

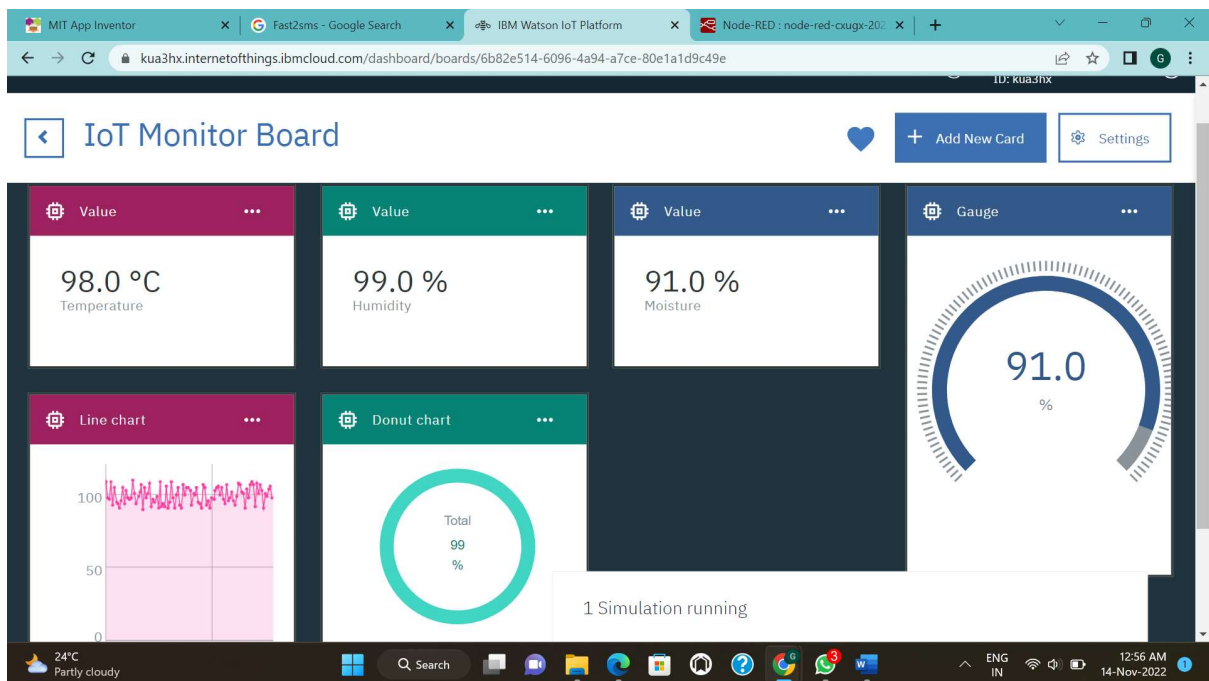
## Project Development Phase

### Delivery Of Sprint-2

<b>TITLE</b>	<b>Smart Farmer-IOT Enabled Smart Farming Application</b>
<b>DOMAIN NAME</b>	INTERNET OF THINGS
<b>TEAM ID</b>	PNT2022TMID23830
<b>Project Name</b>	Smart Farmer - IoT Enabled Smart Farming Application
<b>Leader Name</b>	GOWSALYA L
<b>Team Members Name</b>	DEEPIKA B K MEGAVARSHINI G MONISHA N
<b>MENTOR NAME</b>	THIRUPPATHI M

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

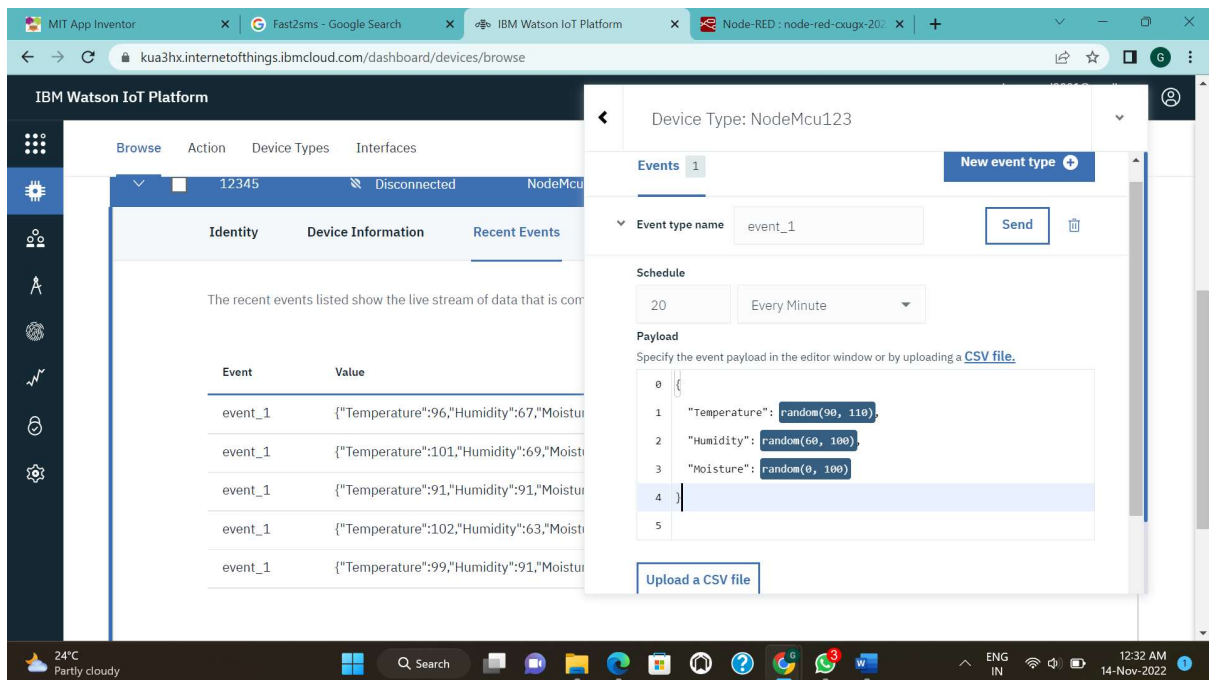
- Open link provided in below image
- Give the credentials of your device in IBM Watson Platform
- Click on connect
- My credentials given to simulator are:
  - **API:** a-kua3hx-bm5z9ikjfu
  - **Device type:** NodeMcu123
  - **Token:** 123456789
  -



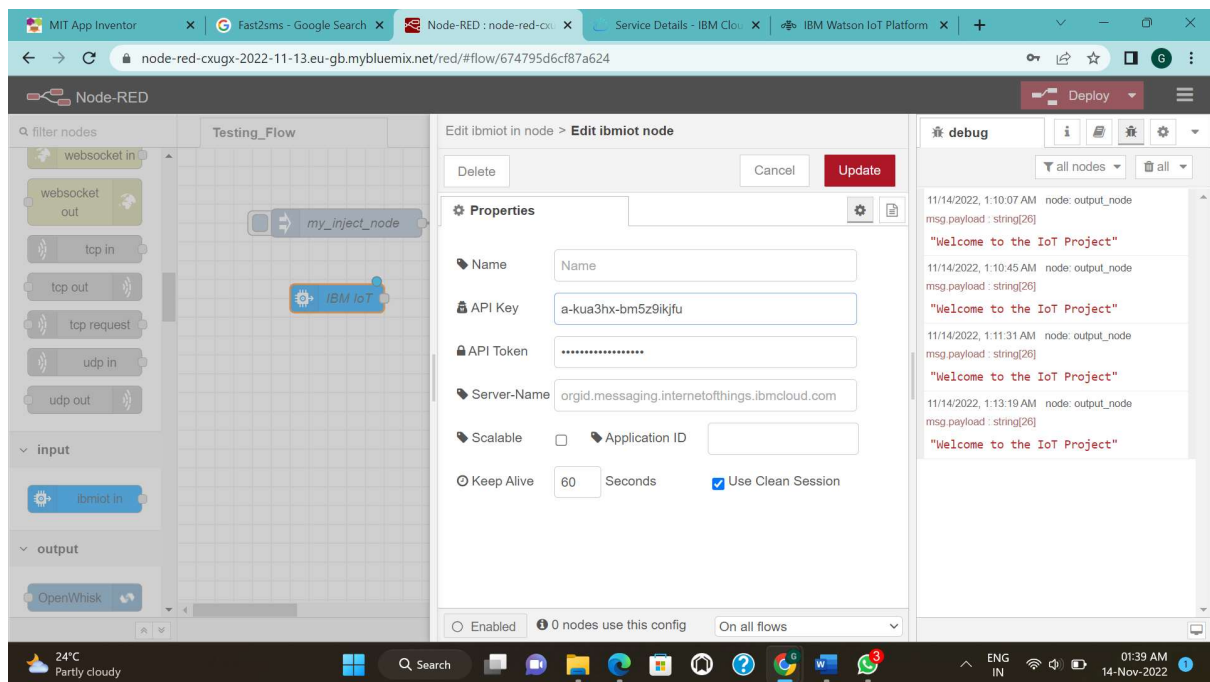
You can see the received data in graphs by creating cards in Boards tab

- 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":
  {
    ▪ "temperature": 96,
    ▪ "humidity": 67,
    ▪ "Moisture ": 25
  }
}
```



## 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

Service Details - IBM Cloud | IBM Watson IoT Platform | MIT App Inventor | Node-RED : node-red-ctrl | Node-RED Dashboard

node-red-cxugx-2022-11-13.eu-gb.mybluemix.net/red/#flow/674795d6cf87a624

Node-RED

Testing\_Flow

filter nodes

numeric, text input, date picker, colour picker, form, text, gauge, chart, audio out, notification, ui control, template

IBM IoT (connected)

Temperature, Humidity, Moisture

msg\_payload

debug

all nodes

```

iot-2/type/NodeMcu123/id/12345/evt/event_1/fmt/json :
msg.payload : number
75

11/14/2022, 12:14:24 PM node: msg_payload
iot-2/type/NodeMcu123/id/12345/evt/event_1/fmt/json :
msg.payload : Object
{ Temperature: 107, Humidity: 91, Moisture: 10 }

11/14/2022, 12:14:24 PM node: msg_payload
iot-2/type/NodeMcu123/id/12345/evt/event_1/fmt/json :
msg.payload : number
107

11/14/2022, 12:14:25 PM node: msg_payload
iot-2/type/NodeMcu123/id/12345/evt/event_1/fmt/json :
msg.payload : number
91

11/14/2022, 12:14:26 PM node: msg_payload
iot-2/type/NodeMcu123/id/12345/evt/event_1/fmt/json :
msg.payload : number
10

```

27°C Cloudy

Search

ENG IN

12:14 PM 14-Nov-2022

➤ Data received from the cloud in Node-Red console

Service Details - IBM Cloud | IBM Watson IoT Platform | MIT App Inventor | Node-RED : node-red-ctrl | Node-RED Dashboard

node-red-cxugx-2022-11-13.eu-gb.mybluemix.net/red/#flow/674795d6cf87a624

Node-RED

Testing\_Flow

filter nodes

tone analyzer v3

dashboard

button, dropdown, switch, slider, numeric, text input, date picker, colour picker, form

IBM IoT (connected)

Temperature, Humidity, Moisture

msg\_payload

debug

all nodes

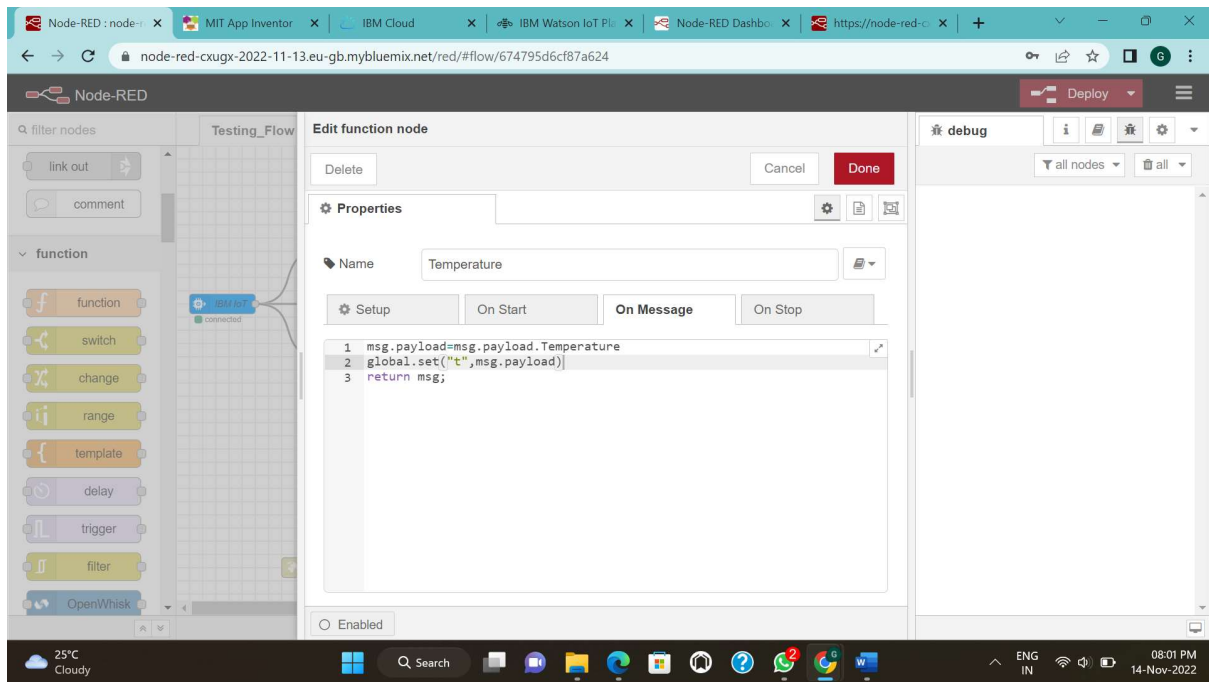
28°C Mostly cloudy

Search

ENG IN

01:11 PM 14-Nov-2022

➤ 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.

### 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.
- 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":"brokencloud","icon":"04n"}],
    "base":"stations",
    "main":{"temp":30759,
    "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

