

**IOT ENABLED SMART FARMING  
APPLICATION**

**SPRINT DELIVERY – 2**

**TEAM ID : PNT2022TMID18018**

## 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:

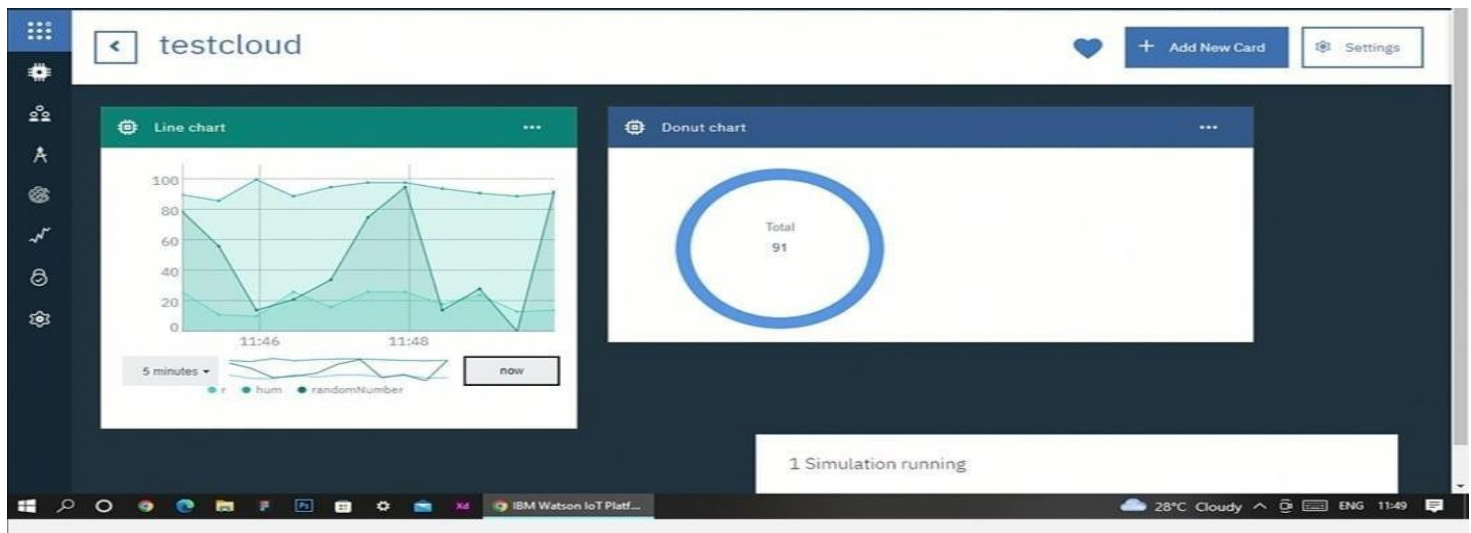
6ogMaaQHNWFEgOD8R?

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  
  }  
}
```

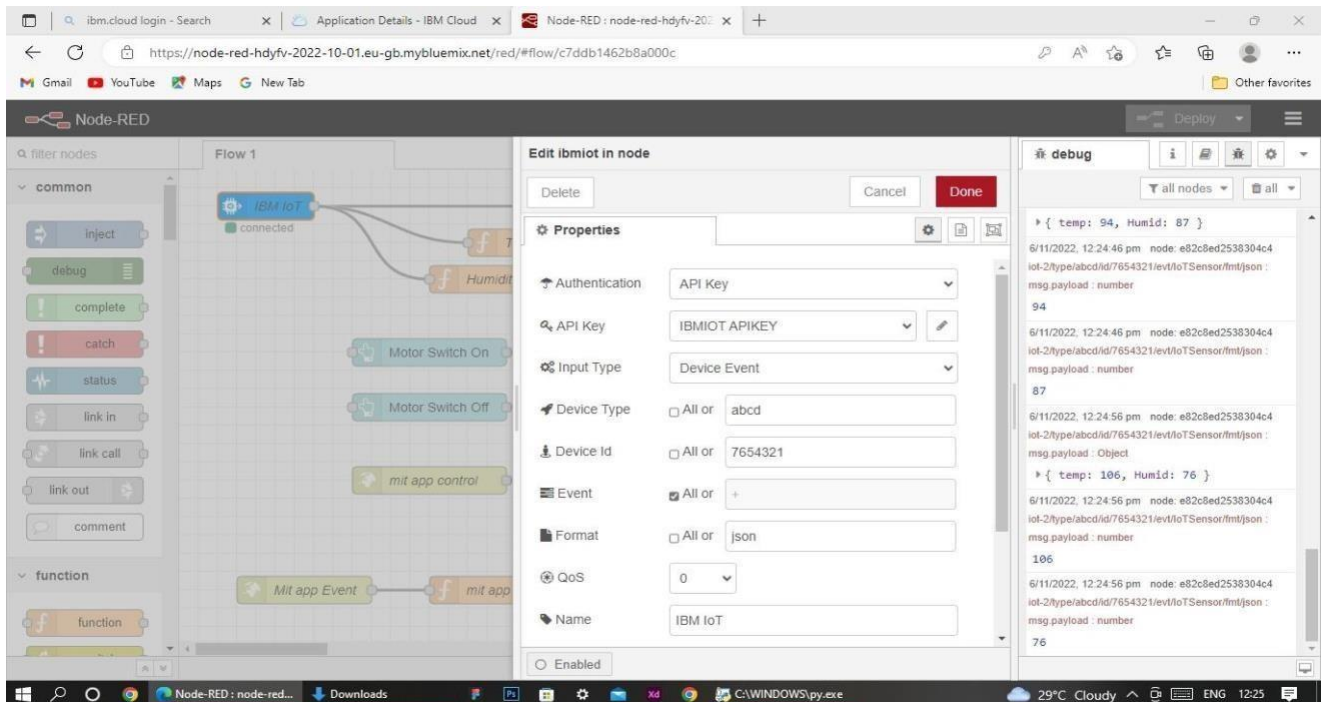
The screenshot displays the IBM Watson IoT Platform interface. The top navigation bar includes 'Browse', 'Action', 'Device Types', and 'Interfaces'. A sidebar on the left contains various icons for navigation. The main content area shows the 'Recent Events' tab for a device, with a sub-header 'Identity'. Below the header, a message states: 'The recent events listed show the live stream of data that is coming and going from this device.' A table lists the events:

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

At the bottom of the table, it says 'Items per page 50 | 1-2 of 2 items'. The right side of the table has pagination controls showing '1 of 1 page'.

## 5.1 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 device

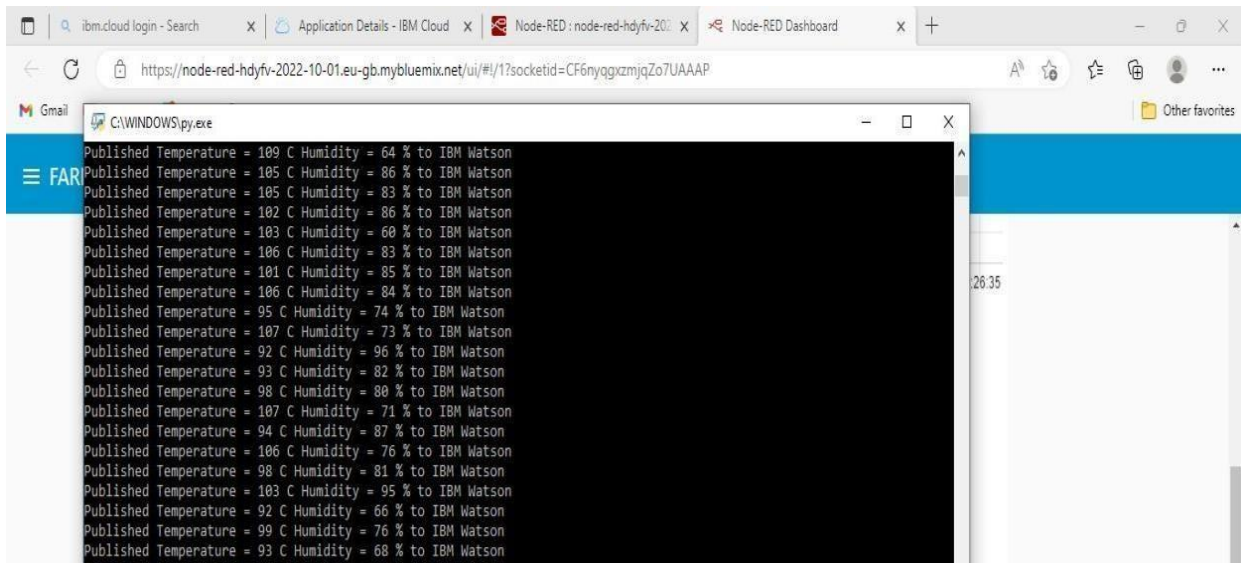
Display the data using debug node for verification

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

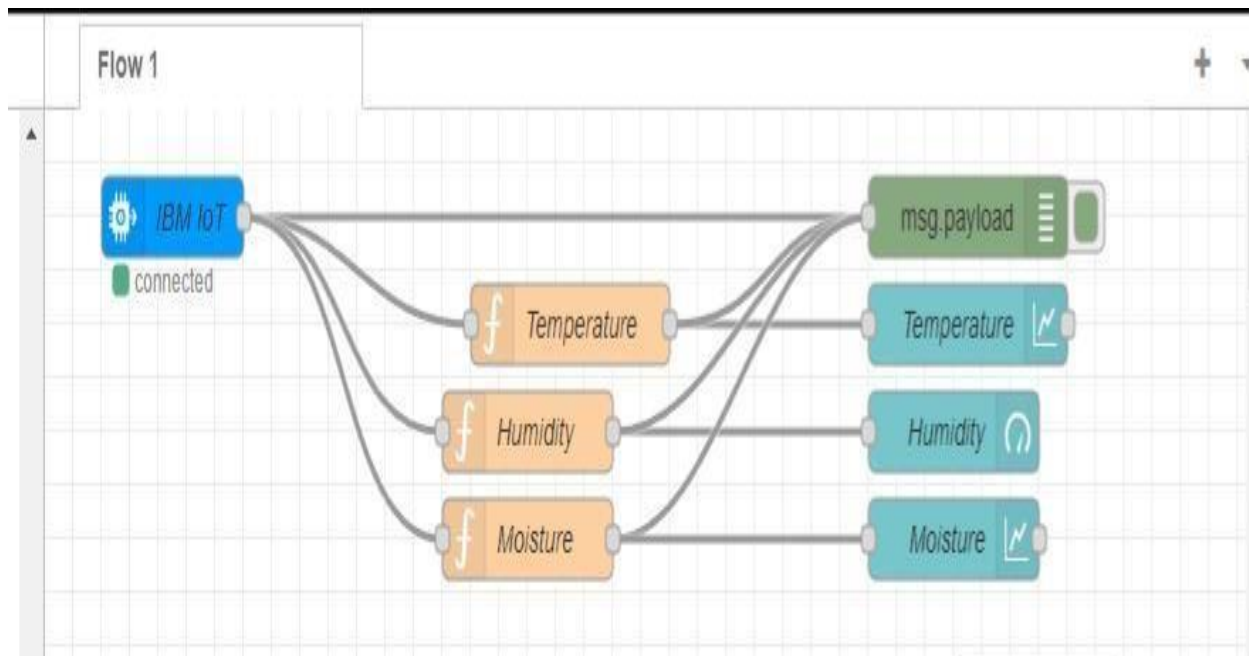
The Java script code for the function node is:

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

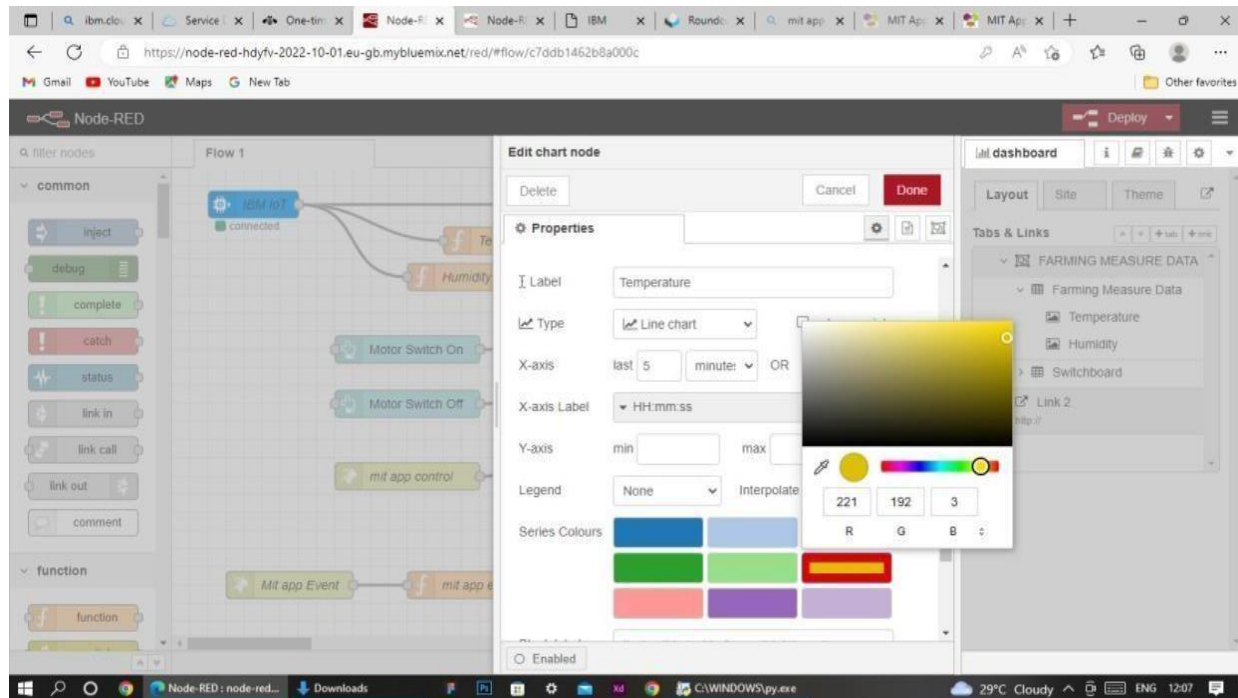
Finally connect Gauge nodes from dashboard to see the data in UI



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.

## 5.2 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

The screenshot displays the Node-RED web interface in a browser. The main workspace shows a flow with an 'IBM IoT' node connected to a 'function' node. The 'function' node is selected, and its configuration panel is open. The 'Name' field is set to 'Temperature'. The 'On Message' tab is active, showing the following JavaScript code:

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

The 'debug' console on the right shows a series of messages. The first message is a JSON object: `{ temp: 107, Humid: 73 }`. The subsequent messages show the raw JSON payload from the IoT sensor, including fields like `iot-2/type/abcd/id/7654321/ev/IoTSensor/fmt/json` and `msg.payload : number`.