

# Smart Farmer-IOT Enabled Smart Farming Application

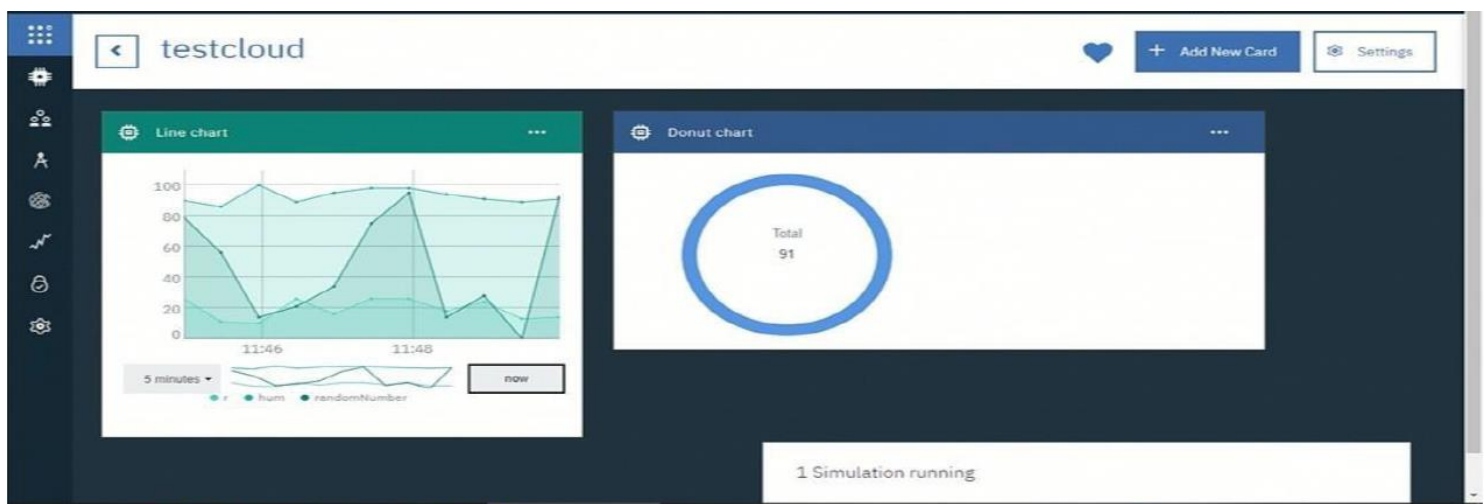
## SPRINT DELIVERY- 2

TITLE	Smart Farmer-IOT Enabled Smart Farming Application
DOMAIN NAME	INTERNET OF THINGS
TEAM ID	PNT2022TMID21357

### Building Project

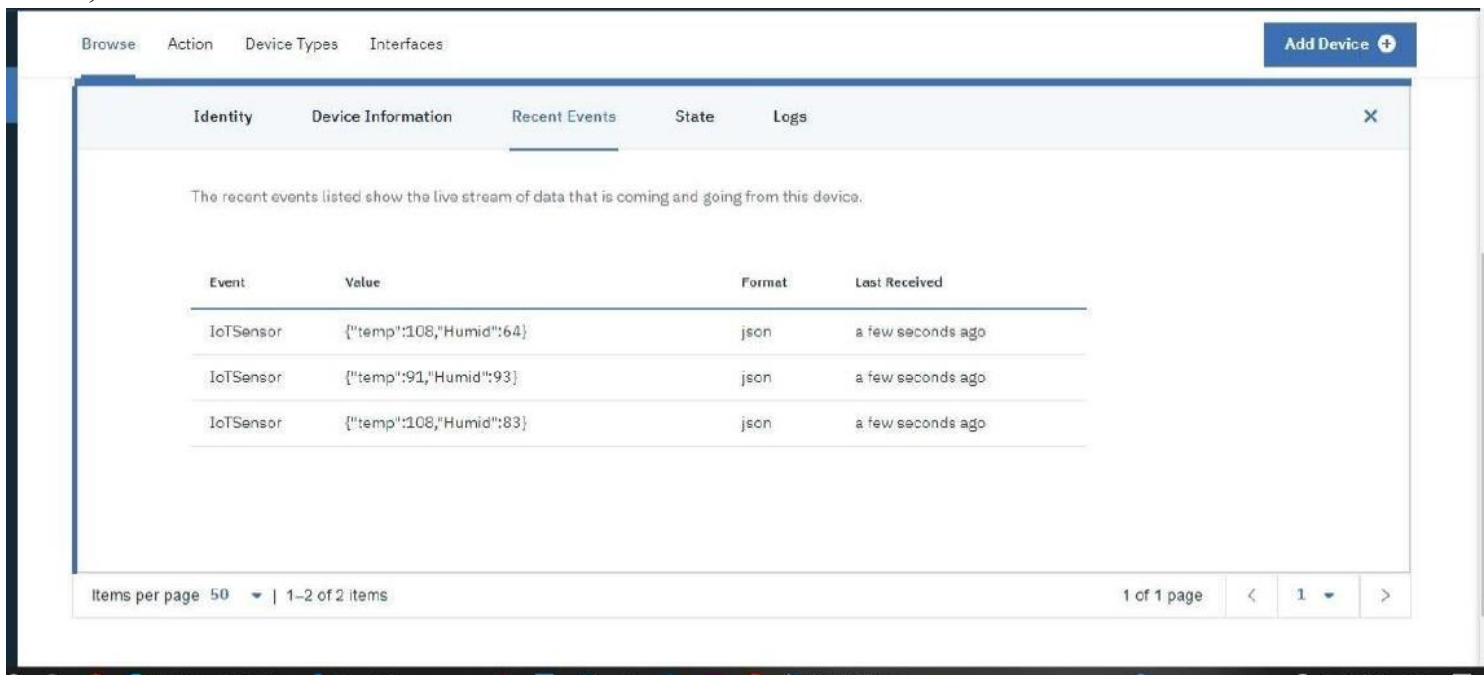
#### Connecting IoT Simulator to IBM Watson IoT Platform

- Open link provided in above section 4.3
- Give the credentials of your device in IBM Watson IoT Platform Click on connect
- Using the credentials
- 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
  }
}
```



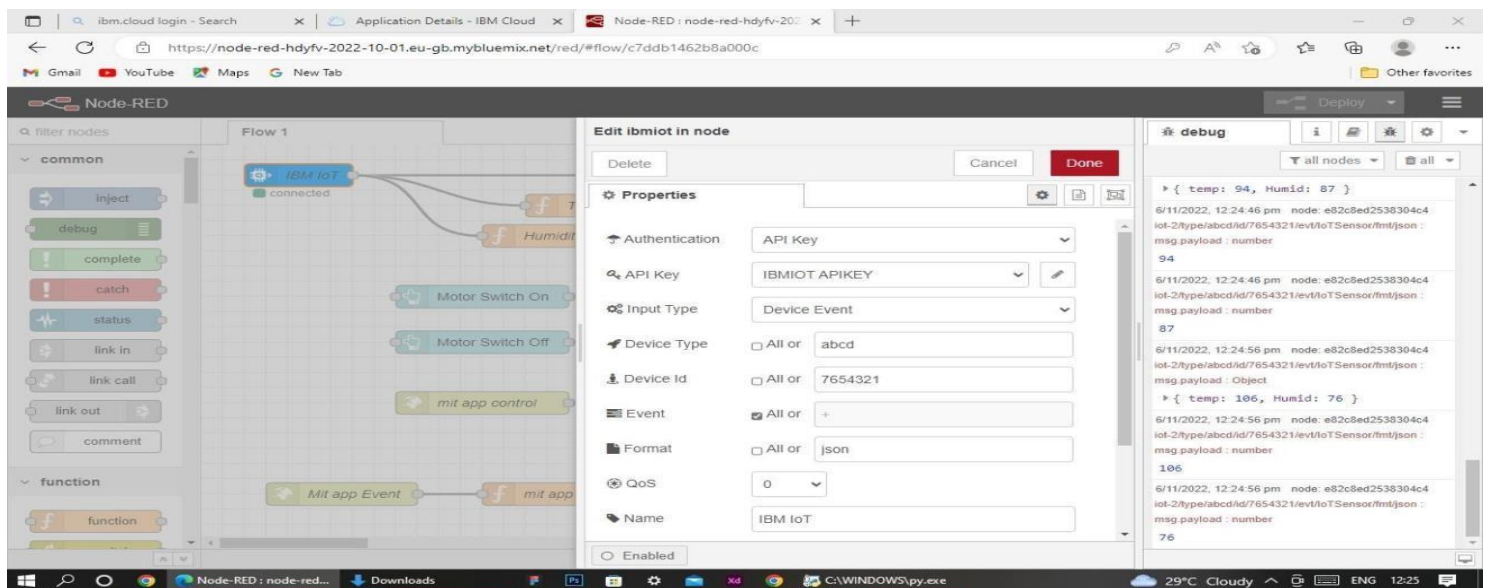
The screenshot shows the IBM IoT Dashboard interface. At the top, there are tabs for 'Browse', 'Action', 'Device Types', and 'Interfaces'. On the right, there is a blue button labeled 'Add Device +'. Below these, a modal window is open with tabs for 'Identity', 'Device Information', 'Recent Events', 'State', and 'Logs'. The 'Recent Events' tab is selected, displaying a message: 'The recent events listed show the live stream of data that is coming and going from this device.' Below this message is a table with the following data:

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 modal, there is a pagination bar showing 'Items per page 50' and '1-2 of 2 items'. On the right side of the modal, there is a '1 of 1 page' indicator and navigation buttons for previous and next pages.

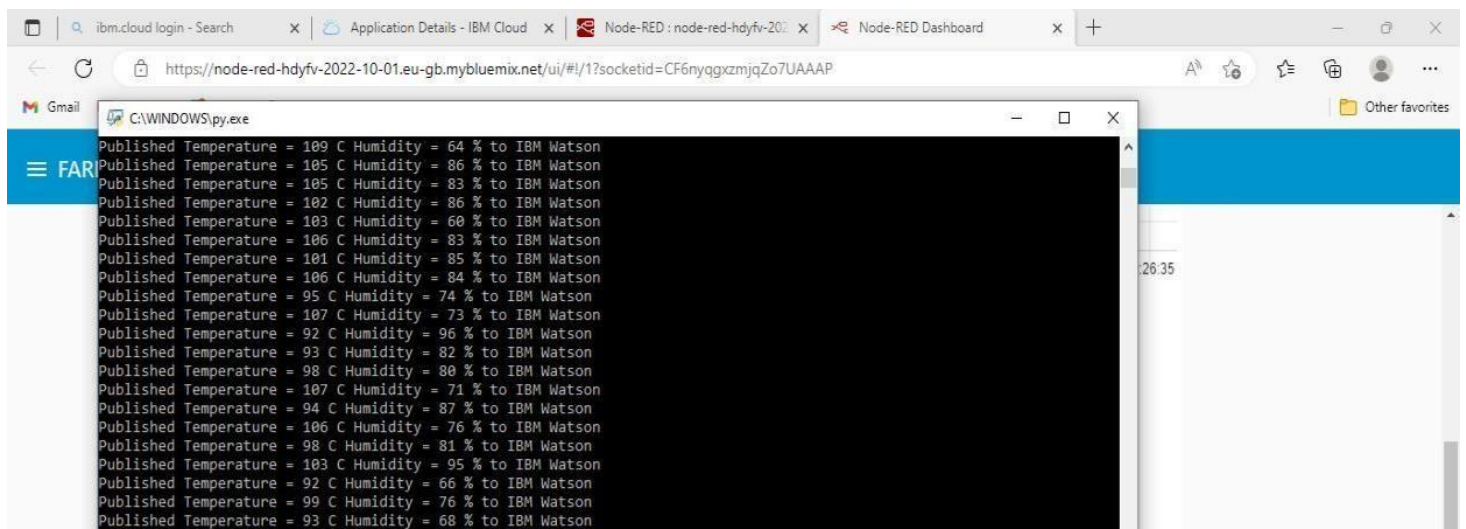
## Configuration of Node-Red to collect IBM cloud data

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

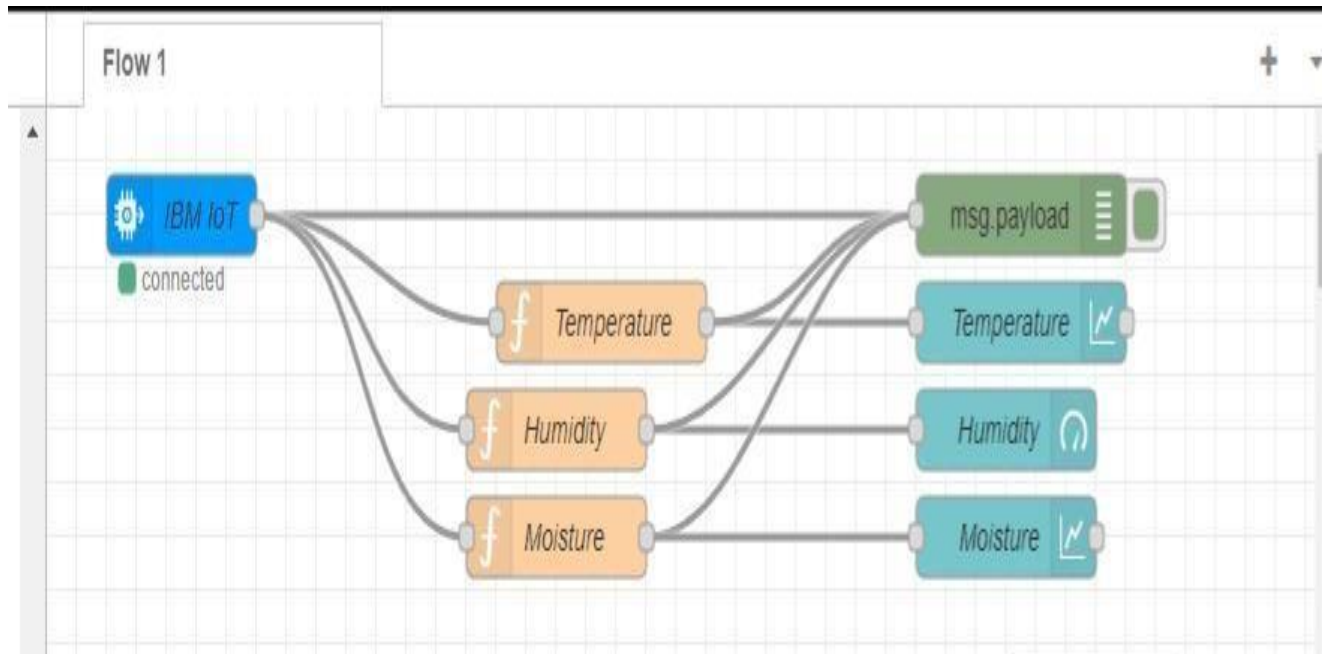


- Once it is connected Node-Red receives data from the deviceDisplay the data using debug node for verification
- Connect function node and write the Java script code to get each reading separately.
- The Java script code for the function node is:  

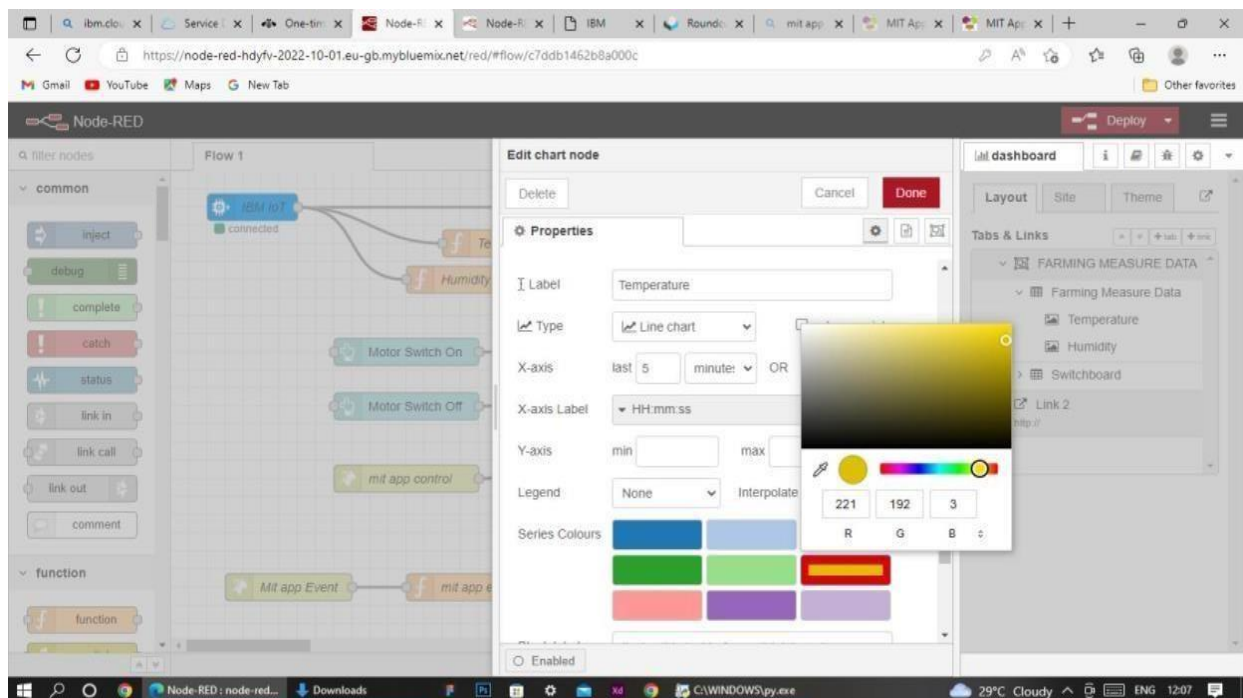
```
msg.payload=msg.payload.d.temperature return msg;
```
- Finally connect Gauge nodes from dashboard to see the data in UI



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 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.

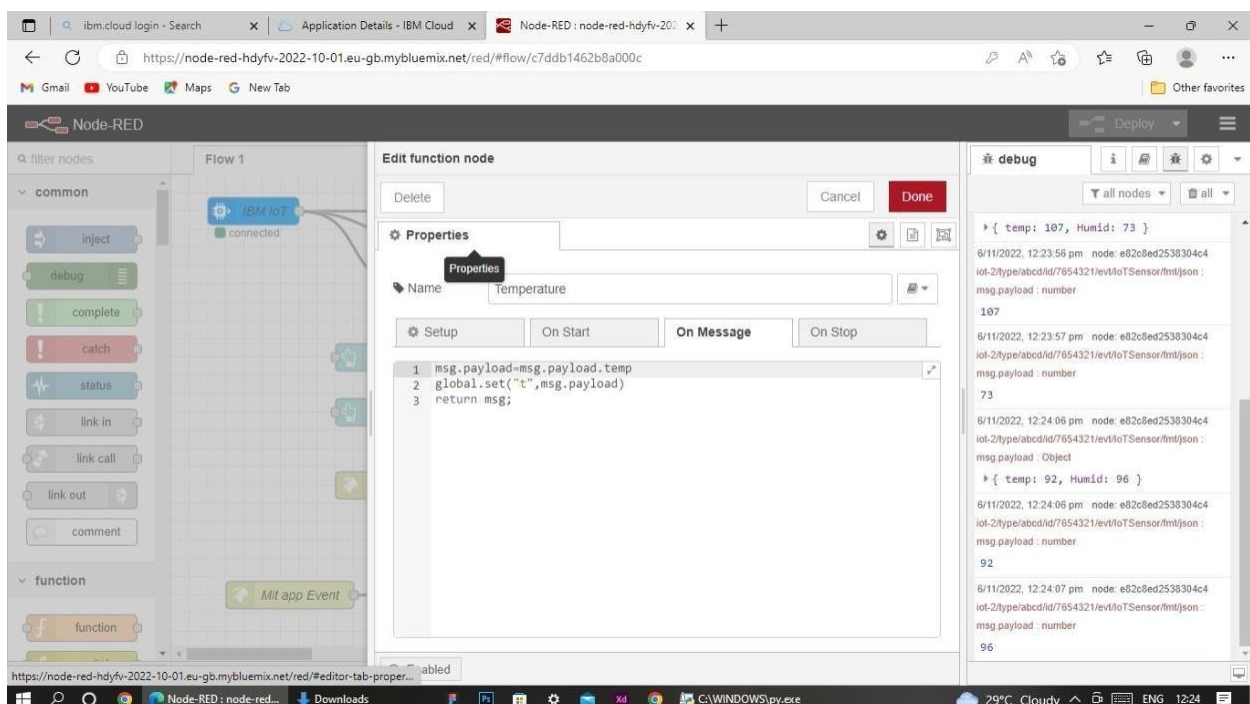
HTTP request node is configured with URL we saved before in section 4.4 The data we receive from Open Weather after request is in below JSON

```
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":"Gūdür","cod":200}
```

In order to parse the JSON string, we use Java script functions and get each parameter

```
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