

SPRINT DELIVERY – 2

Team ID	PNT2022TMID47645
Project	IoT Enabled Smart Farming Application
Date	15 November 2022

5. Building Project

Connecting IOT Simulator
to IBM WatsonIOTPlatform

Open link provided in above
section 4.3 Give the credentials of
your device in IBM Watson
IOTPlatformClick on connect
My credentials given to simulator are:

OrgID: **4clor3** api: **a-**

157uf3- f5rg4qxp3

Device type: **NodeMCU**

token: **6ogMaaQHNWF**

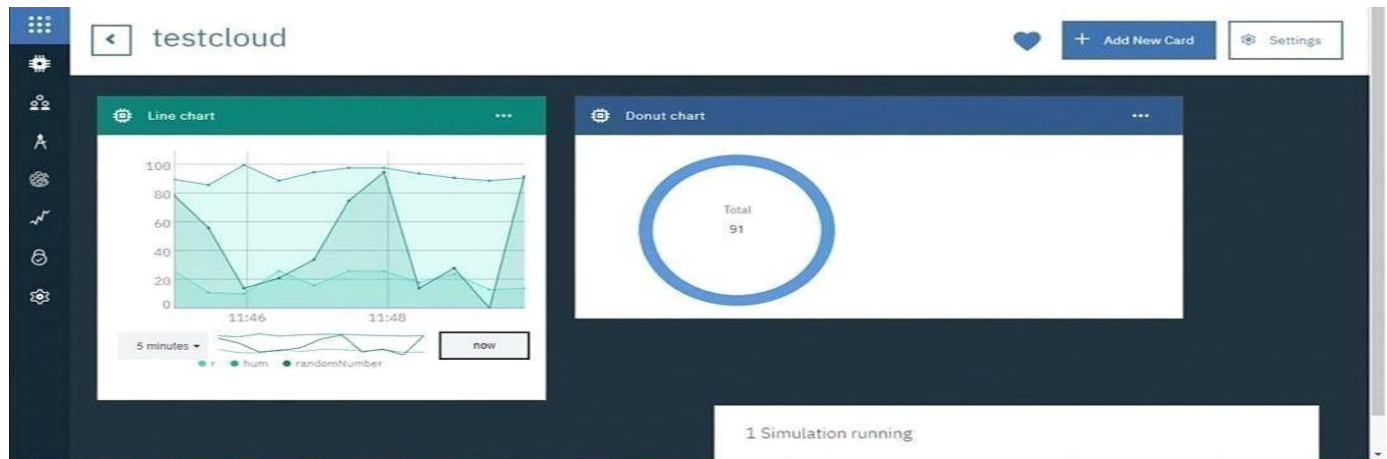
EgOD8R?Device ID :

1234

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": {
```

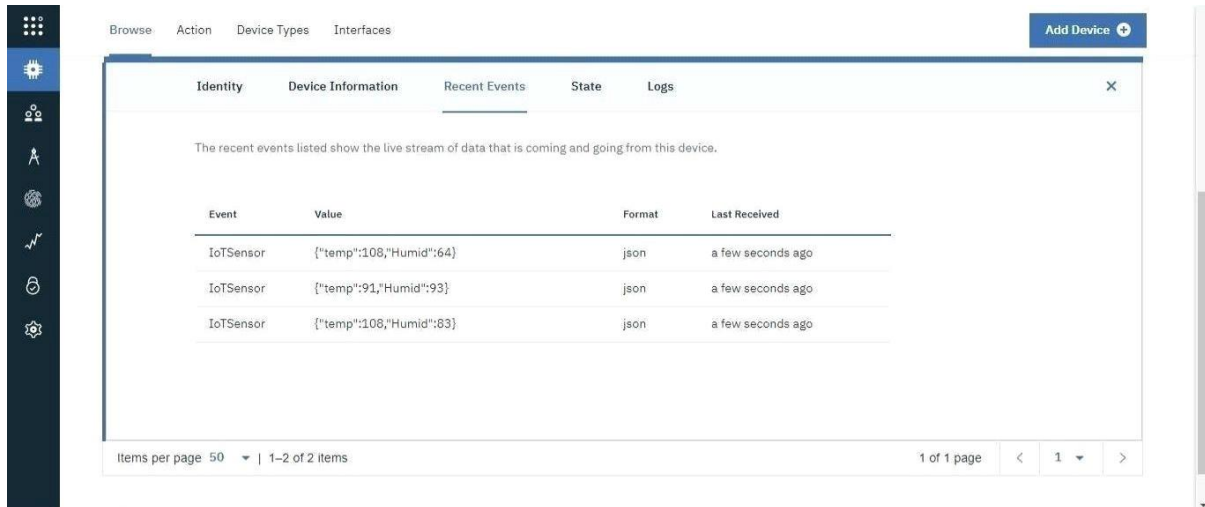
- "name": "NodeMCU",

- "temperature": 17,

- "humidity": 76,

- "Moisture ": 25

```
}  
  
}
```

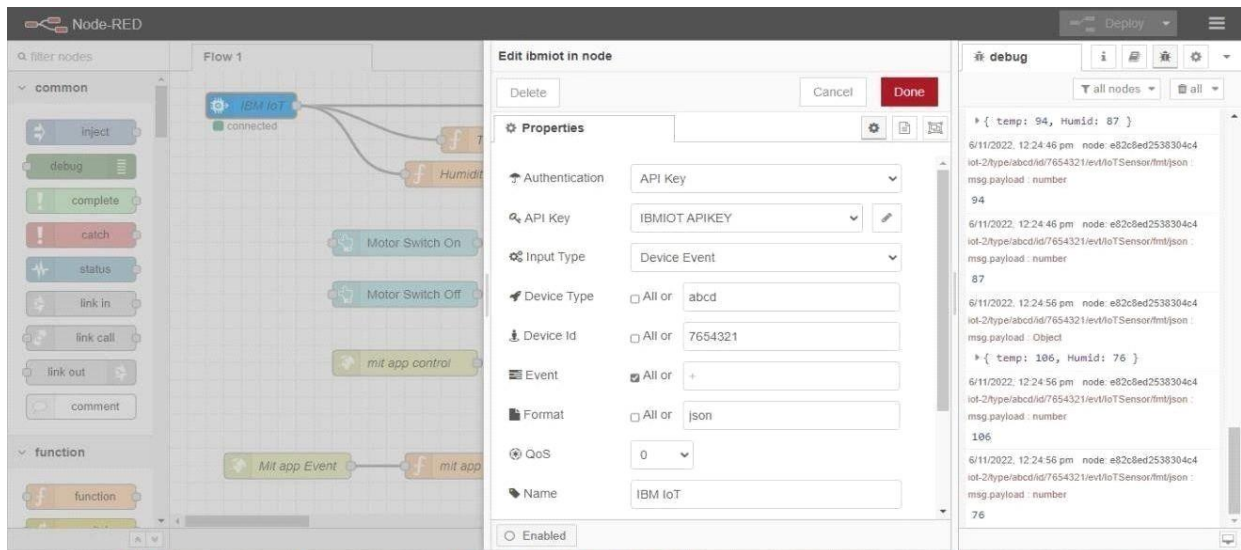


The screenshot shows the IBM IoT Dashboard interface. The 'Recent Events' tab is selected, displaying a table of events. The table has four columns: Event, Value, Format, and Last Received. There are three rows of data, all from an 'IoT Sensor' with a 'json' format, received 'a few seconds ago'. The values are JSON objects containing temperature and humidity data.

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

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.

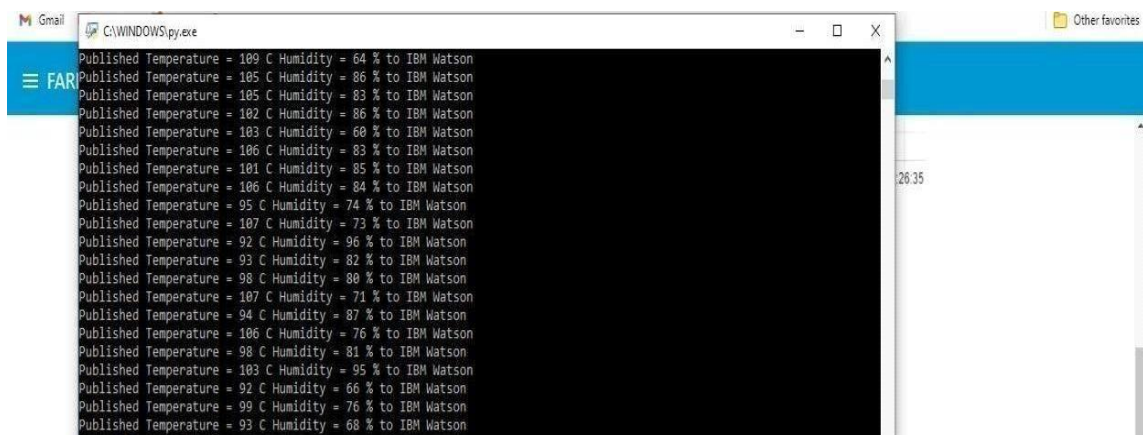


Once it is connected Node-Red receives data from the deviceDisplaythe data using debug node for verification

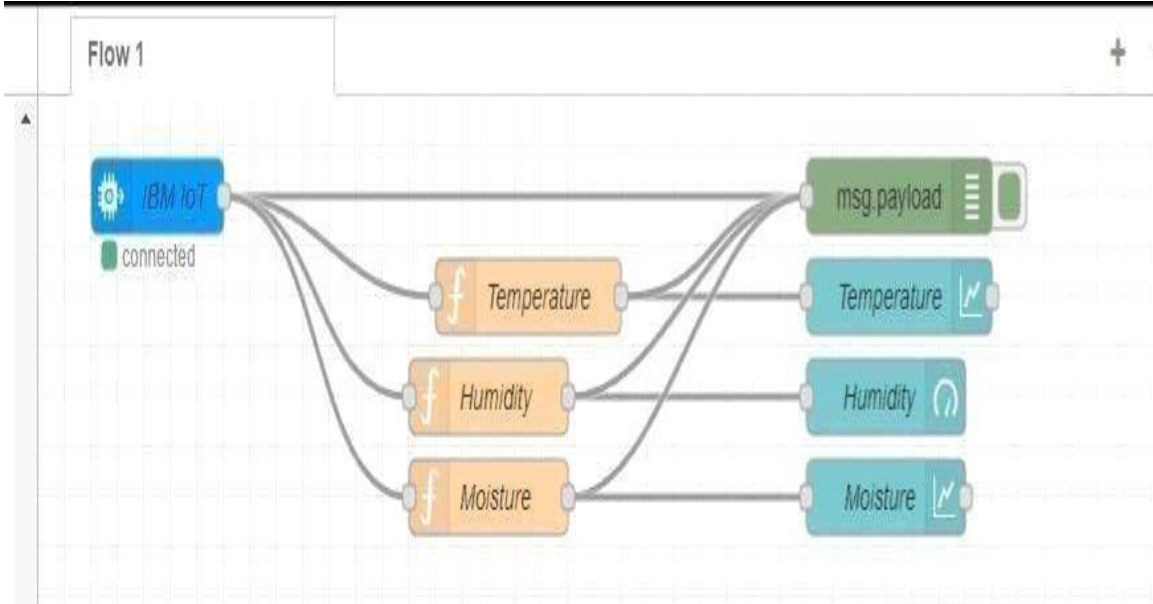
Connect function node and write the Java script code to get each readingseparately.The Java script code for the function node is:

```
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

The image shows the Node-RED interface with the 'Edit chart node' panel open. The 'Properties' section is visible, showing the following configuration:

- Label:** Temperature
- Type:** Line chart
- X-axis:** last 5 minute: OR
- X-axis Label:** HH:mm:ss
- Y-axis:** min max
- Legend:** None Interpolate
- Series Colours:** A color palette is shown with various colors.

The 'Enabled' checkbox is checked. The background shows a flow diagram with an 'IBM IoT' node connected to 'Temperature' and 'Humidity' function nodes. The right sidebar shows a 'dashboard' with a 'FARMING MEASURE DATA' tab and a 'Switchboard' link.

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":"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

