

SPRINT – 2

Date	05 NOVEMBER 2022
Team ID	PNT2022TMID04663
Project Name	Smart Farmer-IoT Enabled smartFarming Application

Connecting IOT Simulator to IBM Watson IOT Platform

Give the credentials of your device in IBM Watson

My credentials given to simulator are:

OrgID:**yy3qcm**

Device type: **ibm22**

Device ID : **123**

Token: **12345678**

- 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)

```
{
  "Moisture":77,
  "temp":53,
  "Humid":30
}
```

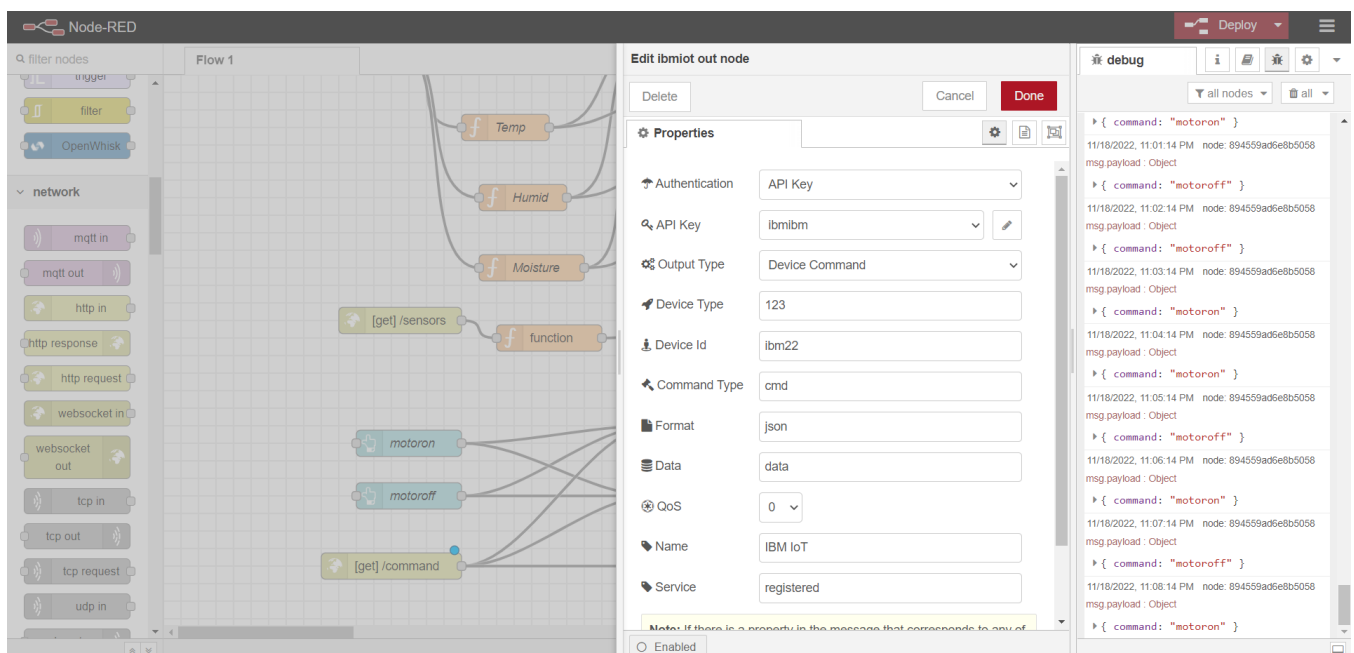
The screenshot displays the IBM Watson IoT Platform interface. The top navigation bar includes 'Browse', 'Action', 'Device Types', and 'Interfaces'. A sidebar on the left contains various icons for navigation. The main content area shows a list of devices. The selected device, 'ibm22', is shown in a detailed view with tabs for 'Identity', 'Device Information', 'Recent Events', 'State', and 'Logs'. The 'Recent Events' tab is active, displaying a table of recent data events.

Event	Value	Format	Last Received
IoTSensor	{"temp":38,"Humid":71,"moisture":71}	json	a few seconds ago
IoTSensor	{"temp":27,"Humid":19,"moisture":63}	json	a few seconds ago
IoTSensor	{"temp":52,"Humid":34,"moisture":100}	json	a few seconds ago
IoTSensor	{"temp":54,"Humid":92,"moisture":62}	json	a few seconds ago
IoTSensor	{"temp":58,"Humid":95,"moisture":19}	json	a few seconds ago

Below the main device view, another device 'esp8266' is partially visible. The bottom of the interface shows pagination controls: 'Items per page 50' and '1-2 of 2 items'.

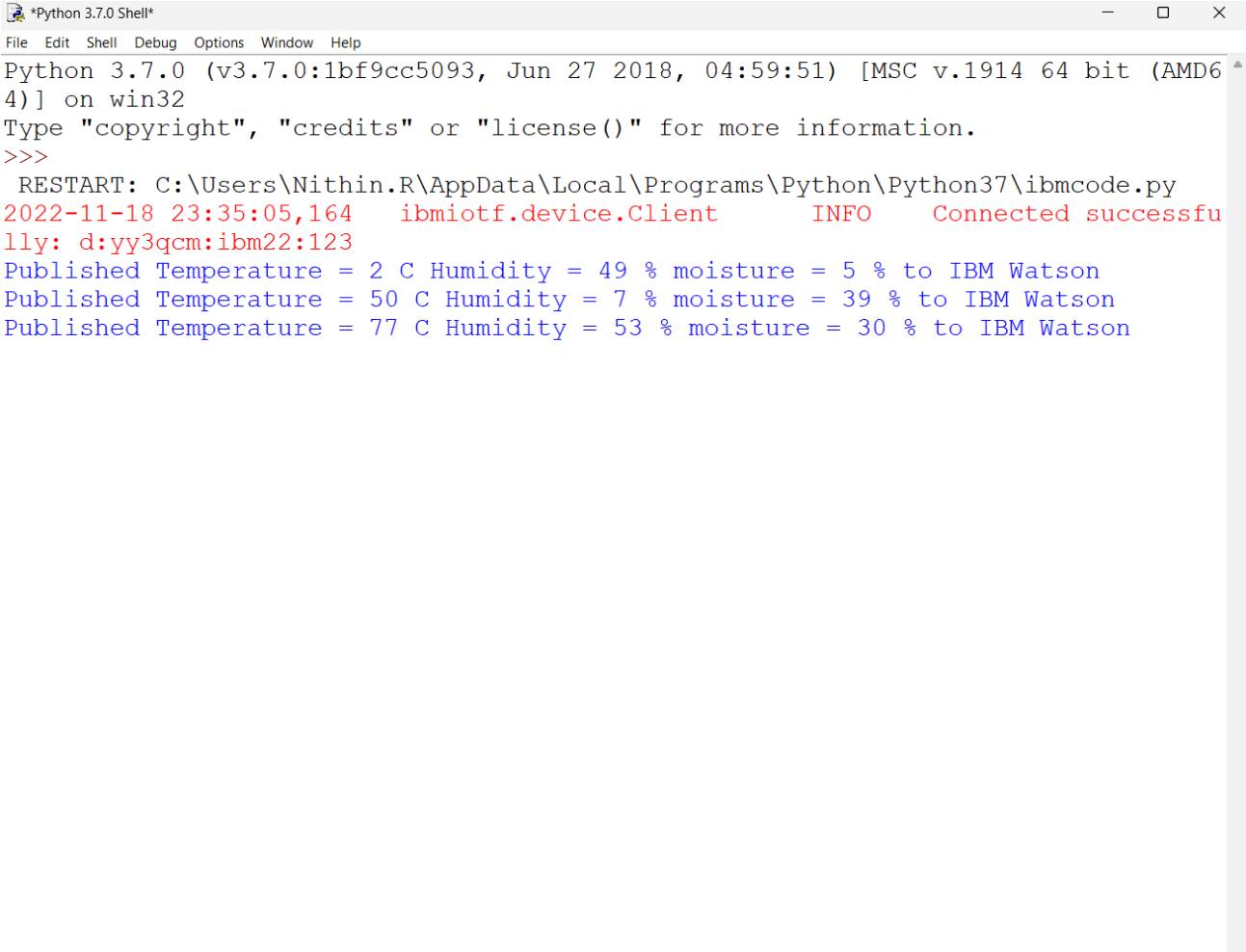
Configuration of Node-Red to collect IBM cloud data

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.



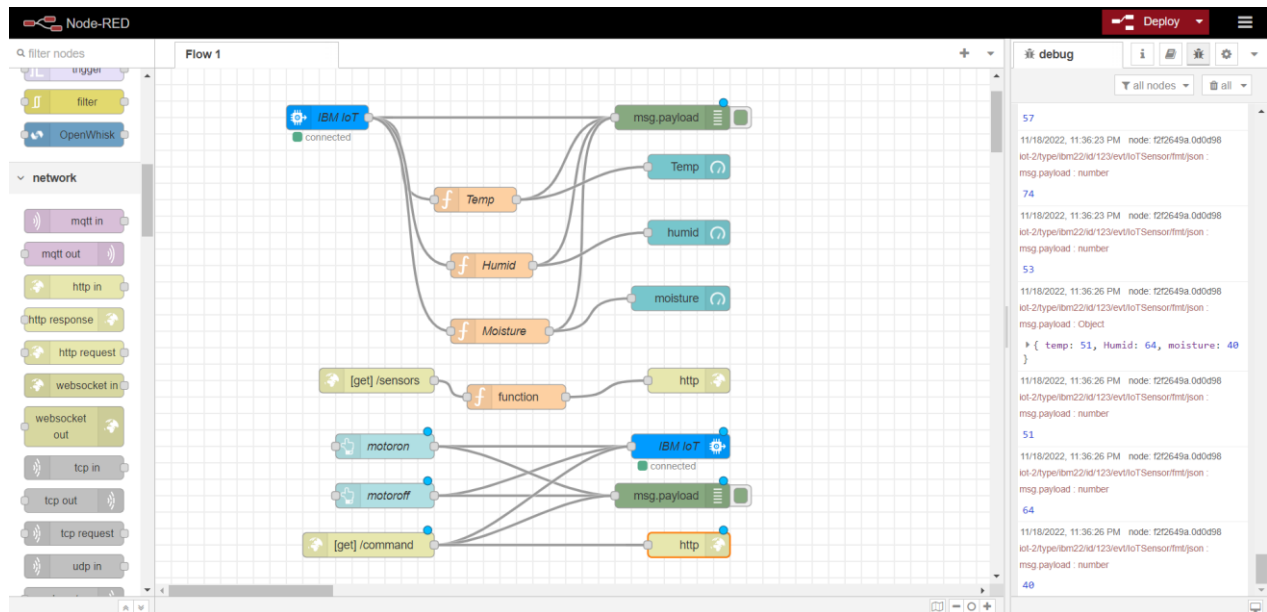
- Once it is connected Node-Red receives data from the device.
- Display the data using debug node for verification.
- Connect function node and write the Java script code to get each reading separately.
- The Java script code for the function node is:
msg.payload = msg.payload.temp
return msg;
- Finally connect Gauge nodes from dashboard to see the data in UI.

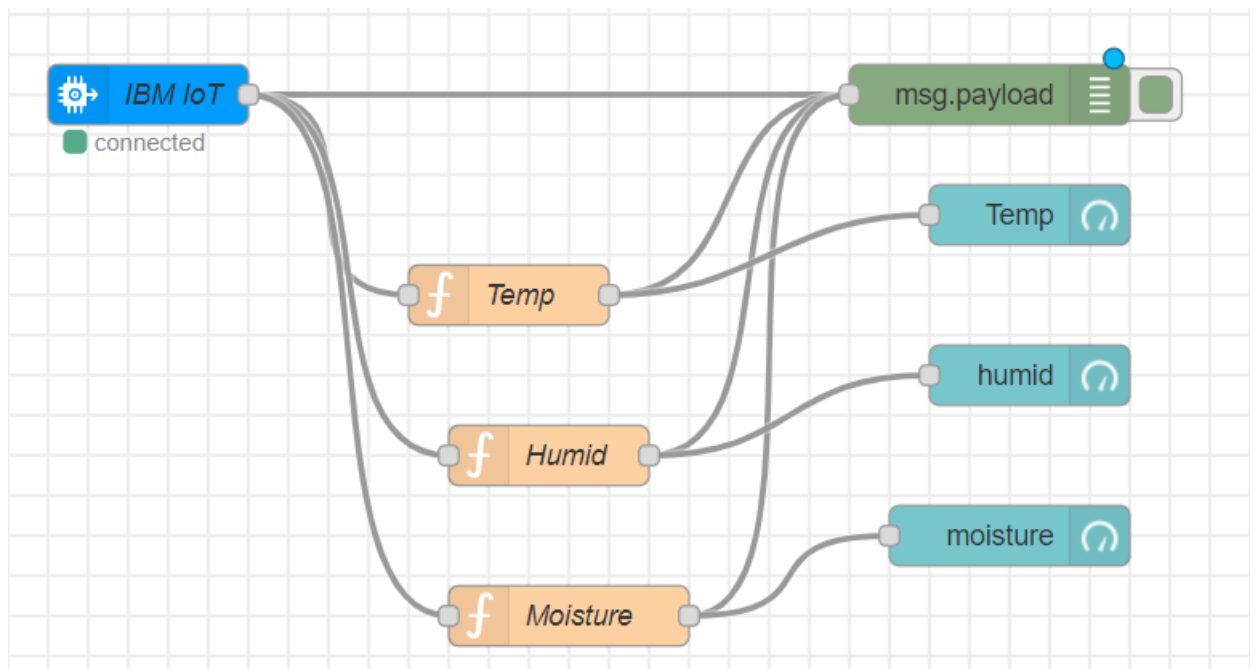
- Data send by the python code



```
*Python 3.7.0 Shell*
File Edit Shell Debug Options Window Help
Python 3.7.0 (v3.7.0:1bf9cc5093, Jun 27 2018, 04:59:51) [MSC v.1914 64 bit (AMD64)] on win32
Type "copyright", "credits" or "license()" for more information.
>>>
RESTART: C:\Users\Nithin.R\AppData\Local\Programs\Python\Python37\ibmcode.py
2022-11-18 23:35:05,164 ibmiotf.device.Client INFO Connected successfully: d:yy3qcm:ibm22:123
Published Temperature = 2 C Humidity = 49 % moisture = 5 % to IBM Watson
Published Temperature = 50 C Humidity = 7 % moisture = 39 % to IBM Watson
Published Temperature = 77 C Humidity = 53 % moisture = 30 % to IBM Watson
```

- Data received from the cloud in Node-Red console





- Nodes connected in following manner to get each reading separately .

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.
- The link to get open weather API :
<https://api.openweathermap.org/data/2.5/weather?lat=11.4383197&lon=77.5402674&appid=124d808d2039542453a0b1b05f37e900>
- The data we receive from Open Weather after request is in below JSON format.
- ```
{ "coord": { "lon": 77.5403, "lat": 11.4383 }, "weather": [{ "id": 804, "main": "Clouds", "description": "overcast clouds", "icon": "04d" }], "base": "stations", "main": { "temp": 300.33, "feels_like": 303.19, "temp_min": 300.33, "temp_max": 300.33, "pressure": 1009, "humidity": 79, "sea_level": 1009, "grnd_level": 986 }, "visibility": 10000, "wind":
```

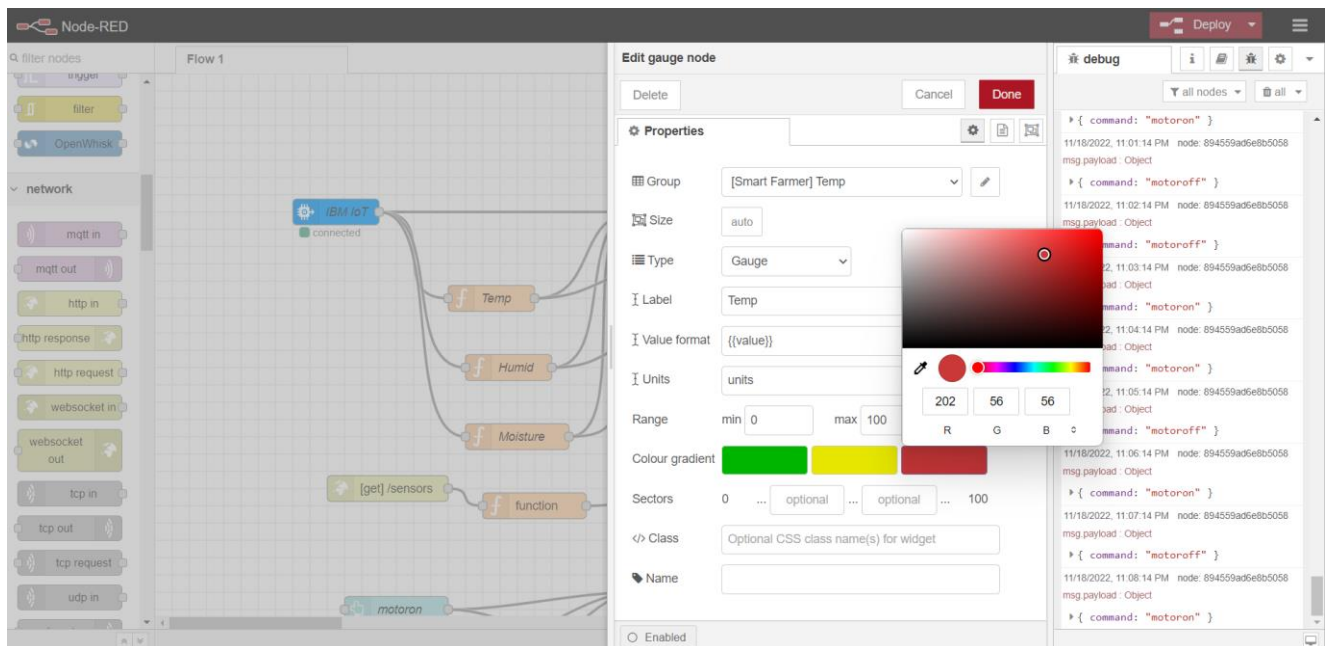
```
{ "speed":2.3,"deg":113,"gust":3.05},"clouds":{"all":97},"dt":1668332957
,"sys":{"country":"IN","sunrise":1668300334,"sunset":1668342165},"tim
ezone":19800,"id":1270947,"name":"Gobichettipalayam","cod":200}
```

- In order to parse the JSON string we use Java script functions and get each parameters

```
msg.payload = { "temp" : global.get("t") ,
 "Humid" : global.get("h") ,
 "moisture" : global.get("m")
 }

return msg;
```

- Then we add Gauge and text nodes to represent data visually in UI.



- You can the data in the node-red dashboard.

