

## Sprint Delivery - 2

**SmartFarmer - IoT Enabled Smart Farming Application**

**Team ID: PNT2022TMID22052**

**Date:19/11/2022**

### Building Project

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

Open link provided in above section

Give the credentials of your device in IBM Watson IoT Platform

My credentials given to simulator are:

Organization ID	uqweg2
Device Type	ESP8266
Device ID	5661
Authentication Method	use-token-auth
Authentication Token	cN3d)1wN70yDQsSFUQ

---

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 a table of devices with columns: Device ID, Status, Device Type, Class ID, Date Added, and Descriptive Location. Two devices are listed: one with ID 12345 (Disconnected, Jarvis) and another with ID 5661 (Connected, ESP8266). The device with ID 5661 is selected, and its 'Recent Events' tab is active. This tab shows a list of events with columns: Event, Value, Format, and Last Received. The events are JSON payloads containing temperature, humidity, and moisture data, all received 'a few seconds ago'.

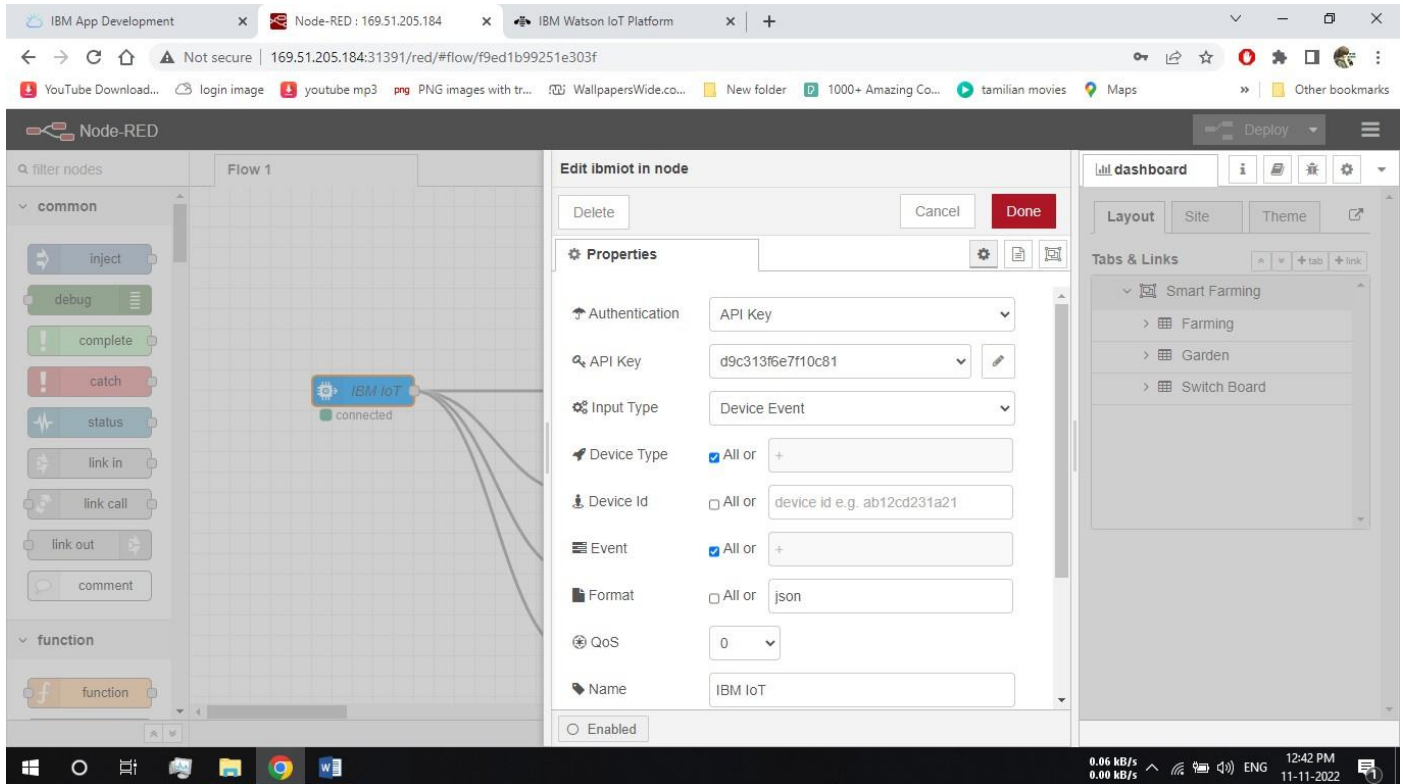
Device ID	Status	Device Type	Class ID	Date Added	Descriptive Location
12345	Disconnected	Jarvis	Device	8 Nov 2022 11:32 AM	
5661	Connected	ESP8266	Device	12 Nov 2022 10:20 PM	

Event	Value	Format	Last Received
status	{"d":{"temp":28.79999924,"humidity":52.90000...	json	a few seconds ago
status	{"d":{"temp":28.79999924,"humidity":52.90000...	json	a few seconds ago
status	{"d":{"temp":28.60000038,"humidity":52.90000...	json	a few seconds ago
status	{"d":{"temp":28.60000038,"humidity":52.90000...	json	a few seconds ago
status	{"d":{"temp":28.79999924,"humidity":52.70000...	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

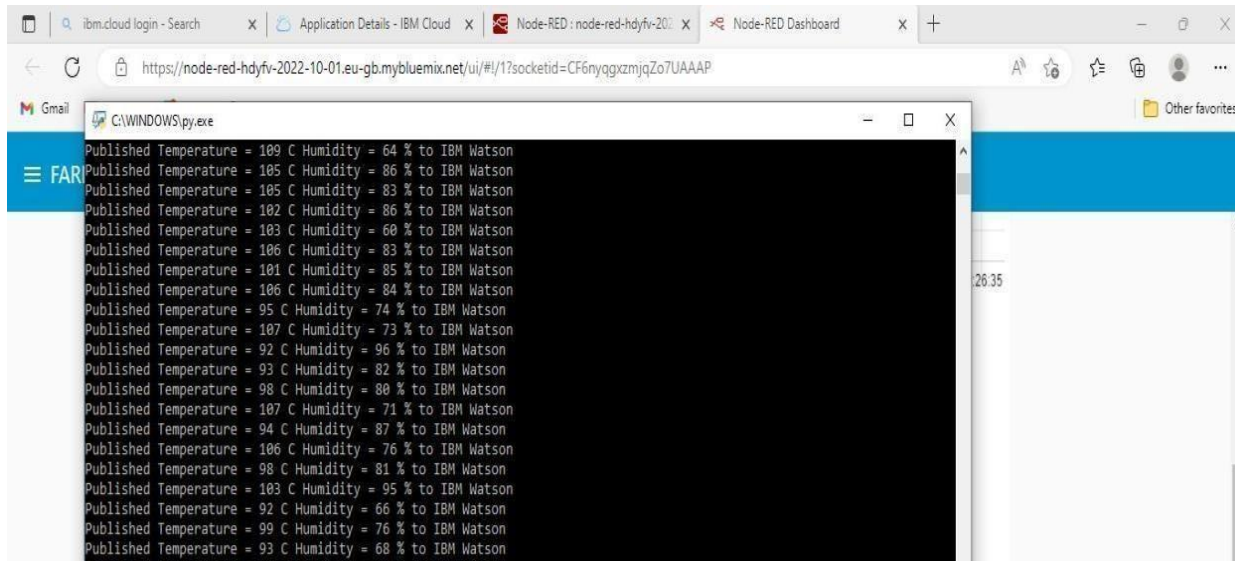
Display the data using debug node for verification

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

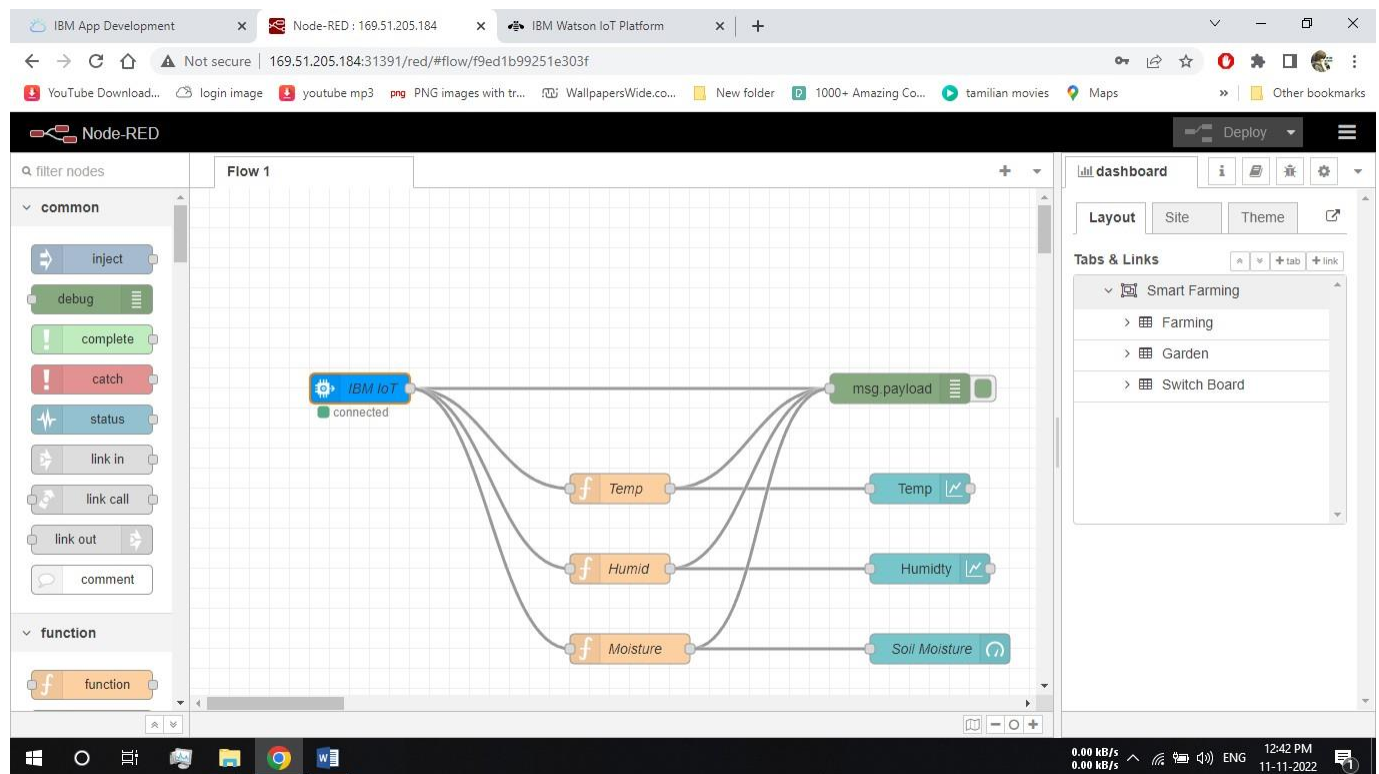
The Java script code for the function node is:

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

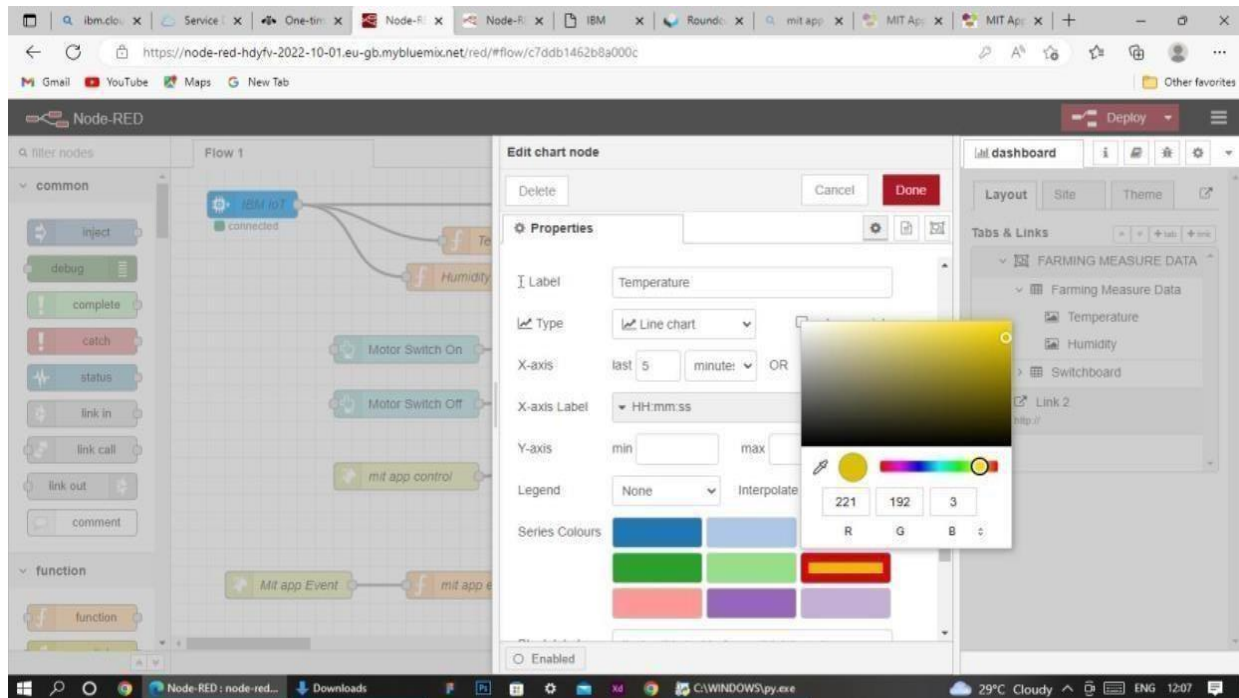
Finally connect Gauge nodes from dashboard to see the data in UI



## 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 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,"h
umidity":35,"sea_level":1002,"grnd_level":1000},"wind":{"speed":6.23,"deg":170
}
,"clouds":{"all":68},"dt":1589991979,"sys":{"country":"IN","sunrise":158993355
3,
```

```
"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

