Smart Farmer-IOT Enabled Smart Farming Application

SPRINT DELIVERY-2

TEAM ID: PTN2022TMID02589

TEAM MEMBERS

KESHIKA B

ABINAYA E

KAAVYA P

ARUNA K

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:

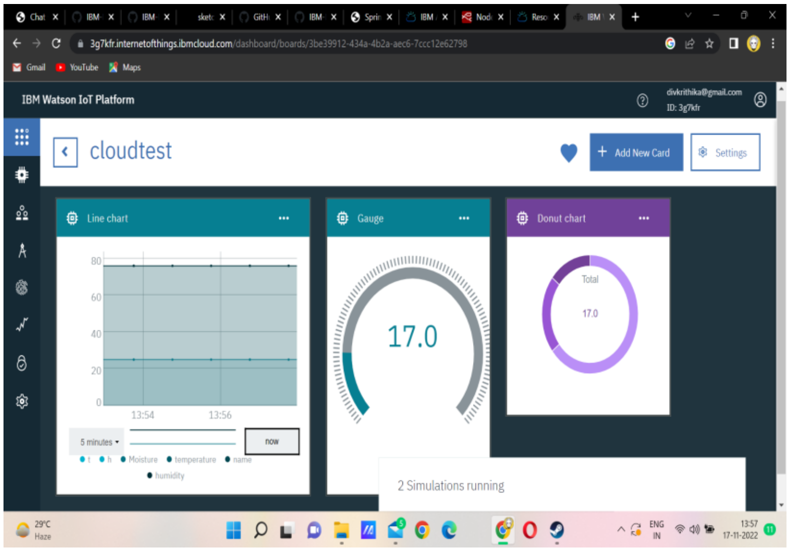
api: a-3g7kfr-wzcassbg77

Device type: abcd

Device ID : abcd_1

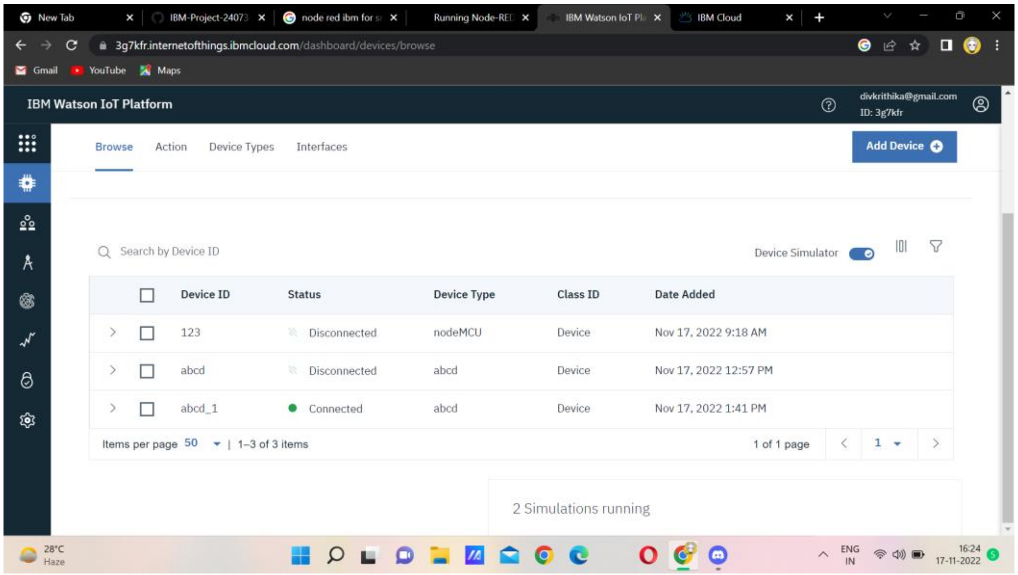
Authentication Token: kEgHg5*rkxhA-Sqx

Organisation id : 3g7kfr



- 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
 }
}
```



3g7kfr.internetofthings.ibmcloud.com/dashboard/devices/browse

IBM Watson IoT Platform

divkrithika@gmail.com
ID: 3g7kfr

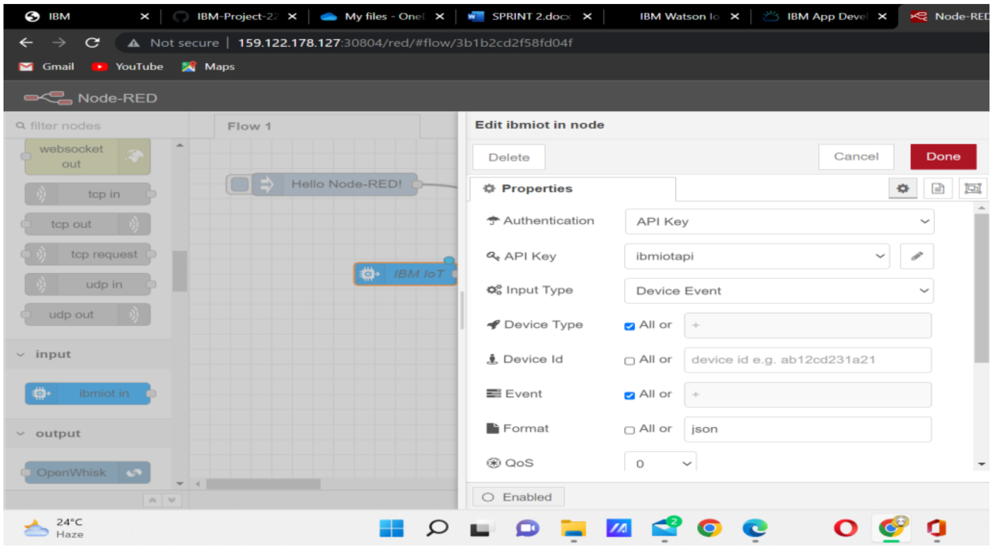
	Device ID	Status	Device Type	Class ID	Date Added
□	123	Disconnected	nodeMCU	Device	Nov 17, 2022 9:18 AM
□	abcd	Disconnected	abcd	Device	Nov 17, 2022 12:57 PM
□	abcd_1	● Connected	abcd	Device	Nov 17, 2022 1:41 PM

Items per page 50 | 1–3 of 3 items   1 of 1 page

2 Simulations running

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.



Not secure | 159.122.178.127:30804/red/#flow/3b1b2cd2f58fd04f

Edit ibmiot in node

Authentication: API Key

API Key: ibmiotapi

Input Type: Device Event

Device Type: ☑ All or

Device Id: □ All or device id e.g. ab12cd231a21

Event: ☑ All or

Format: □ All or json

QoS: 0

○ Enabled

Once it is connected Node-Red receives data from the device

Display the data using debug node for verification

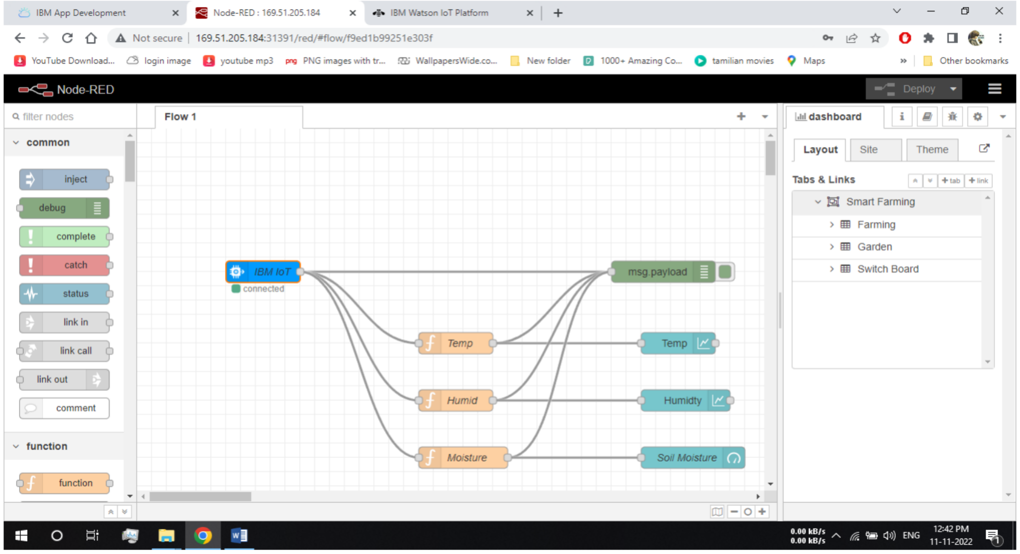
Connect function node and write the Java script code to get each reading separately.

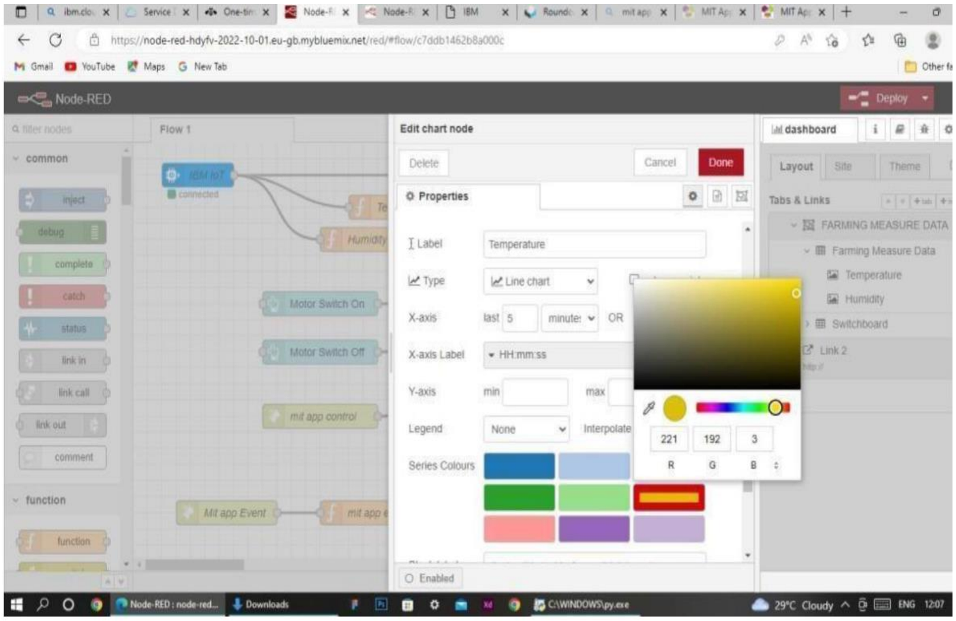
The Java script code for the function node is:

```
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





Label Temperature

Type Line chart

X-axis last 5 minute OR

X-axis Label HH:mm:ss

Y-axis min max

Legend None

Interpolate

221 192 3
R G B

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":3
07
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":
20
0}

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