

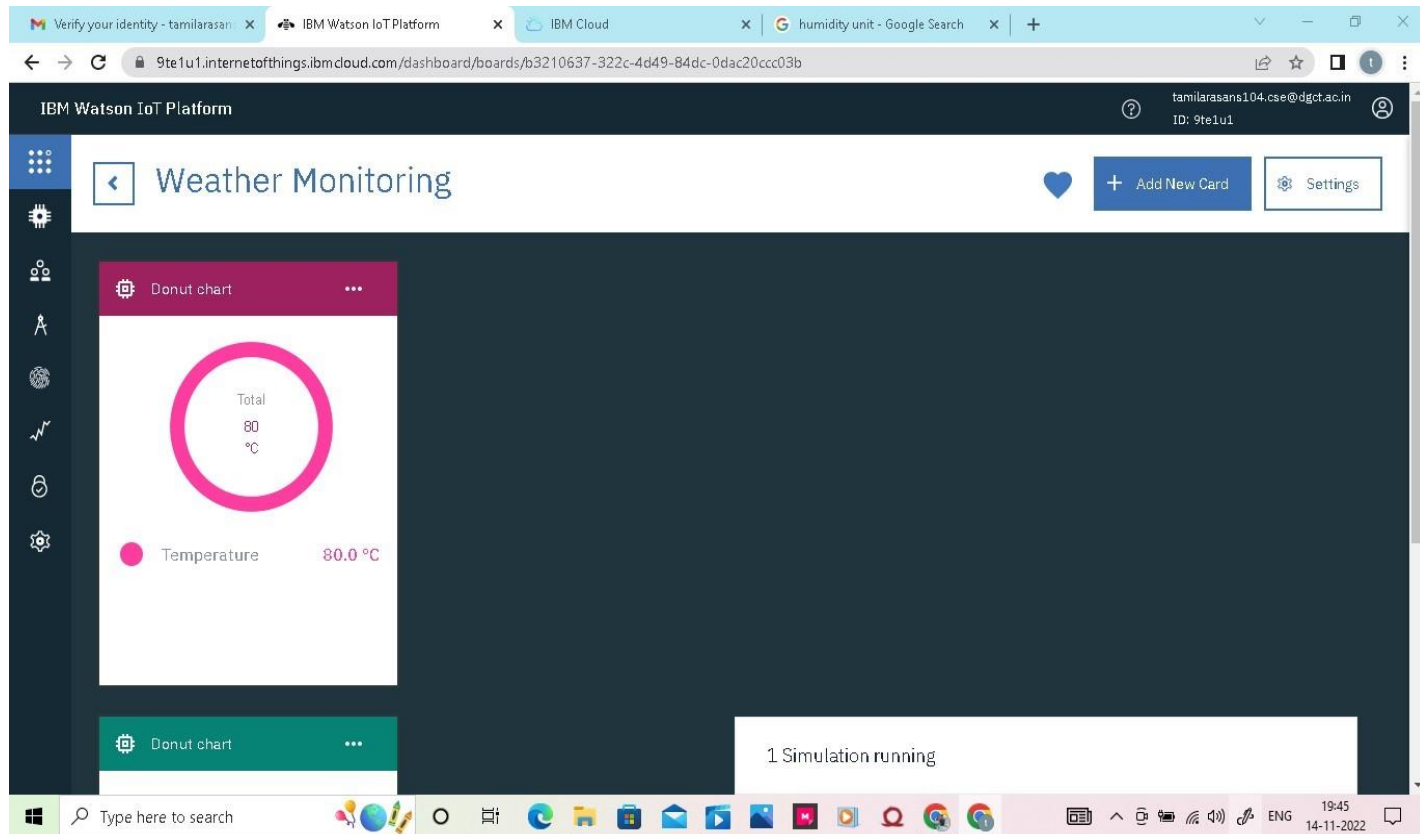
IOT ENABLED SMART FARMING APPLICATION

SPRINT DELIVERY – 2

TEAM ID:PNT2022TMID29852

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:

Organization ID

9te1u1

Device Type

SFTTMS00

Device ID

SFTTMS11

Authentication Method

use-token-auth

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

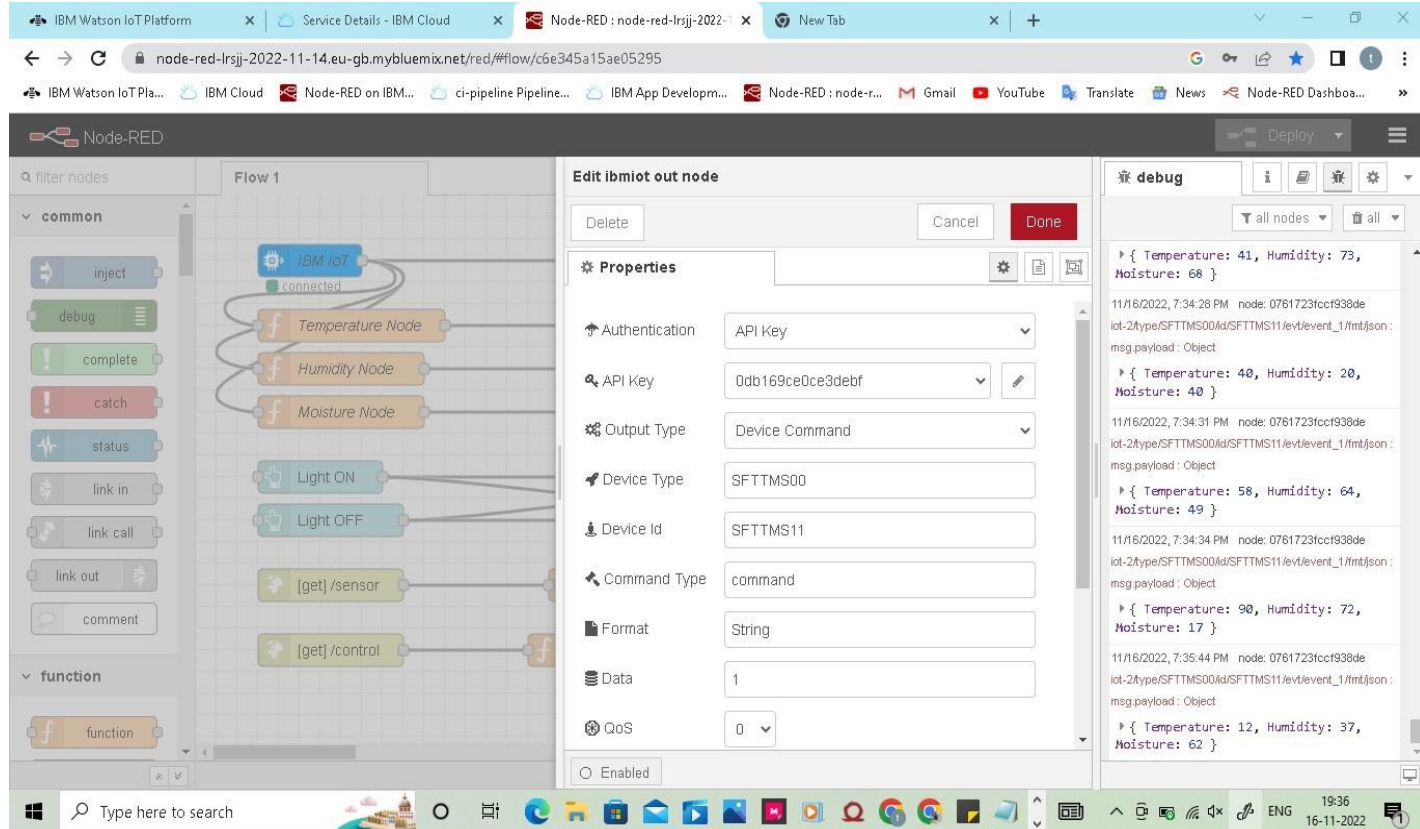
The screenshot displays the IBM Watson IoT Platform dashboard. The browser address bar shows the URL `9te1u1.internetofthings.ibmcloud.com/dashboard/devices/browse`. The dashboard header includes the IBM Watson IoT Platform logo and a user profile for `tamilarasans104.cse@dgct.ac.in` with ID `9te1u1`. The main navigation menu on the left includes icons for Dashboard, Devices, Actions, and Settings. The top navigation bar has tabs for `Browse`, `Action`, `Device Types`, and `Interfaces`, along with an `Add Device` button.

The `Browse` tab is active, showing a list of devices. The first device, `smartfarming123`, is highlighted. Below the device list, a table displays the recent events for this device. The table has four columns: `Event`, `Value`, `Format`, and `Last Received`. The events are listed as `event_1` with JSON values for Temperature, Humidity, and Moisture, all in `json` format and received 'a few seconds ago'.

At the bottom of the dashboard, a status bar shows the device `Smart_Farming123` as `Disconnected` and the time `Nov 16, 2022 12:31 PM`. A notification bubble indicates `1 Simulation running`. The Windows taskbar at the bottom shows the search bar and various application icons.

Event	Value	Format	Last Received
event_1	{"Temperature":98,"Humidity":35,"Moisture":43}	json	a few seconds ago
event_1	{"Temperature":60,"Humidity":85,"Moisture":40}	json	a few seconds ago
event_1	{"Temperature":99,"Humidity":8,"Moisture":17}	json	a few seconds ago

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

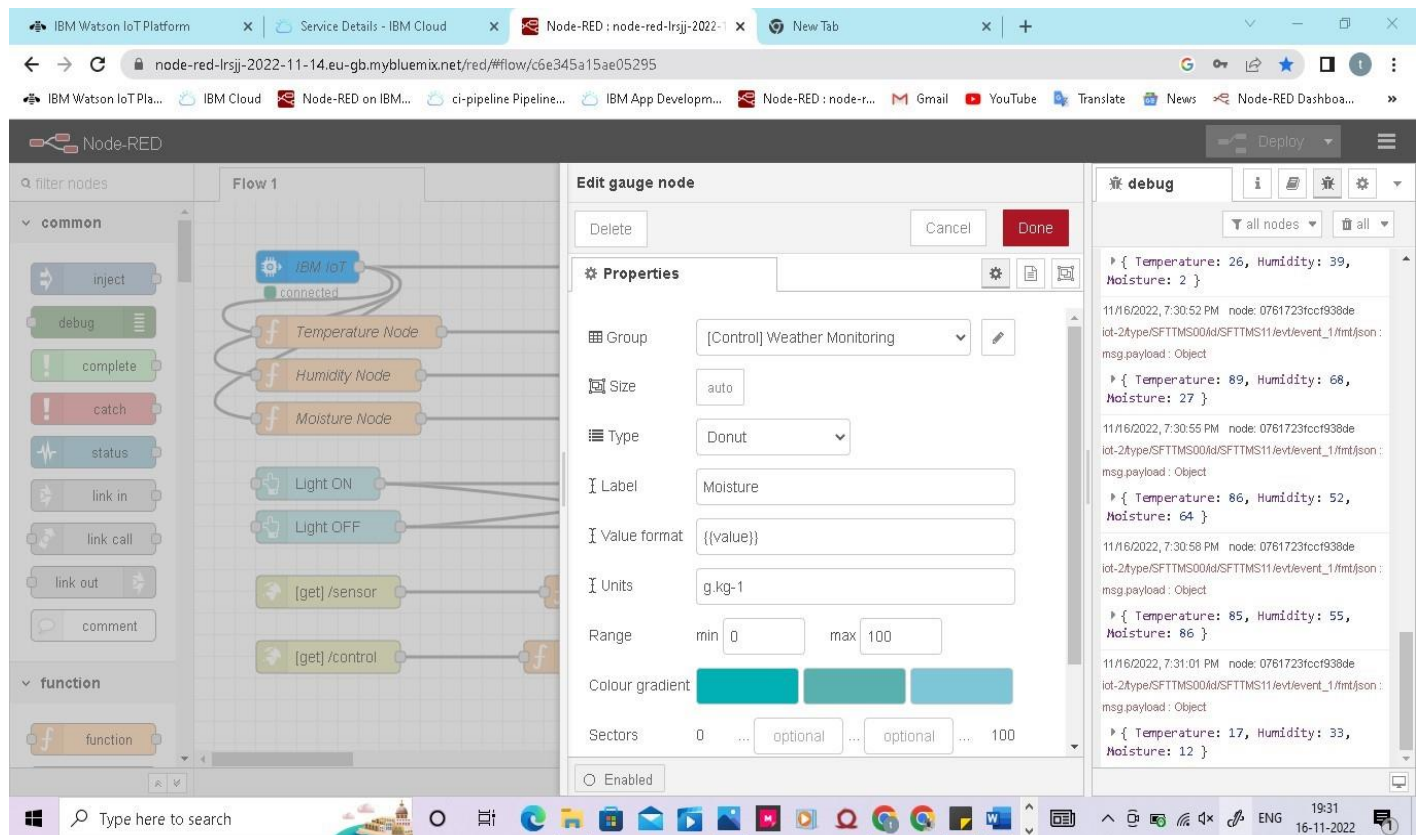
The screenshot displays the Node-RED web interface in a browser. The top navigation bar includes links to IBM Watson IoT Platform, Node-RED, and IBM Cloud. The main workspace shows a flow named 'Flow 1' with the following components and connections:

- Input:** An 'ibmiot in' node is connected to a 'msg.payload' node.
- Processing:** The 'msg.payload' node is connected to three function nodes: 'Temperature Node', 'Humidity Node', and 'Moisture Node'.
- Output:** Each function node is connected to a corresponding output node: 'Temperature', 'Humidity', and 'Moisture'.
- Control:** Two 'Light ON' and 'Light OFF' nodes are connected to a 'msg.payload' node, which is then connected to an 'ibmiot out' node.

The right-hand 'debug' console shows the following log entries:

```
11/15/2022, 8:34:18 PM node: 85653cb4c5216dc9  
msg.payload: Object  
{ command: "lighton" }  
11/15/2022, 8:34:20 PM node: 85653cb4c5216dc9  
msg.payload: Object  
{ command: "lightoff" }  
11/15/2022, 8:34:34 PM node: 85653cb4c5216dc9  
msg.payload: Object  
{ command: "lightoff" }
```

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 named 'Flow 1' with the following components:

- IBM IoT Node:** Labeled 'connected', it receives data from an IoT device.
- Function Node:** Labeled 'httpfunction Node', it contains the following JavaScript code:

```
1 msg.payload={"Temperature":global.get("t"),"Humidity":global.get("h"), "Moisture":global.get("m")};  
2 return msg;
```
- Output Nodes:** Three nodes labeled 'Temperature Node', 'Humidity Node', and 'Moisture Node' are connected to the function node. These nodes are further connected to 'Light ON', 'Light OFF', and '[get]/sensor' nodes respectively.

The right sidebar shows the 'debug' console with the following log entries:

```
{ Temperature: 11, Humidity: 10, Moisture: 89 }  
11/16/2022, 7:30:19 PM node: 0761723fct938de  
iot-2type/SFTTMS000Id/SFTTMS11/evt/event_1/fmt/json :  
msg.payload : Object  
{ Temperature: 70, Humidity: 61, Moisture: 24 }  
11/16/2022, 7:30:22 PM node: 0761723fct938de  
iot-2type/SFTTMS000Id/SFTTMS11/evt/event_1/fmt/json :  
msg.payload : Object  
{ Temperature: 28, Humidity: 3, Moisture: 62 }  
11/16/2022, 7:30:25 PM node: 0761723fct938de  
iot-2type/SFTTMS000Id/SFTTMS11/evt/event_1/fmt/json :  
msg.payload : Object  
{ Temperature: 9, Humidity: 58, Moisture: 94 }  
11/16/2022, 7:30:28 PM node: 0761723fct938de  
iot-2type/SFTTMS000Id/SFTTMS11/evt/event_1/fmt/json :  
msg.payload : Object  
{ Temperature: 88, Humidity: 10, Moisture: 96 }
```