

**DR. MAHALINGAM COLLEGE OF ENGINEERING AND
TECHNOLOGY, POLLACHI**

Department of Computer Science and Engineering

**Smart Farmer-IOT Enabled Smart Farming
Application**

IBM NALAIYATHIRAN

SPRINT-2

TITLE	Smart Farmer-IOT Enabled Smart Farming Application
DOMAIN NAME	INTERNET OF THINGS
TEAM ID	PNT2022TMID08684
LEADER NAME	KRISHNAPRASATH U
TEAM MEMBER NAME	SIVAROHITH A GIRIPRASATH S S NIRUTHEESH R HARIHARASUTHAN M
MENTOR NAME	PRABHU K

Building Project

Connecting IoT Simulator to IBM Watson IoT Platform

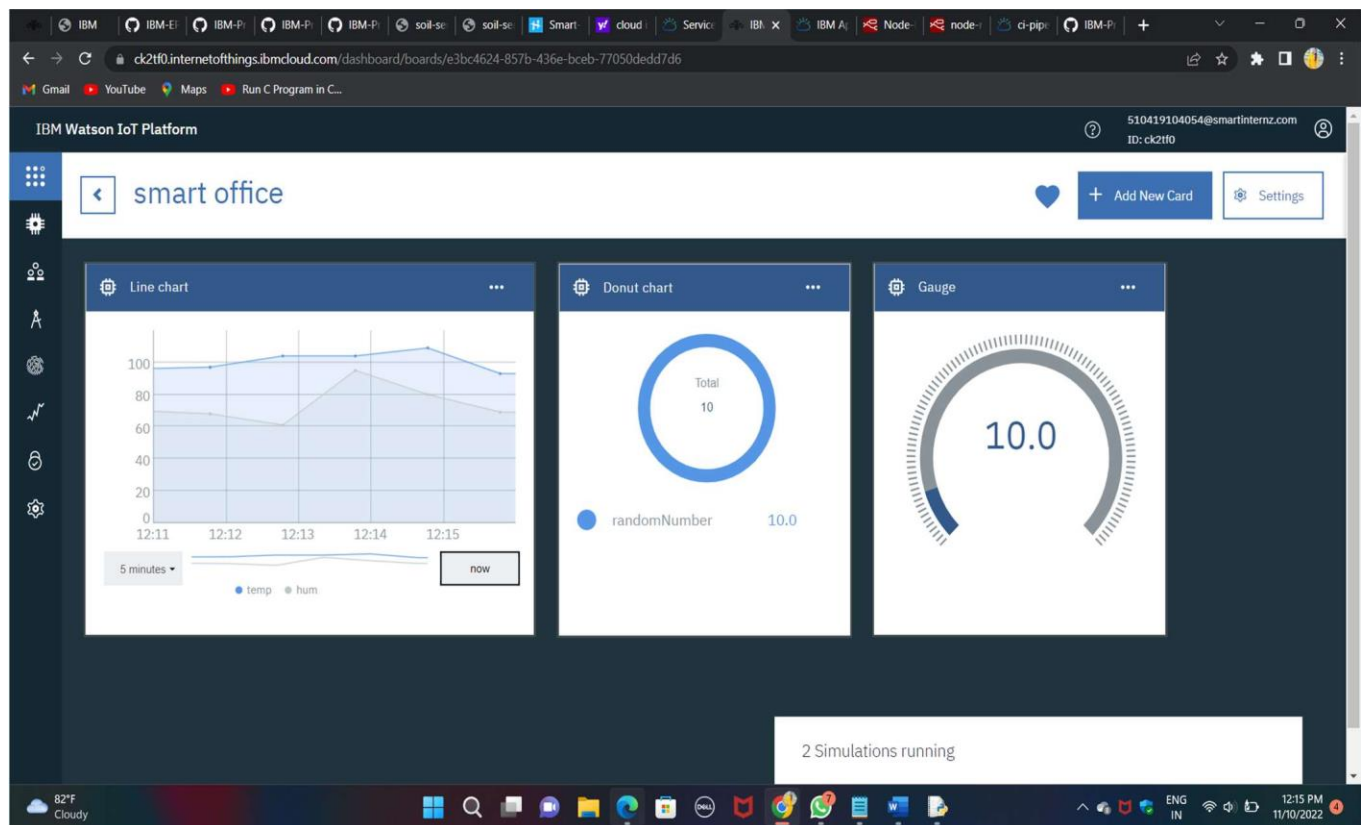
- Open link provided in below image
- Give the credentials of your device in IBM Watson
- Platform Click on connect

My credentials given to simulator are:

Api: **a-ck2tf0-yutwjnphx**

Device type: 1234

Token: **MaZKMomT_thHLD54ml**



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)

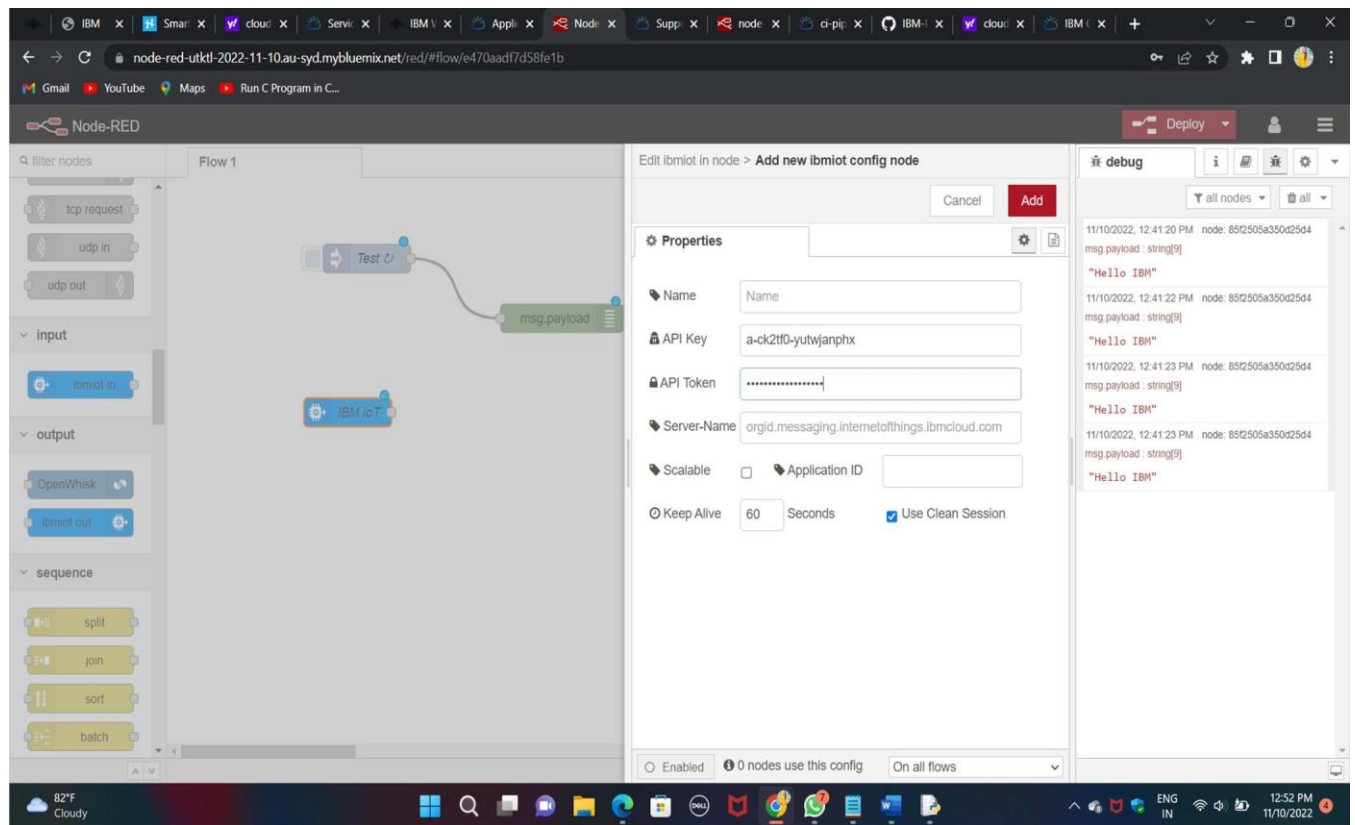
```
{  
  "d": {  
    ▪ "name": "abcd",  
    ▪ "temperature": 17,  
    ▪ "humidity": 76,  
    ▪ "Moisture ": 25  
  }  
}
```

The screenshot displays the IBM Watson IoT Platform interface. The main dashboard shows a list of devices under the 'Browse' tab. Two devices are listed: '1234' and '12345', both of type 'NodeMCU' and status 'Disconnected'. The '12345' device is selected, and its 'Recent Events' tab is active. Below the device information, a table lists recent events:

Event	Value	Format	Last Received
eventflow	{"randomNumber":17,"temp":103,"hum":91}	json	a few seconds ago
eventflow	{"randomNumber":9,"temp":109,"hum":66}	json	a few seconds ago
eventflow	{"randomNumber":77,"temp":101,"hum":98}	json	a few seconds ago

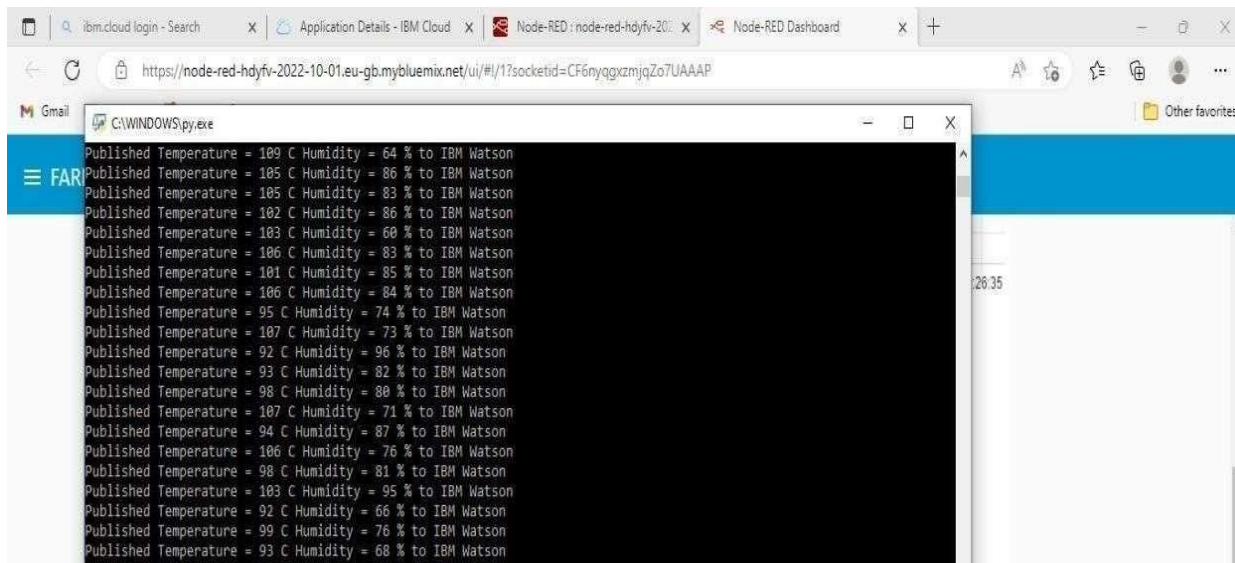
An overlay modal titled 'Device Type: NodeMCU' is open, showing the 'EVENTS' configuration. It includes an 'Event type name' field set to 'eventflow', a 'Schedule' dropdown set to 'Every Minute', and a 'Payload' editor. The payload is a JSON object with random values for 'randomNumber', 'temp', and 'hum'. The modal also has 'Send', 'Upload a CSV file', 'Cancel', and 'Save' buttons.

Configuration of Node-Red to collect IBM cloud data

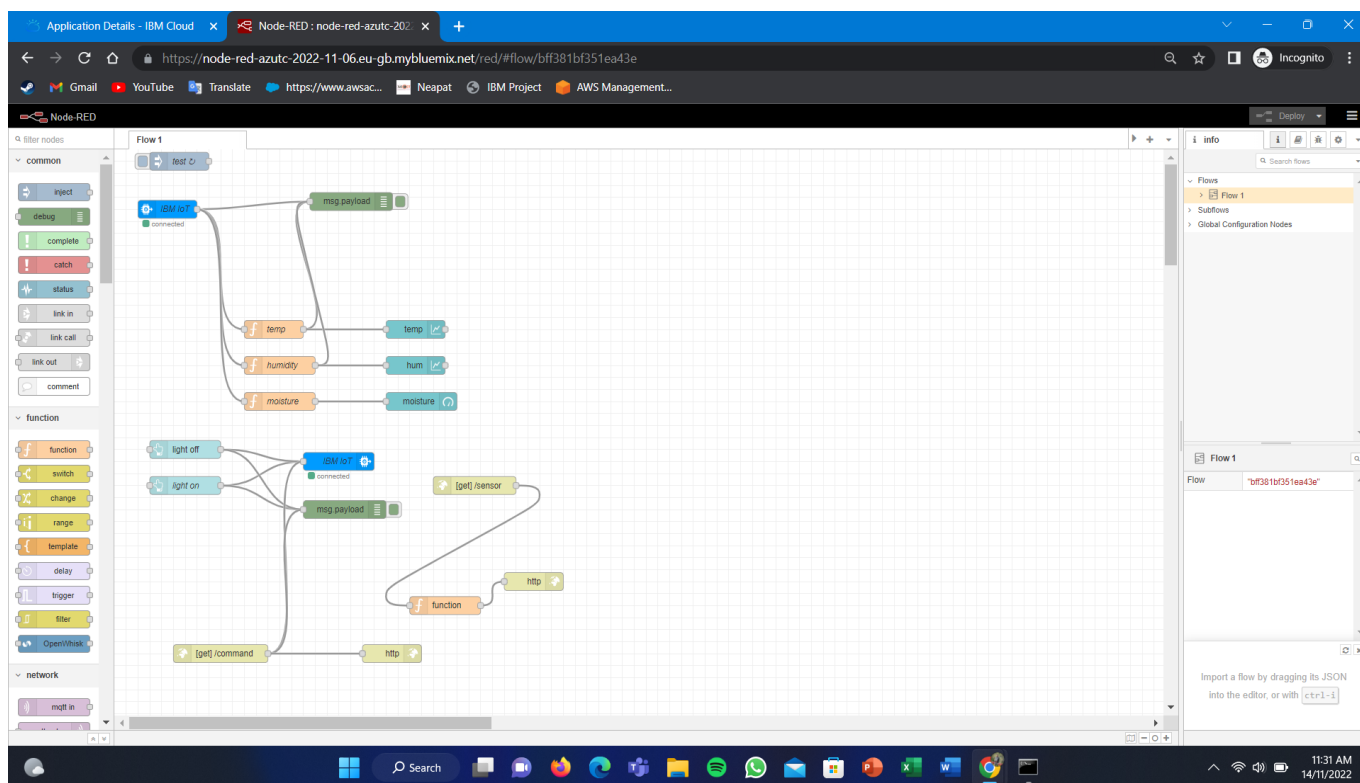


- The node IBM IoT App In is added to Node-Red workflow. Then the appropriate device credentials obtained earlier are entered into the node to connect and fetch device telemetry to Node-Red.
- Once it is connected Node-Red receives data from the device
Display the data using debug node for verification
- Connect function node and write the Java script code to get each reading separately.
- The Java script code for the function node is:

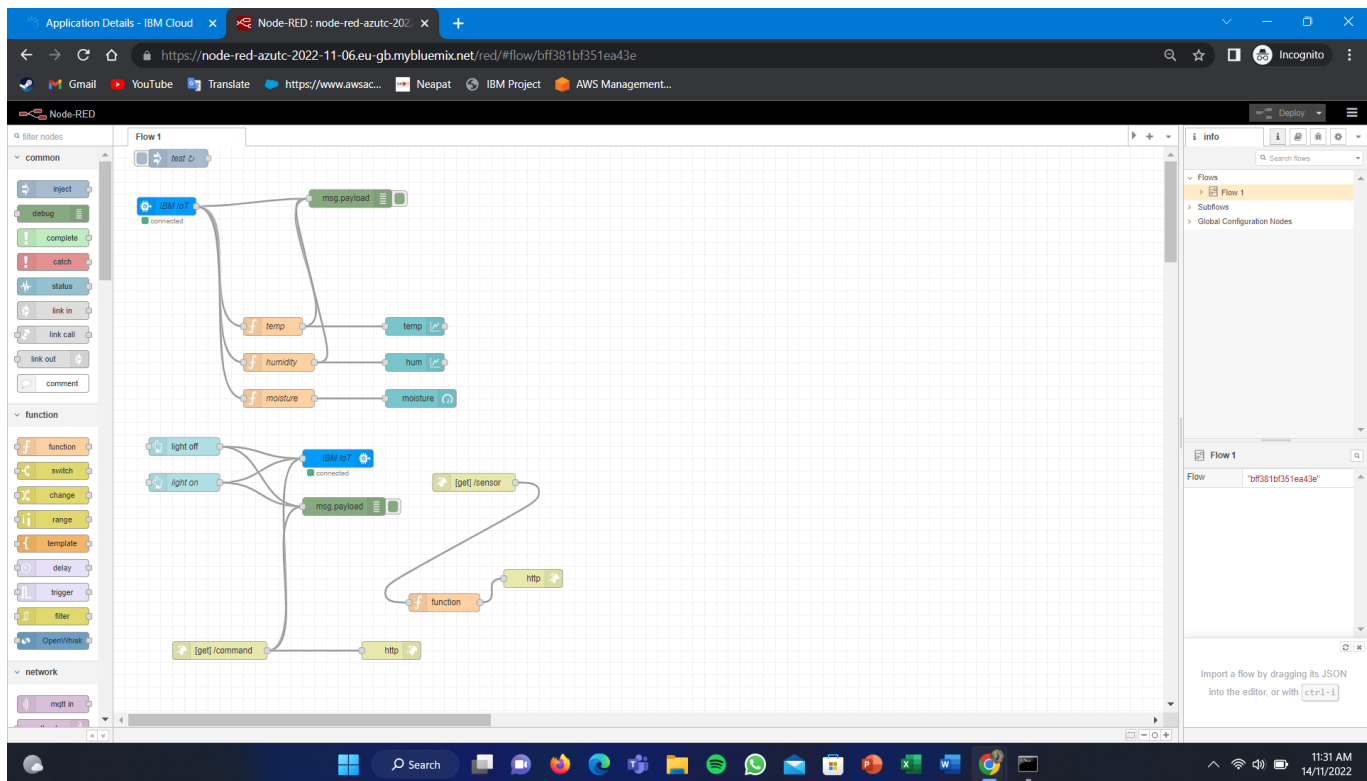
```
msg.payload=msg.payload.d.temperature  
return msg;
```
- Finally connect Gauge nodes from dashboard to see the data in UI



➤ 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.

Configuration of Node-Red to collect data from OpenWeather

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": 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

