

IOT ENABLED SMART FARMINGAPPLICATION

SPRINT DELIVERY – 2

**TEAM ID
:PNT2022TMID44735**

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: **157uf3** api: a-

157uf3- f5rg4qxp3

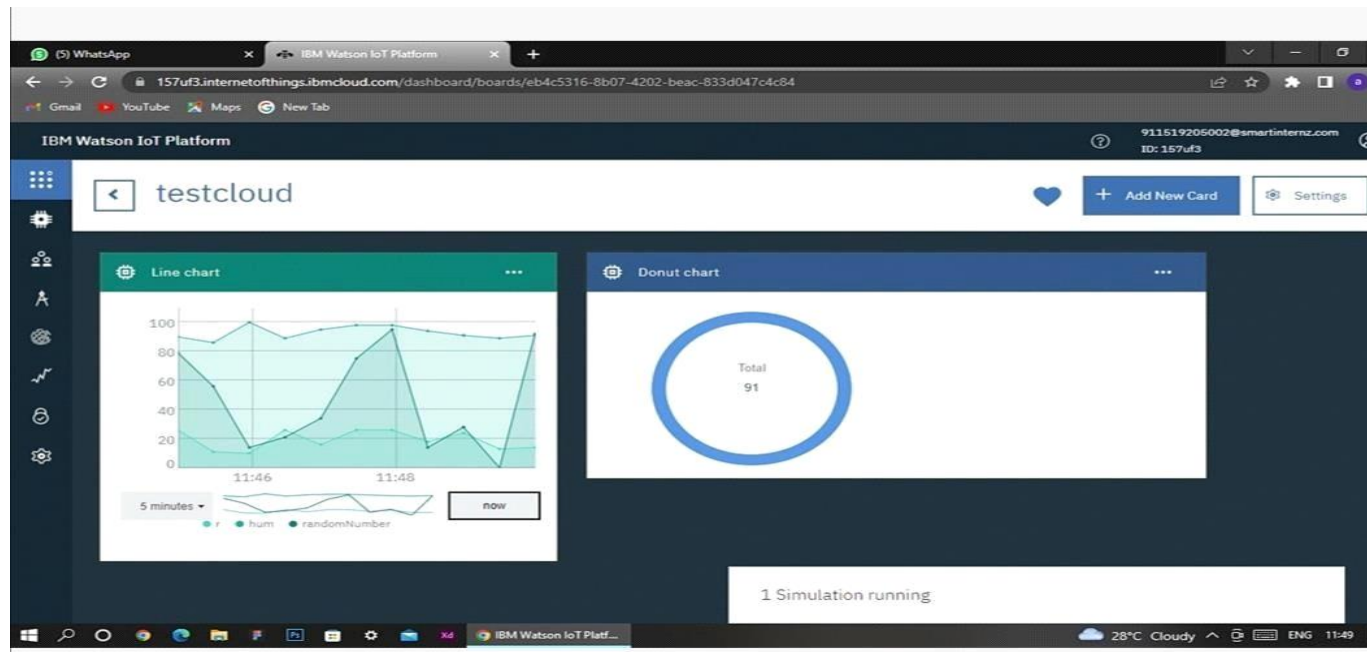
Device type: **abcd**

token:

6ogMaaQHWNWEgO

D8R?

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
  }
}

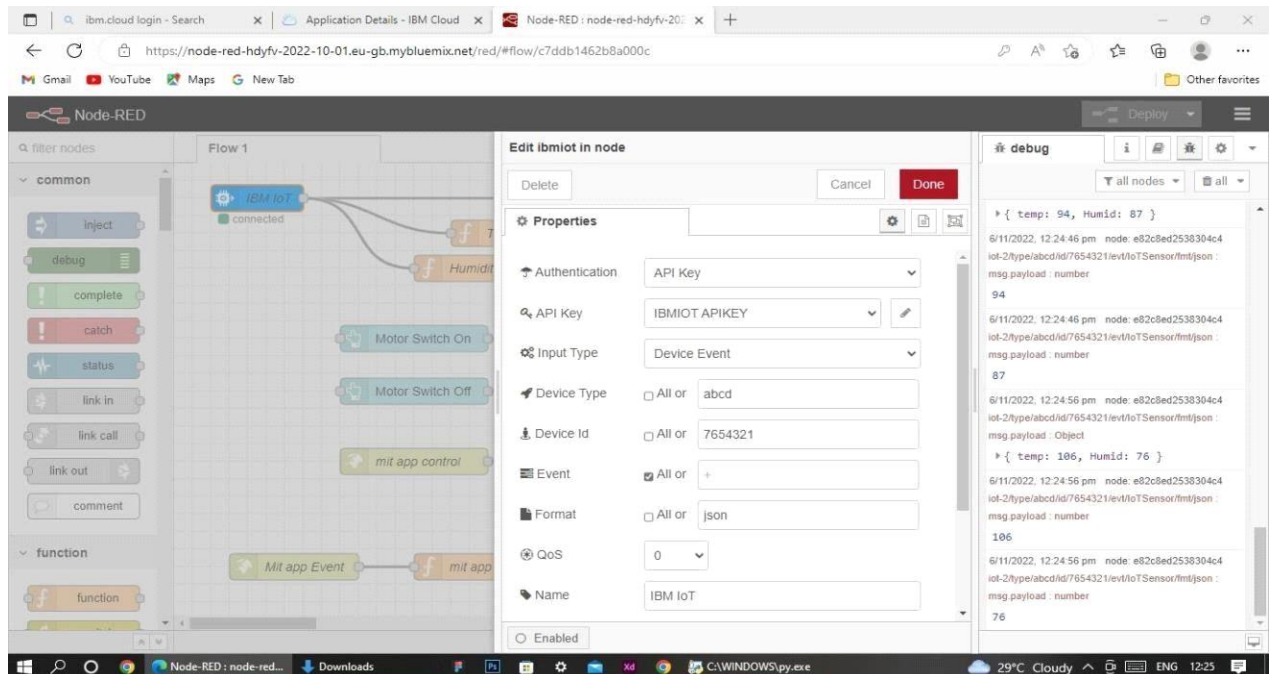
```

The screenshot shows the IBM Watson IoT Platform dashboard. The top navigation bar includes 'Browse', 'Action', 'Device Types', and 'Interfaces'. A sidebar on the left contains icons for various functions. The main content area displays a table of recent events for a 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

At the bottom of the table, it indicates 'Items per page 50' and '1-2 of 2 items'. The page number '1 of 1 page' is also visible.

- 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 JavaScript code to get each reading separately.

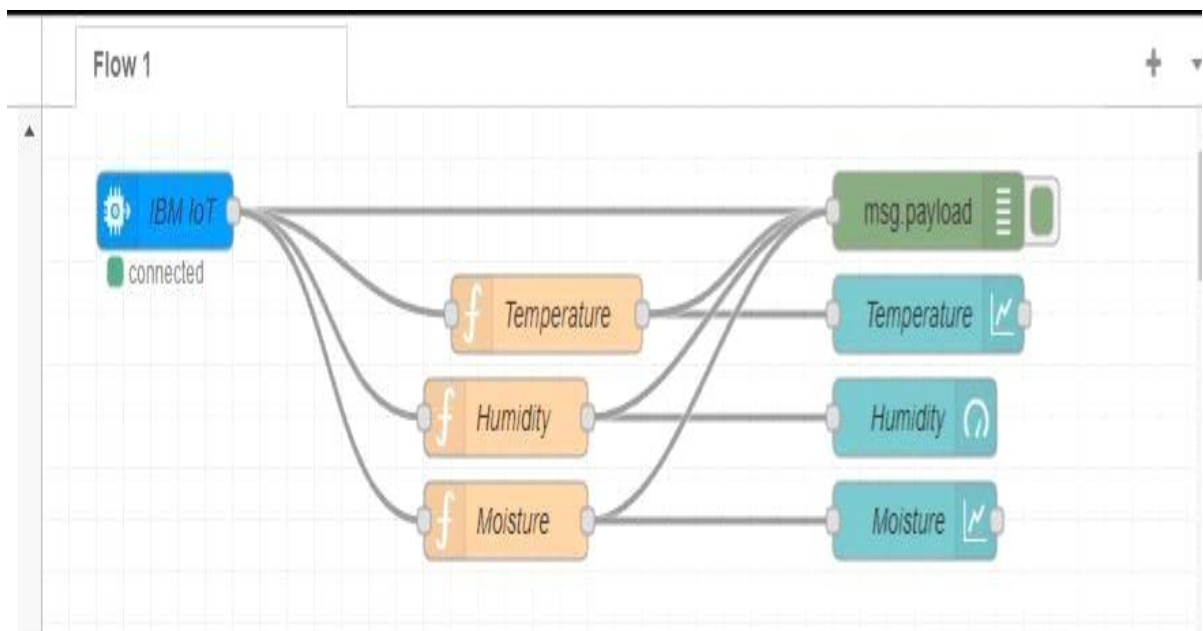
The JavaScript code for the function node is:

```
msg.payload=msg.payload.d.tempera
ture returnmsg;
```

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

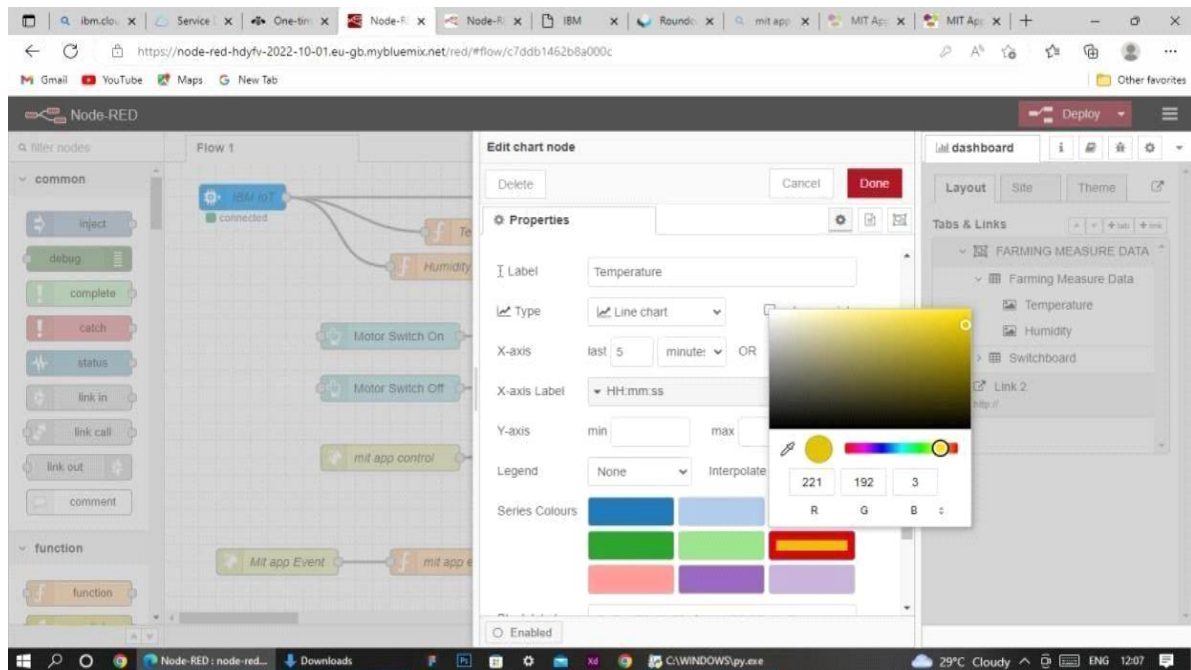
The screenshot shows a web browser with multiple tabs. The active tab is 'Node-RED : node-red-hdyfv-20...', displaying a dashboard at the URL <https://node-red-hdyfv-2022-10-01.eu-gb.mybluemix.net/ui/#/17/socketid=CF6nygqxmjqZo7UAAAP>. A terminal window titled 'C:\WINDOWS\py.exe' is overlaid on the dashboard, showing a list of published sensor data:

```
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 with URL we saved before in section 4.4 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":307.59,"pre
```

```

ssure":1002,"h
umidity":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":"G
ūdūr","cod":20 0}

```

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

