

Project Delivery Sprint - 2

Date	28 Oct 2022
Team ID	PNT2022TMID04699
Project Name	Smart Farmer - IoT Enabled Smart Farming Application

Connecting IOT Simulator to IBM Watson IOT Platform

Give the credentials of your device in IBM WatsonMy

credentials given to simulator are:

organization = "95a96q"

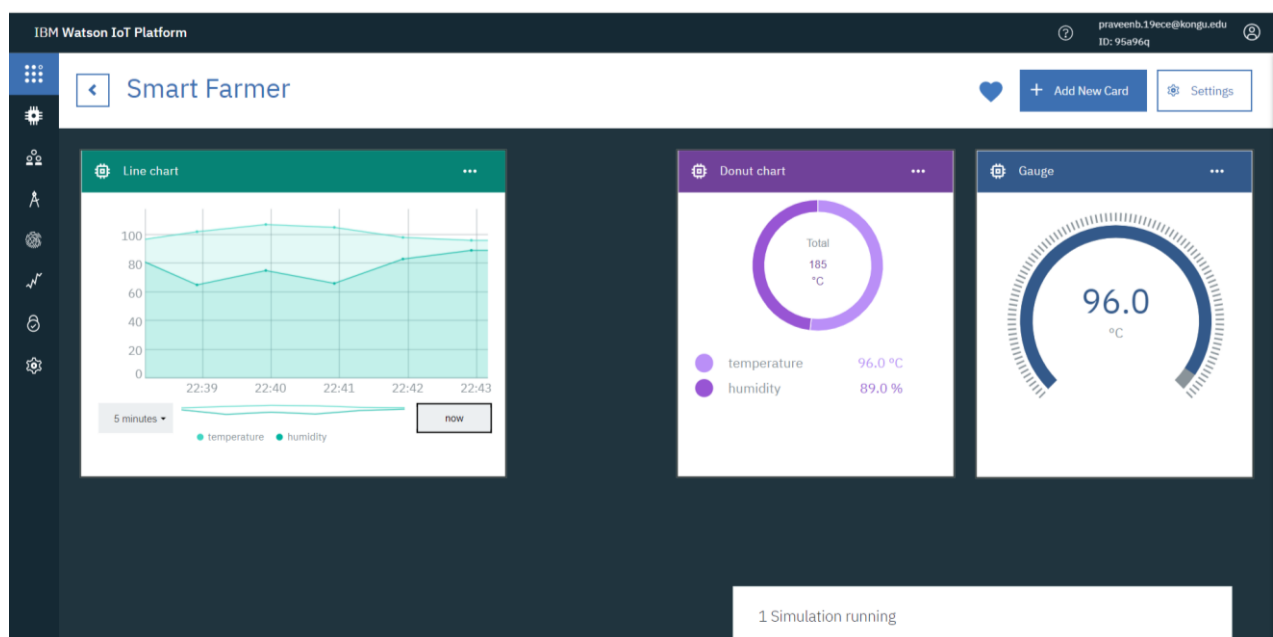
deviceType = "NodeMCu"

deviceId = "123456"

authMethod = "use-token-auth"

authToken = "P123@456"

- 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)

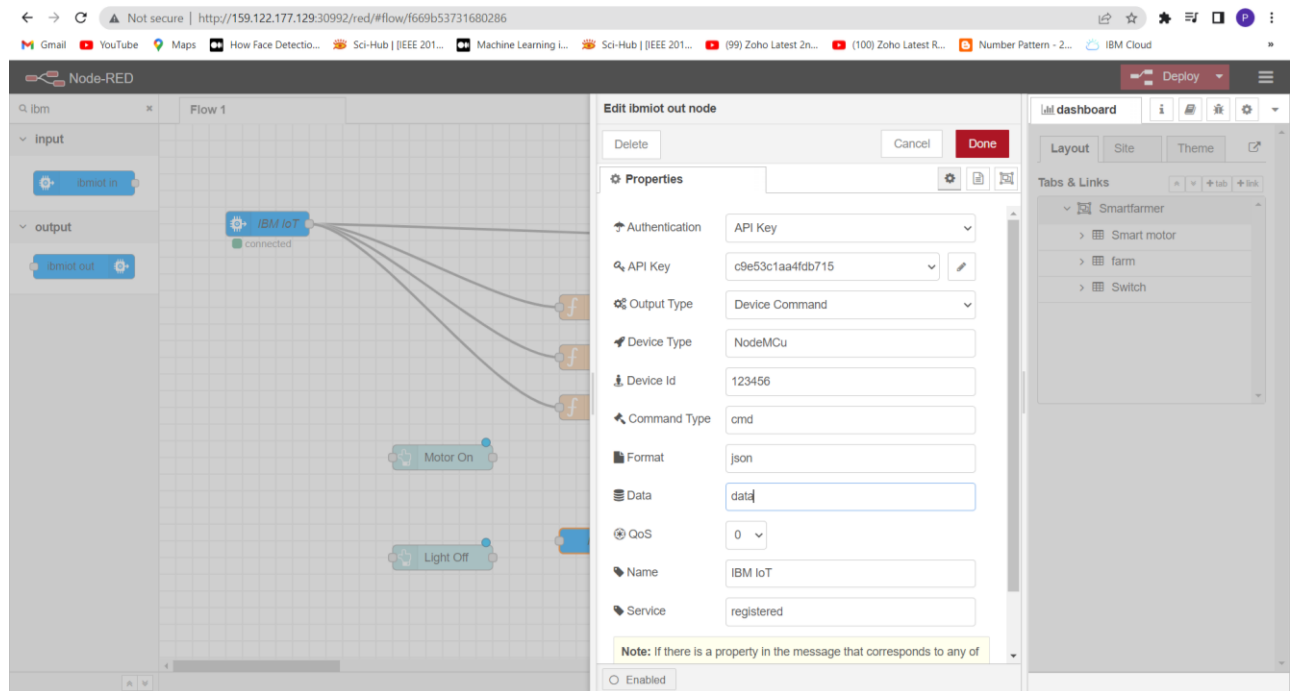
```
{
  "Moisture":89,
  "temp":96.0,
  "Humid":89
}
```

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 device card for '123456' with status 'Disconnected' and type 'NodeMCu'. Below the card, the 'Recent Events' tab is selected, showing a table of live data events.

Event	Value	Format	Last Received
IoTSensor	{"temp":96,"Humid":79}	json	a few seconds ago
IoTSensor	{"temp":103,"Humid":98}	json	a few seconds ago
IoTSensor	{"temp":104,"Humid":82}	json	a few seconds ago
eventflow	{"randomNumber":17,"temperature":102,"humi...	json	a few seconds ago
IoTSensor	{"temp":107,"Humid":87}	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 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.

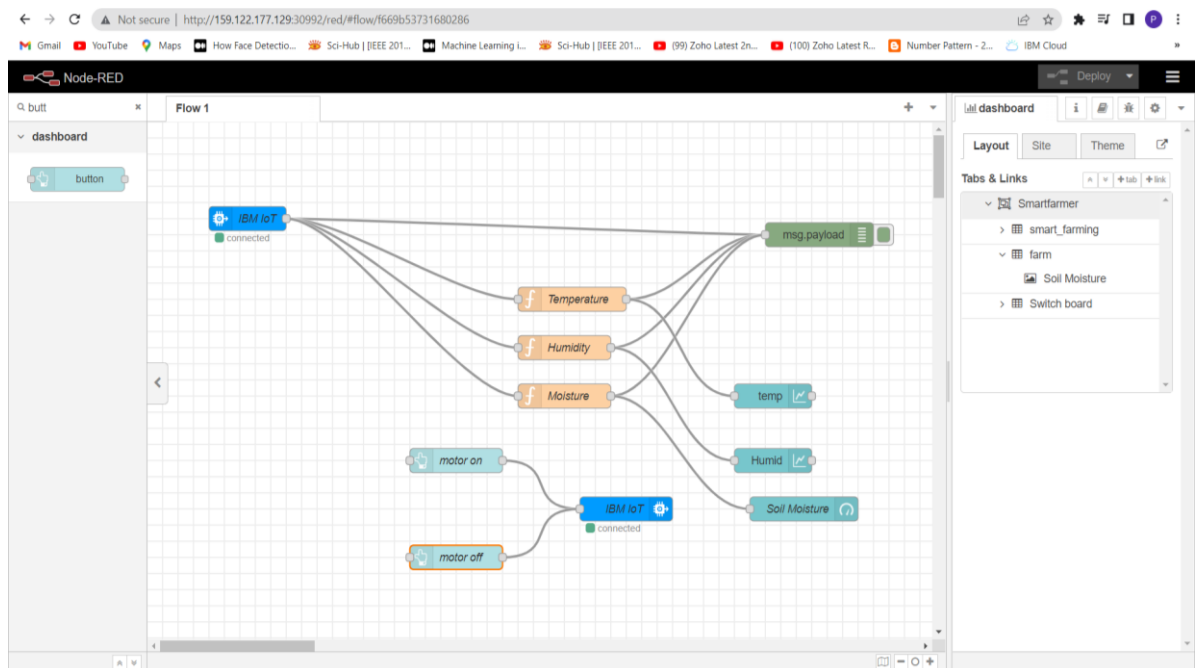
- Data send by the python code

```

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\hp\Downloads\ibms1.py =====
2022-11-18 10:35:42,016 ibmiotf.device.Client INFO Connected successfully: d:95a96q:NodeMCu:123456
Published Temperature = 108 C Humidity = 91 % to IBM Watson
Published Temperature = 99 C Humidity = 68 % to IBM Watson
Published Temperature = 95 C Humidity = 97 % to IBM Watson
Published Temperature = 107 C Humidity = 87 % to IBM Watson
Published Temperature = 104 C Humidity = 82 % to IBM Watson
Published Temperature = 103 C Humidity = 98 % 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 HTTPGET 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.

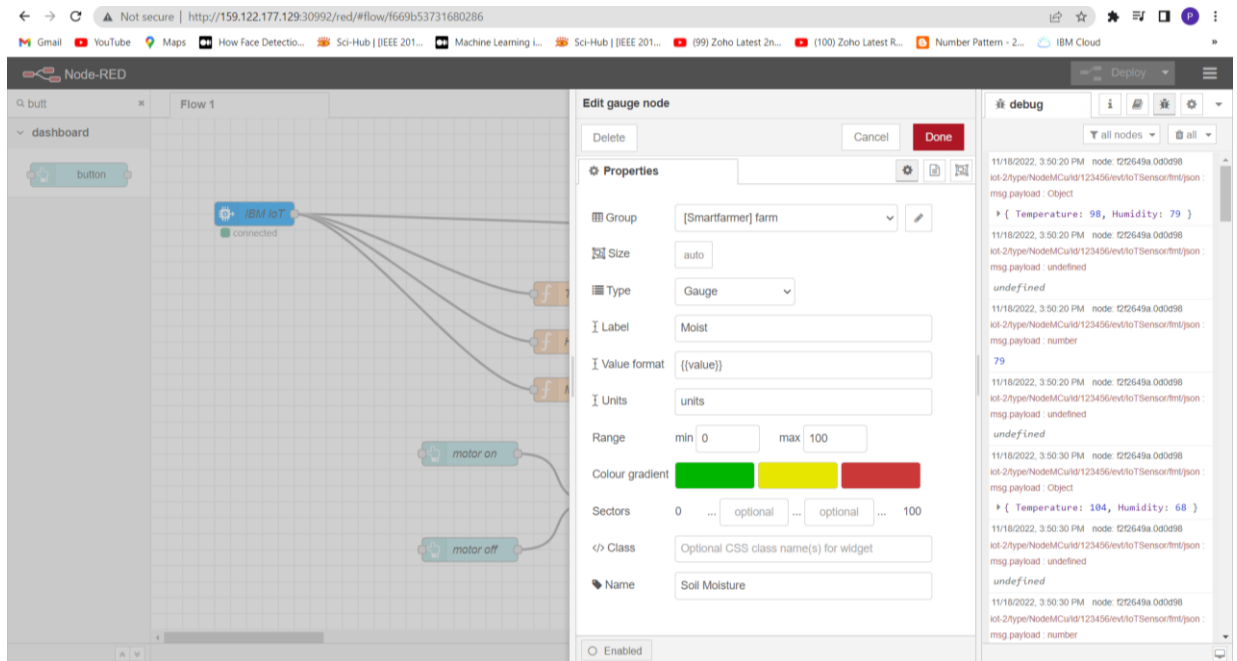
- ```
{"coord":{"lon":77.5403,"lat":11.4383},"weather":[{"id":804,"main":"Clouds","description":"overcast clouds","icon":"04d"}],"base":"stations","main":{"temp":300.33,"feels_like":303.19,"temp_min":300.33,"temp_max":300.33,"pressure":1009,"humidity":79,"sea_level":1009,"grnd_level":986},"visibility":10000,"wind":{"speed":2.3,"deg":113,"gust":3.05},"clouds":{"all":97},"dt":1668332957,"sys":{"country":"IN","sunrise":1668300334,"sunset":1668342165},"timezonedata":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

