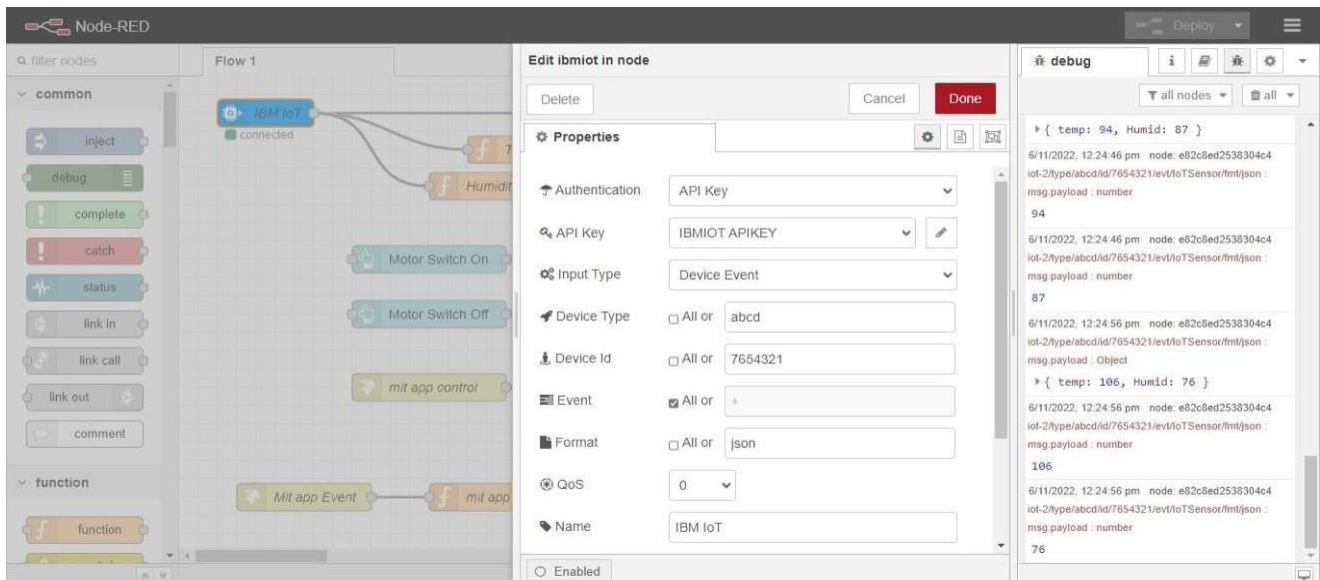
Event	Value	Format	Last Received
IoTSensor	{"temp":108,"Humid":64}	json	a few seconds ago
IoTSensor	{"temp":91,"Humid":93}	json	a few seconds ago
IoTSensor	{"temp":108,"Humid":83}	json	a few seconds ago

Items per page 50 | 1–2 of 2 items 1 of 1 page < 1 >

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 deviceDisplay the data using debug node



for verification

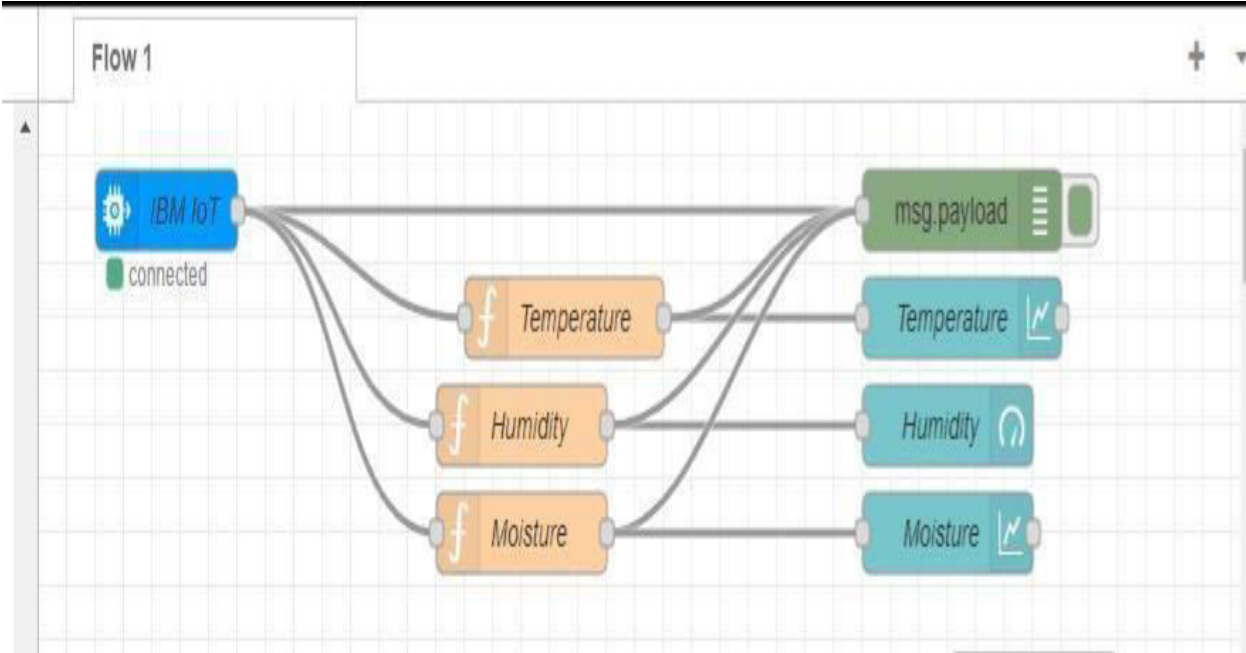
Connect function node and write the Java script code to get each readingseparately.

The Java script code for the function node is: `msg.payload=msg.payload.d.temperature returnmsg;`

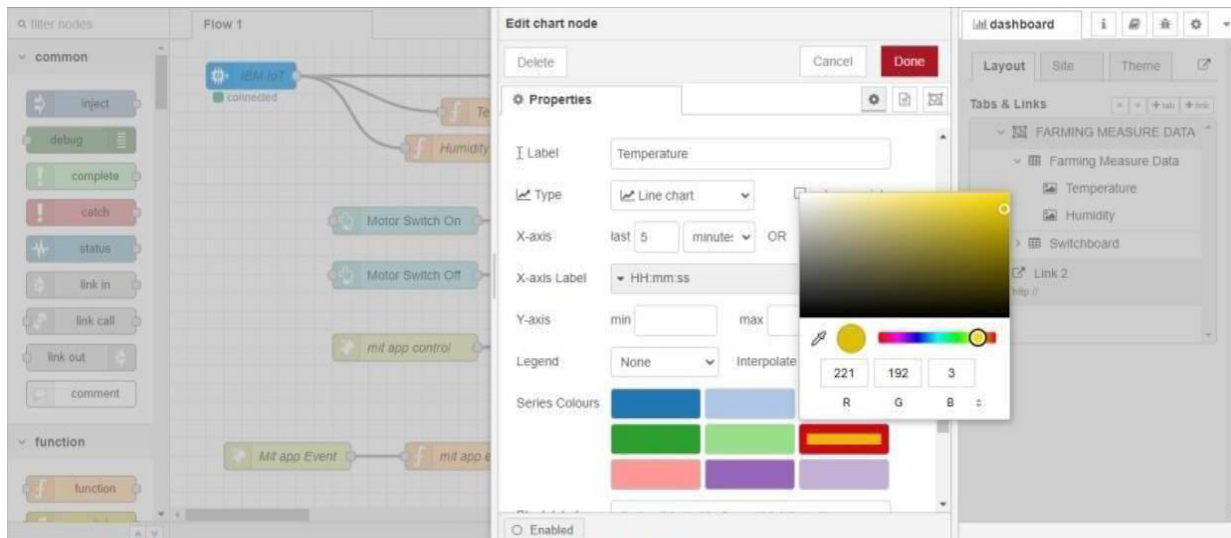
Finally connect Gauge nodes from dashboard to see the data in UI

```
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 Temperatureseparately.
 Configuration of Node-Red to collect data from OpenWeather
 The Node-Red also receive data from the OpenWeather API by HTTP GET request.An inject trigger is added to perform HTTP request for every certain interval.
 HTTP request node is configured.

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

