

IOT ENABLED SMART FARMING APPLICATION

SPRINT DELIVERY – 2

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:

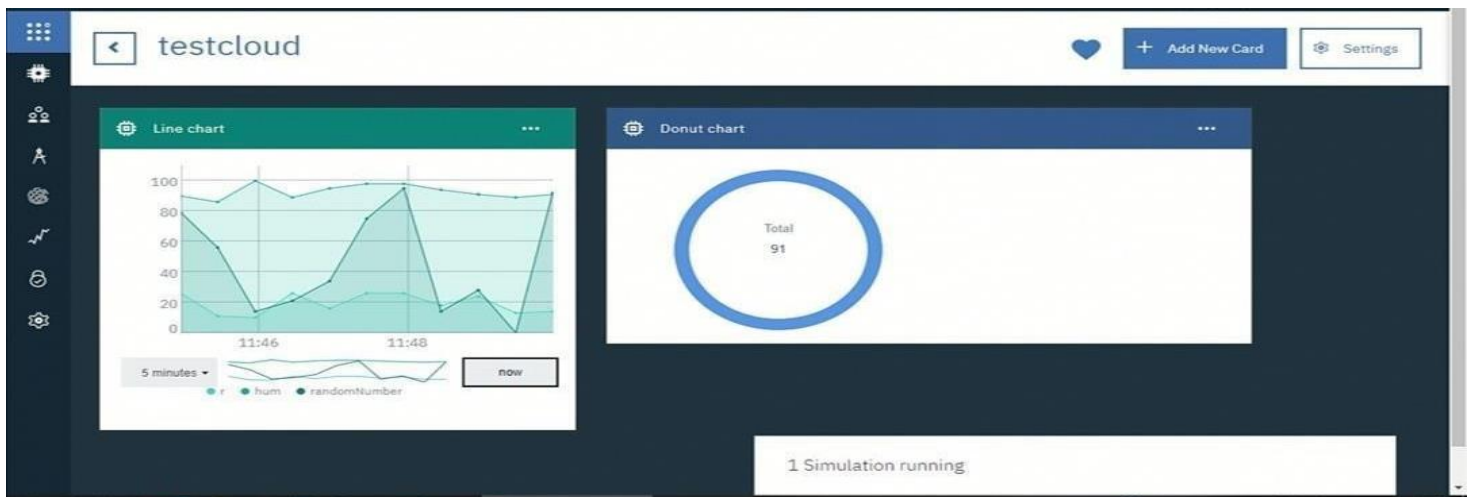
OrgID: **157uf3** api: **a-157uf3-**

f5rg4qxd3 Device type: **abcd** token:

6ogMaaQHWNWFegOD8R?

Device ID : **7654321**

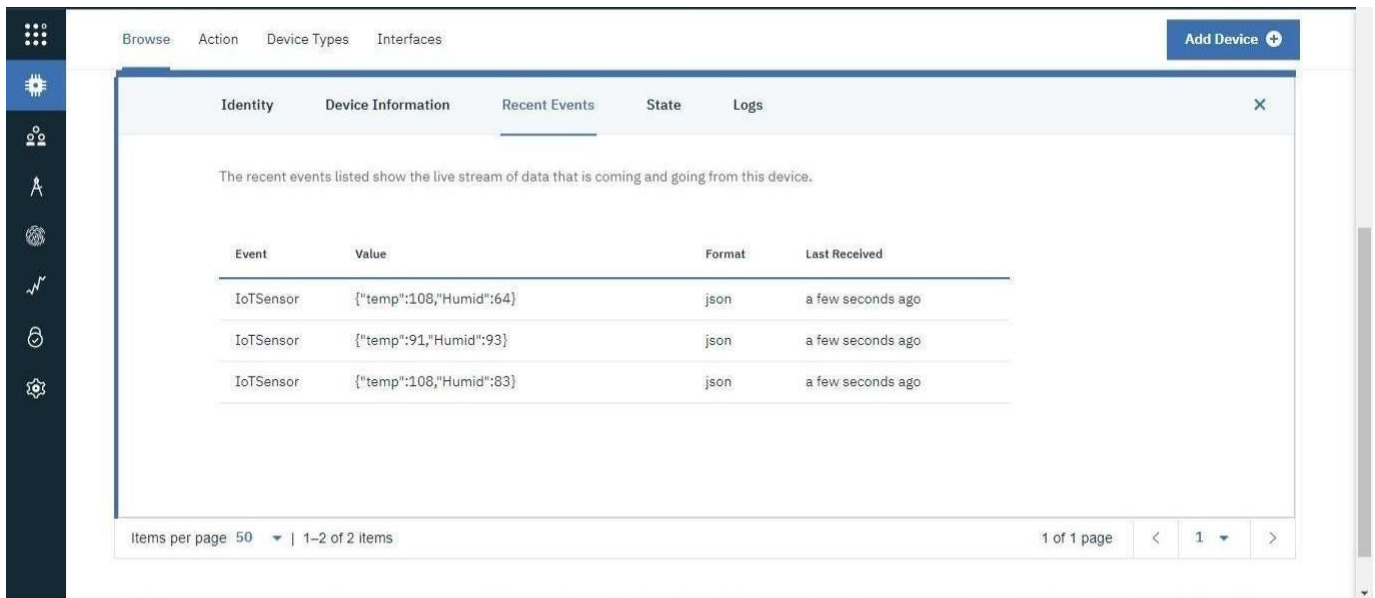
Device Token : **87654321**



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": "abcd",
    "temperature": 17,
    "humidity": 76,
    "Moisture ": 25
  }
}
```

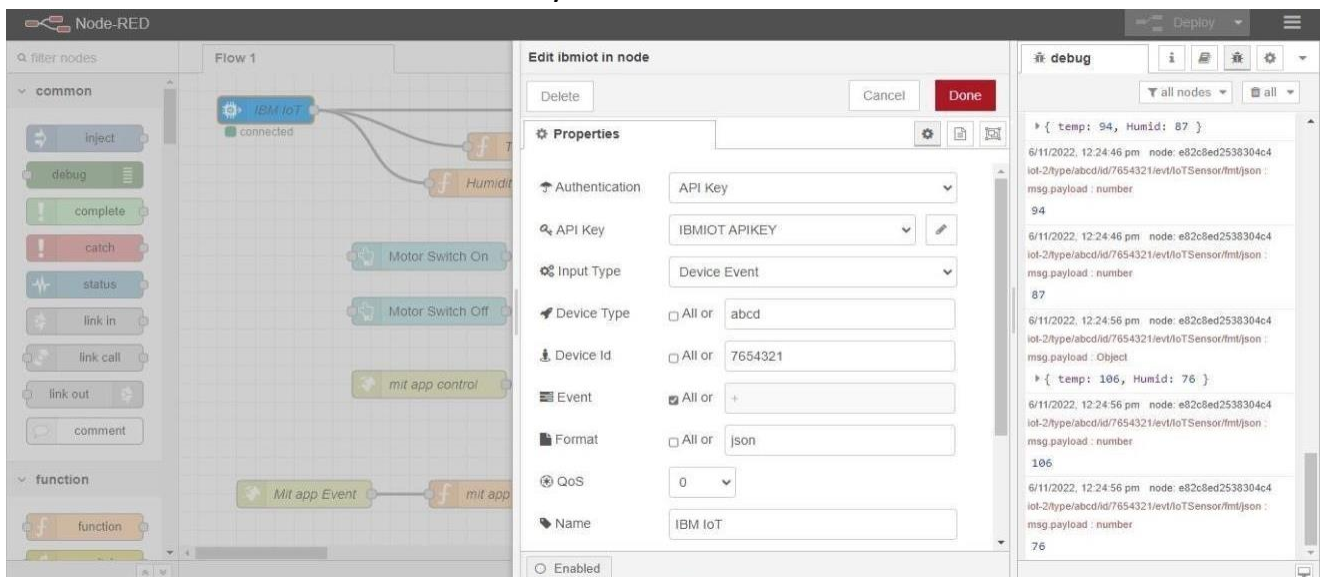


The screenshot shows the IBM IoT Dashboard interface. At the top, there are tabs for 'Browse', 'Action', 'Device Types', and 'Interfaces'. A 'Recent Events' tab is selected, displaying a table of live data streams. The table has four columns: 'Event', 'Value', 'Format', and 'Last Received'. It lists three events from an 'IoT Sensor' device, each providing temperature and humidity data in JSON format. The 'Last Received' column indicates the data was received 'a few seconds ago'. A sidebar on the left contains various icons for navigation. At the bottom of the table, it shows 'Items per page: 50' and '1-2 of 2 items'.

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.



The screenshot displays the Node-RED web interface. On the left, a 'common' node palette is visible. The main workspace shows a flow with an 'IBM IoT' node connected to a 'Humidity' node. A 'debug' node is also present. On the right, the 'Edit ibmiot in node' configuration panel is open, showing fields for 'Authentication' (API Key), 'API Key' (IBMIOT APIKEY), 'Input Type' (Device Event), 'Device Type' (abcd), 'Device Id' (7654321), 'Event' (All or +), 'Format' (All or json), 'QoS' (0), and 'Name' (IBM IoT). The 'Enabled' checkbox is checked. To the right of the configuration panel, a 'debug' console shows a stream of received data, including temperature and humidity values in JSON format.

Once it is connected Node-Red receives data from the device

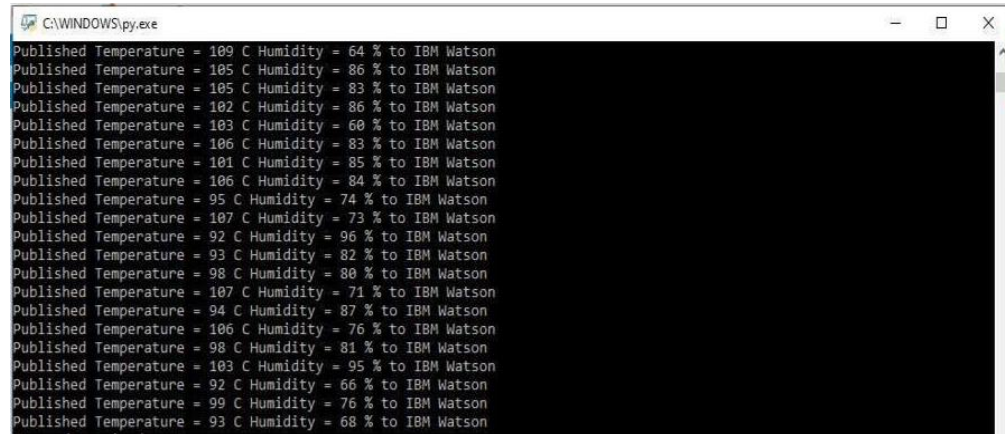
Display the data using debug node for verification

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

The Java script code for the function node is:

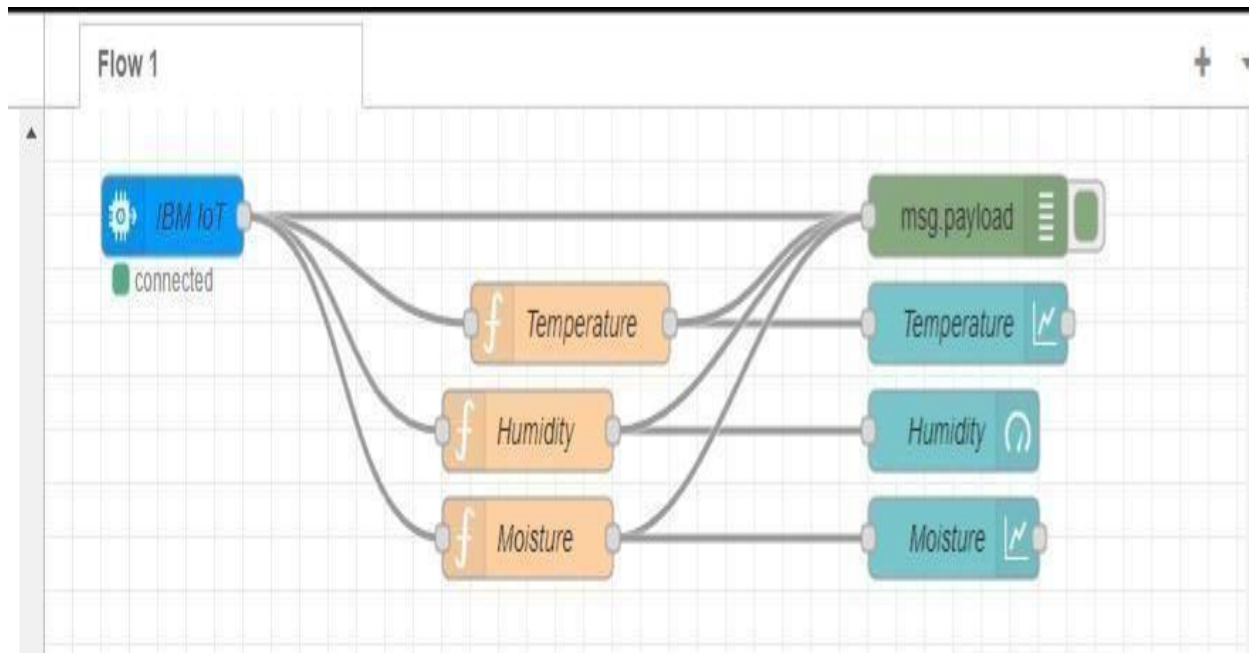
```
msg.payload=msg.payload.d.temperature  
  
return msg;
```

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

The screenshot shows the Node-RED interface with the 'Edit chart node' panel open for the 'Temperature' chart node. The panel includes the following settings:

- Label:** Temperature
- Type:** Line chart
- X-axis:** last 5 minutes
- X-axis Label:** HH:mm:ss
- Y-axis:** min max
- Legend:** None
- Series Colours:** A color palette is visible, showing a gradient from blue to red. The selected color is a yellowish-orange.

The background shows the main Node-RED workspace with the 'Flow 1' diagram and a sidebar with various nodes and a dashboard configuration panel.

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: The flow starts with an **IBM IoT** node (connected) and ends with an **MIT app Event** node.

Edit function node:

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

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

Debug Console:

6/11/2022, 12:23:56 pm node: e82c8ed2538304c4
iot-2/type/abcd/id/7654321/ev/IoTSensor/fmt/json :
msg.payload : number
107

6/11/2022, 12:23:57 pm node: e82c8ed2538304c4
iot-2/type/abcd/id/7654321/ev/IoTSensor/fmt/json :
msg.payload : number
73

6/11/2022, 12:24:06 pm node: e82c8ed2538304c4
iot-2/type/abcd/id/7654321/ev/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/ev/IoTSensor/fmt/json :
msg.payload : number
92

6/11/2022, 12:24:07 pm node: e82c8ed2538304c4
iot-2/type/abcd/id/7654321/ev/IoTSensor/fmt/json :
msg.payload : number
96