

SMART FARMER-IOT ENABLED SMART FARMING APPLICATION

SPRINT DELIVERY – 2

Date	19-11-2022
Team ID	PNT2022TMID34083
Project name	Smart farmer-IOT Enabled Smart Farming Application

Building Project :

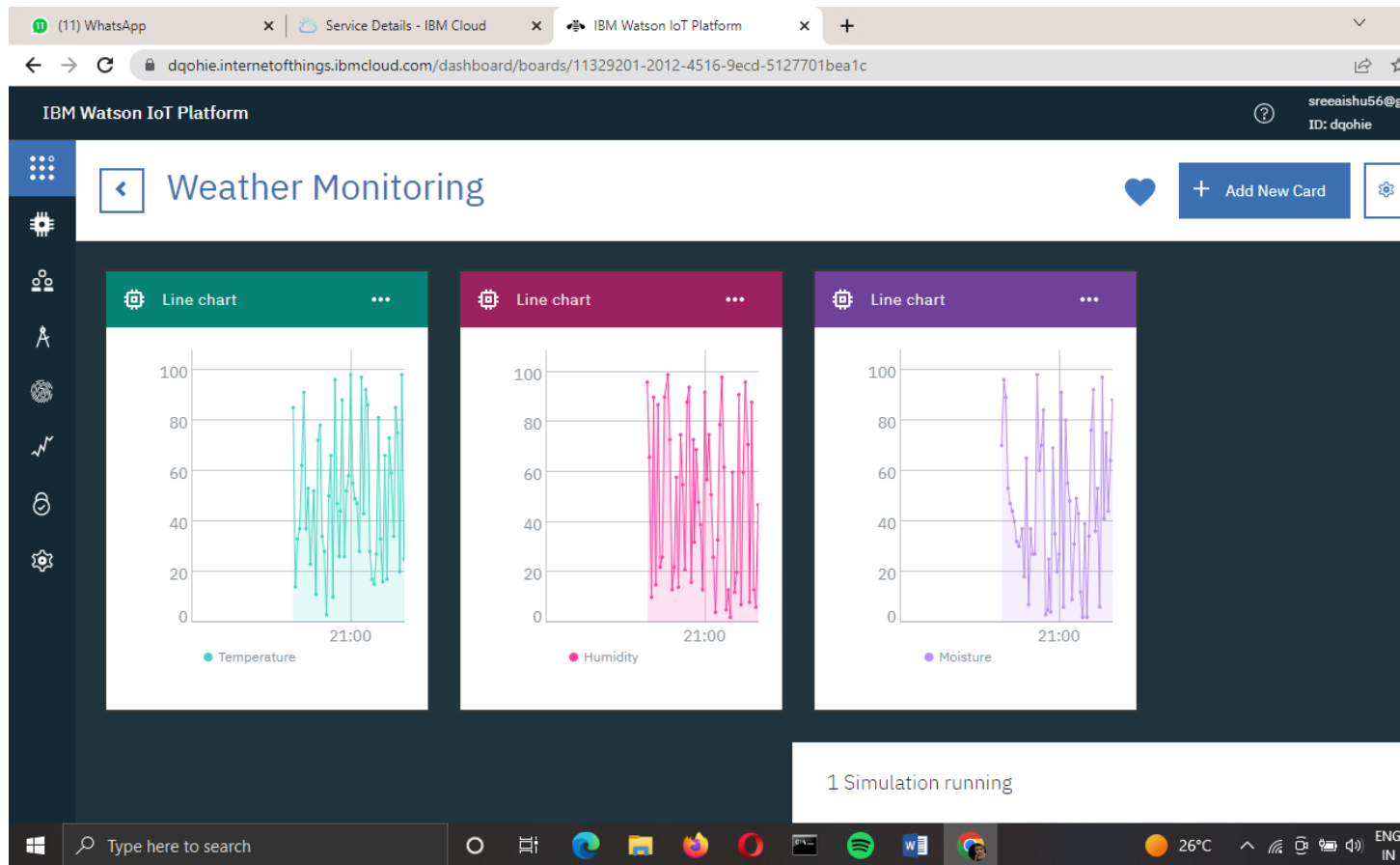
Connecting IoT Simulator to IBM Watson IoT Platform

- ☐ 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 key: a-dqohie-pwsmdt7u3s

Device type: PNT2022TMID34083

Token: 1)q-lg(IHXe8dL4tSe



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)

IBM Watson IoT Platform

Browse Action Device Types Interfaces

Search by Device ID

Device ID	Status	Device Type
PNTID34083	Disconnected	PNT2022TMD34083

Identity Device Information Recent Events State

The recent events listed show the live stream of data that is coming and going

Event	Value
event_1	{"Temperature":87,"Humidity":78,"Moisture":36}
event_1	{"Temperature":48,"Humidity":27,"Moisture":39}
event_1	{"Temperature":19,"Humidity":27,"Moisture":81}
event_1	{"Temperature":20,"Humidity":15,"Moisture":4}

Device Type: PNT2022TMD34083

Events 1

New event type +

Event type name event_1 Send

Schedule 20 Every Minute

Payload

Specify the event payload in the editor window or by uploading a CSV file.

```

0 {
1   "Temperature": random(0, 100),
2   "Humidity": random(0, 100),
3   "Moisture": random(0, 100),
4 }
5

```

Upload a CSV file

Cancel Save

Configuration of Node-Red to collect IBM cloud data

Node-RED

filter nodes

Flow 1

Flow 2

common

inject

debug

complete

catch

status

link in

link call

link out

comment

function

function

switch

IBM IoT

connected

msg.payload

Temperature

Humidity no

Moisture No

[get] /sensor

http f

Motor on

Motor off

[get] /control

Edit ibmiot in node > Edit ibmiot node

Delete Cancel Update

Properties

Name

API Key a-dqohie-pwsmdt7u3s

API Token

Server-Name orgid.messaging.internetofthings.ibmcloud.com

Scalable

Application ID

Keep Alive 60 Seconds

Use Clean Session

Enabled 2 nodes use this config On all flows

debug

selected nodes

all

iot-

2/type/PNT2022TMD34083/Id/PNTID34083/evt/event_1/fn

msg.payload: Object

```

{ Temperature: 34, Humidity: 99,
  Moisture: 23 }

```

11/19/2022, 9:06:47 PM node: 31e3d714cf7e3385

iot-

2/type/PNT2022TMD34083/Id/PNTID34083/evt/event_1/fn

msg.payload: Object

```

{ Temperature: 73, Humidity: 7,
  Moisture: 14 }

```

11/19/2022, 9:06:50 PM node: 31e3d714cf7e3385

iot-

2/type/PNT2022TMD34083/Id/PNTID34083/evt/event_1/fn

msg.payload: Object

```

{ Temperature: 31, Humidity: 22,
  Moisture: 96 }

```

11/19/2022, 9:06:53 PM node: 31e3d714cf7e3385

iot-

2/type/PNT2022TMD34083/Id/PNTID34083/evt/event_1/fn

msg.payload: Object

```

{ Temperature: 3, Humidity: 14,
  Moisture: 77 }

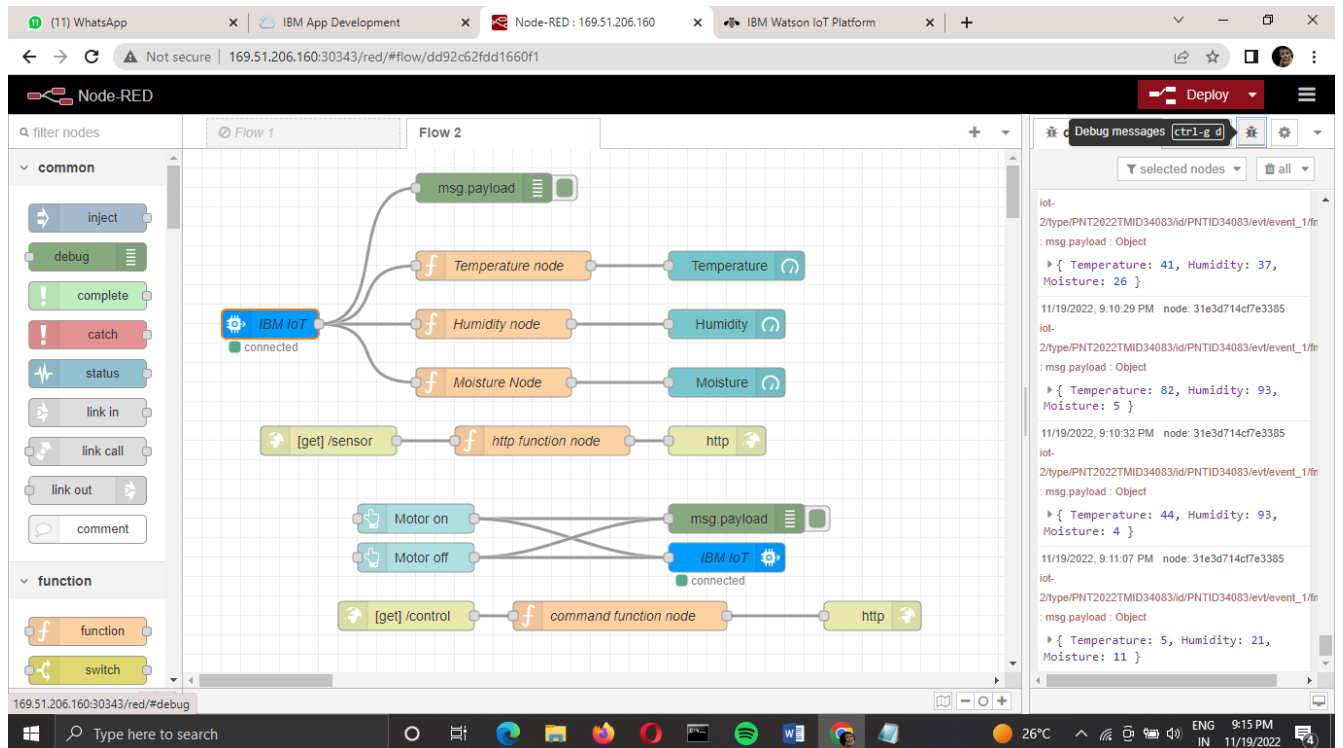
```

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

```
*Python 3.7.0 Shell*
File Edit Shell Debug Options Window Help
Python 3.7.0 (v3.7.0:1bf9cc5093, Jun 27 2018, 04:06:47) [MSC v.1914 32 bit (Intel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>>
RESTART: C:/Users/ADHARSH/AppData/Local/Programs/Python/Python37-32/pythn1.py
2022-11-18 15:04:50,897 ibmiotf.device.Client INFO Connected successfully: d:b4hkg6:12345:54321
Published Temperature = 79 C Humidity = 72 % to IBM Watson
Published Temperature = 48 C Humidity = 43 % to IBM Watson
Published Temperature = 19 C Humidity = 57 % to IBM Watson
Published Temperature = 21 C Humidity = 70 % to IBM Watson
Published Temperature = 79 C Humidity = 33 % to IBM Watson
Published Temperature = 17 C Humidity = 49 % to IBM Watson
Published Temperature = 98 C Humidity = 60 % to IBM Watson
Published Temperature = 96 C Humidity = 16 % to IBM Watson
Published Temperature = 18 C Humidity = 36 % to IBM Watson
Published Temperature = 91 C Humidity = 93 % to IBM Watson
Published Temperature = 41 C Humidity = 89 % to IBM Watson
Published Temperature = 74 C Humidity = 71 % to IBM Watson
Published Temperature = 85 C Humidity = 4 % to IBM Watson
Published Temperature = 12 C Humidity = 40 % to IBM Watson
Published Temperature = 93 C Humidity = 90 % to IBM Watson
Published Temperature = 37 C Humidity = 62 % 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 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":307
59,"feels_like":305.5,"temp_min":307.59,"temp_max":307.59,"pressure":1002,"h
umidity":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":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

The screenshot displays the Node-RED web interface in a browser. The main workspace shows a flow with an 'IBM IoT' node connected to a '[get] /sensor' node. A function node is selected, and its configuration panel is open. The function node is named 'Temperature node' and is set to 'On Message'. The JavaScript code in the function node is as follows:

```
1 msg.payload=msg.payload.Temperature
2 global.set("t",msg.payload)
3 return msg;
```

The debug console on the right shows a series of messages received from the IoT node, each containing a payload object with Temperature, Humidity, and Moisture values. The messages are timestamped and include the node ID '31e3d714cf7e3385'.

Timestamp	Node ID	Temperature	Humidity	Moisture
11/19/2022, 9:10:29 PM	31e3d714cf7e3385	41	37	26
11/19/2022, 9:10:32 PM	31e3d714cf7e3385	82	93	5
11/19/2022, 9:11:07 PM	31e3d714cf7e3385	44	93	4
11/19/2022, 9:11:07 PM	31e3d714cf7e3385	5	21	11