

IOT ENABLED SMART FARMING APPLICATION

SPRINT DELIVERY – 2

5, Building Project

5.1 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:

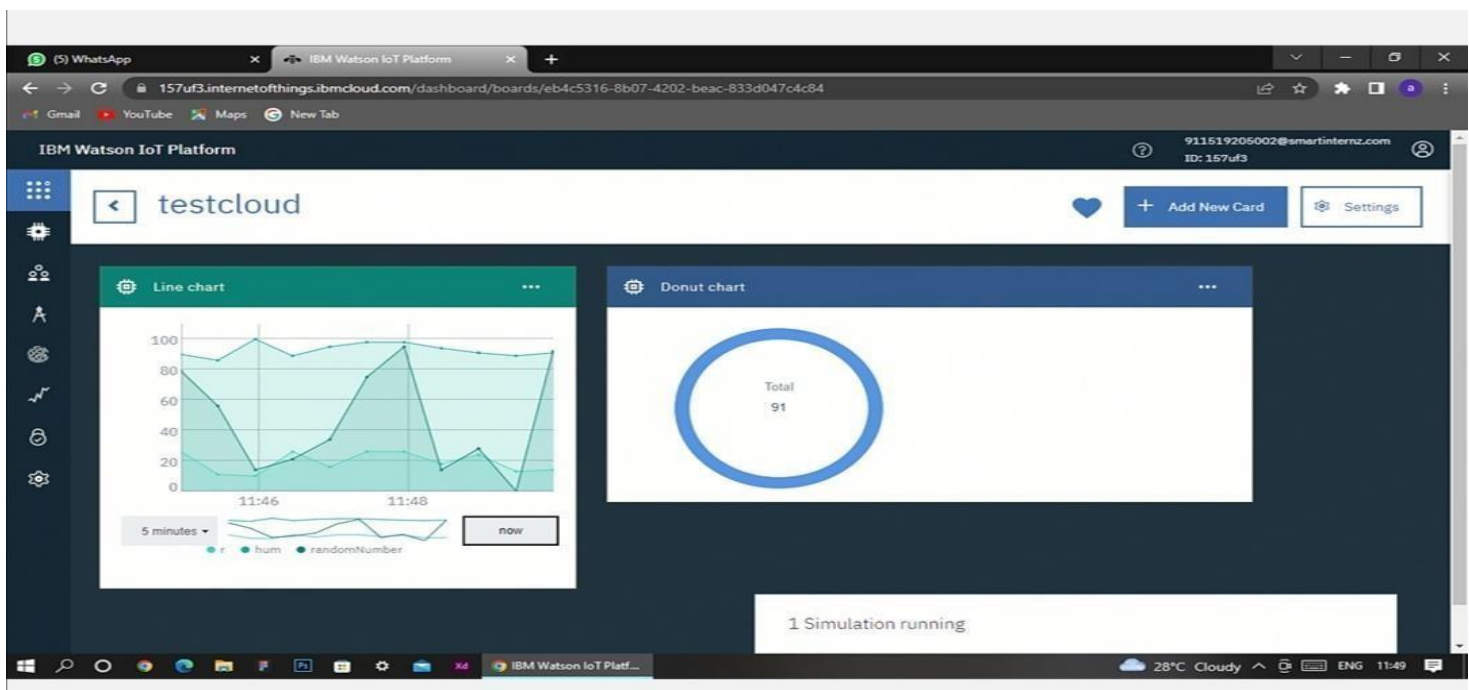
OrgID: **157uf3** api: **a-157uf3-**

f5rg4qxp3 Device type: **abcd** token:

6ogMaaQHNWFEgOD8R?

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)

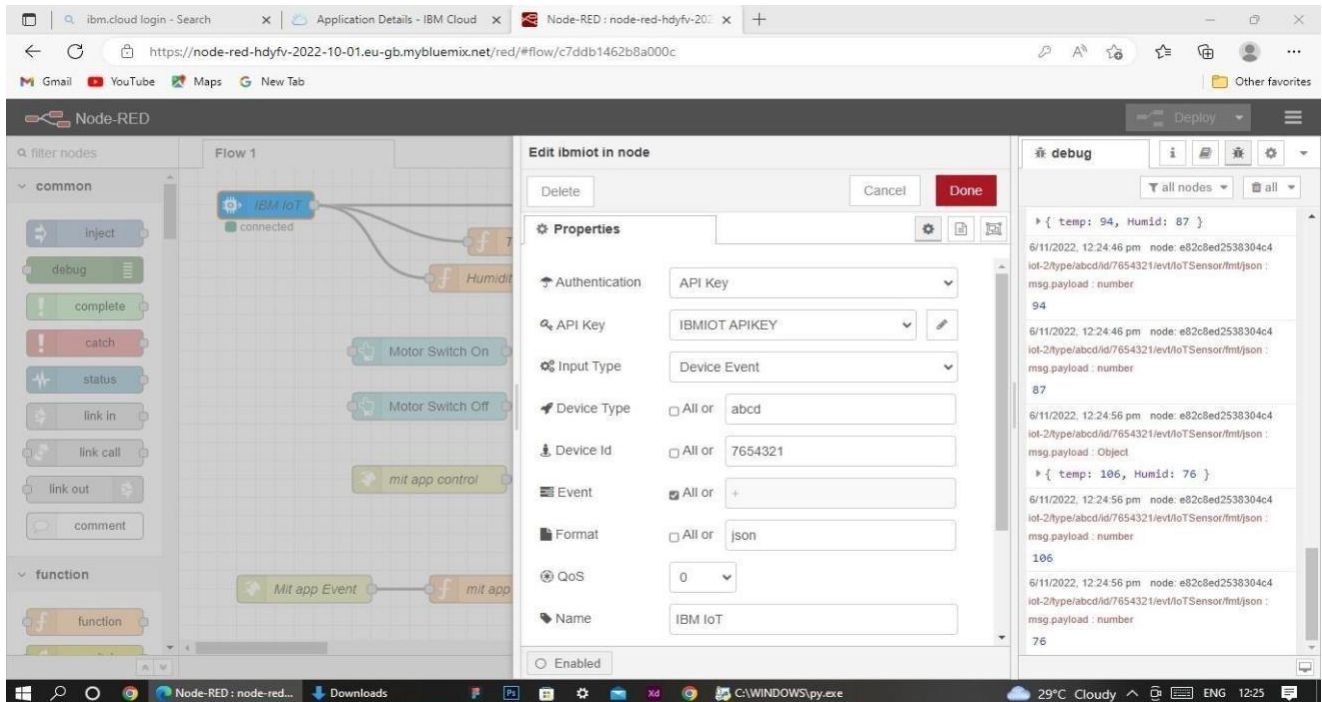
```
{  
  "d": {  
    "name": "abcd",  
    "temperature": 17,  
    "humidity": 76,  
    "Moisture": 25  
  }  
}
```

The screenshot displays the IBM Watson IoT Platform dashboard. The top navigation bar includes tabs for 'Browse', 'Action', 'Device Types', and 'Interfaces'. A sidebar on the left contains various icons for navigation. The main content area shows a modal window titled 'Recent Events' with a table of data. The table has four columns: 'Event', 'Value', 'Format', and 'Last Received'. It lists three events from an 'IoT Sensor' device, each with a JSON value and a timestamp of 'a few seconds ago'. The bottom of the modal shows pagination controls for '1 of 1 page'.

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

5.2 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:

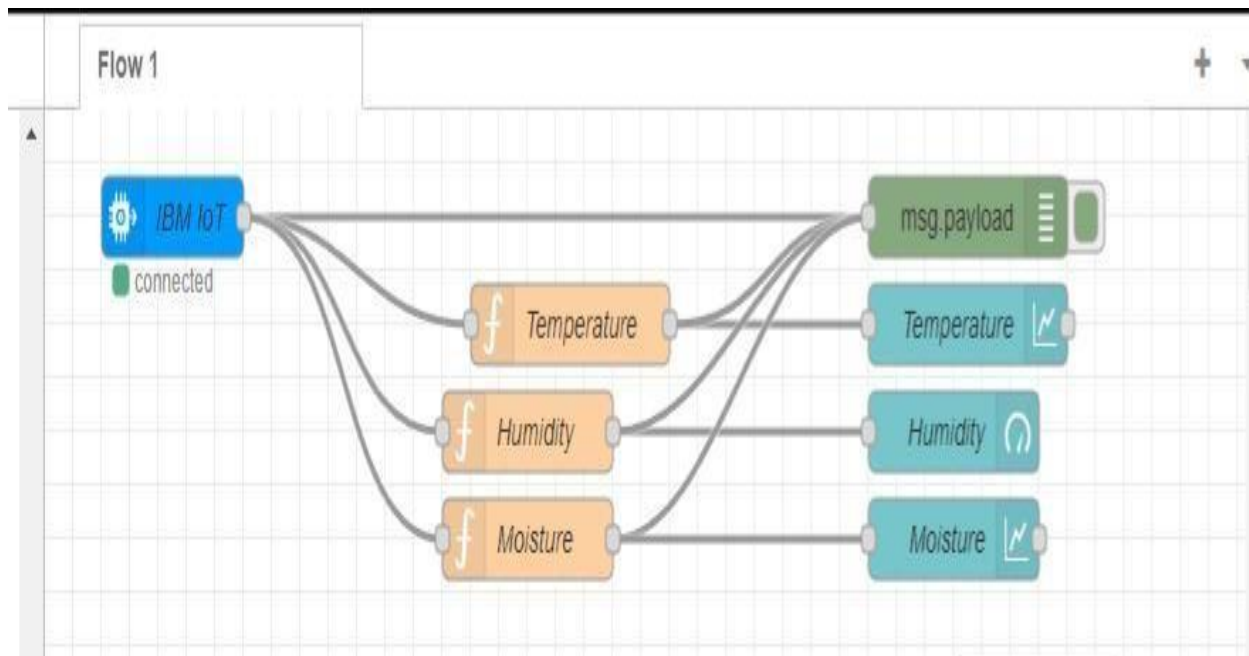
```
msg.payload=msg.payload.d.temperature returnmsg;
```

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

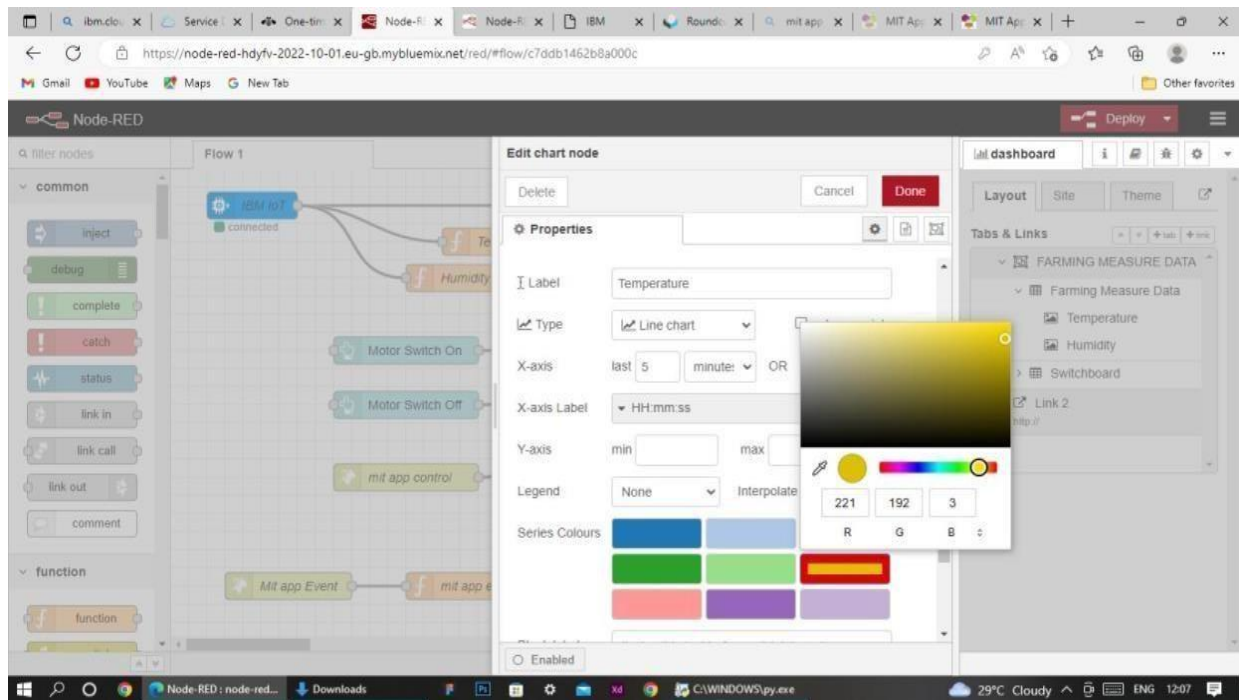
The screenshot shows a web browser window with a Node-RED dashboard. A terminal window titled 'C:\WINDOWS\py.exe' is open, displaying a list of 20 lines of data. Each line represents a published data point with temperature in Celsius and humidity in percent, both sent to IBM Watson. The data is as follows:

Published Temperature (C)	Humidity (%)	Destination
109	64	to IBM Watson
105	86	to IBM Watson
105	83	to IBM Watson
102	86	to IBM Watson
103	60	to IBM Watson
106	83	to IBM Watson
101	85	to IBM Watson
106	84	to IBM Watson
95	74	to IBM Watson
107	73	to IBM Watson
92	96	to IBM Watson
93	82	to IBM Watson
98	80	to IBM Watson
107	71	to IBM Watson
94	87	to IBM Watson
106	76	to IBM Watson
98	81	to IBM Watson
103	95	to IBM Watson
92	66	to IBM Watson
99	76	to IBM Watson
93	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.

5.3 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,"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

