

Smart Farmer-IOT Enabled Smart Farming Application

SPRINT DELIVERY – 2

Project	Smart Farmer-IOT Enabled Smart Farming Application
DOMAIN NAME	INTERNET OF THINGS
TEAM ID	PNT2022TMID35124
Date	11 November 2022

Building Project

Connecting IoT Simulator to IBM Watson IoT Platform

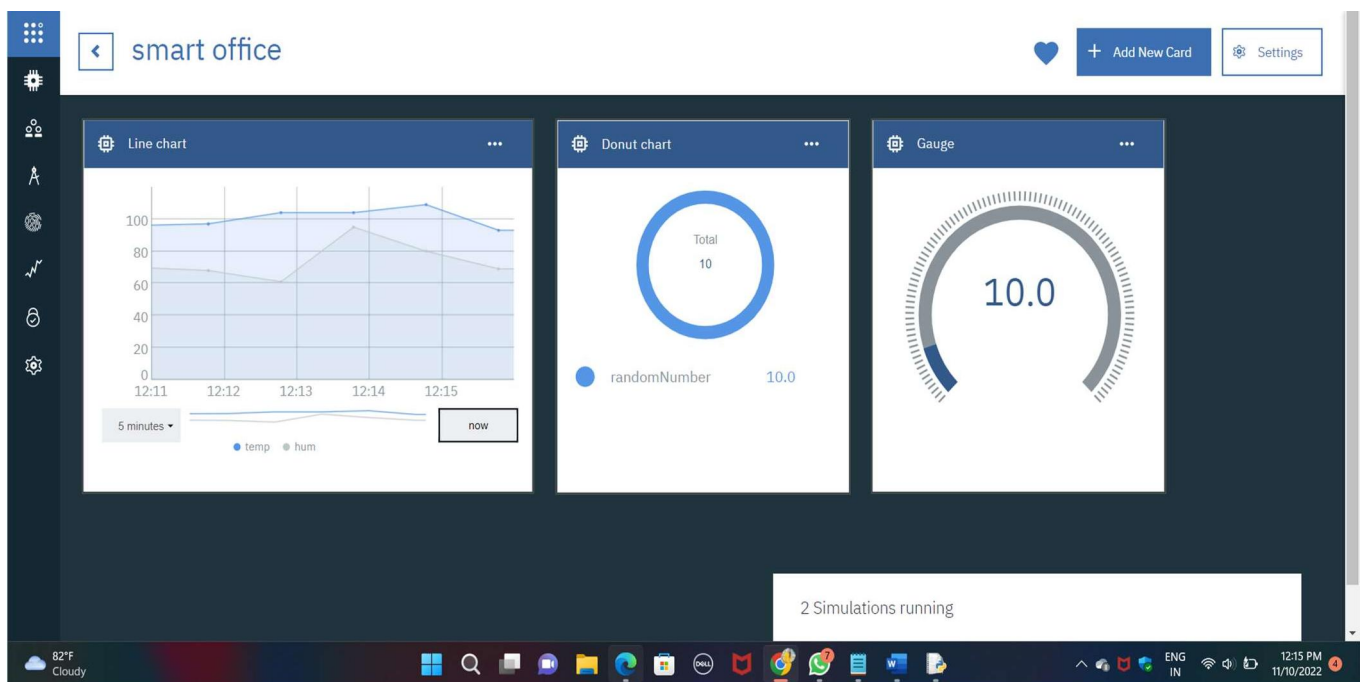
- Open link provided in below image
- Give the credentials of your device in IBM Watson
- PlatformClick on connect

My credentials given to simulator are:

api:**a-ocfkdz-tqg6ytwcvz**

Device type: **abcdef**

Token: **@EkWvqj7n)4bq@kW@h**



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 view shows a list of devices, with device 12345 selected. The 'Recent Events' tab is active, displaying a table of events. A modal window is open for configuring a new event type named 'eventflow' for a NodeMCU device. The modal includes a schedule setting of 'Every Minute' and a payload editor showing a JSON payload with random values for 'randomNumber', 'temp', and 'hum'.

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

Device Type: NodeMCU

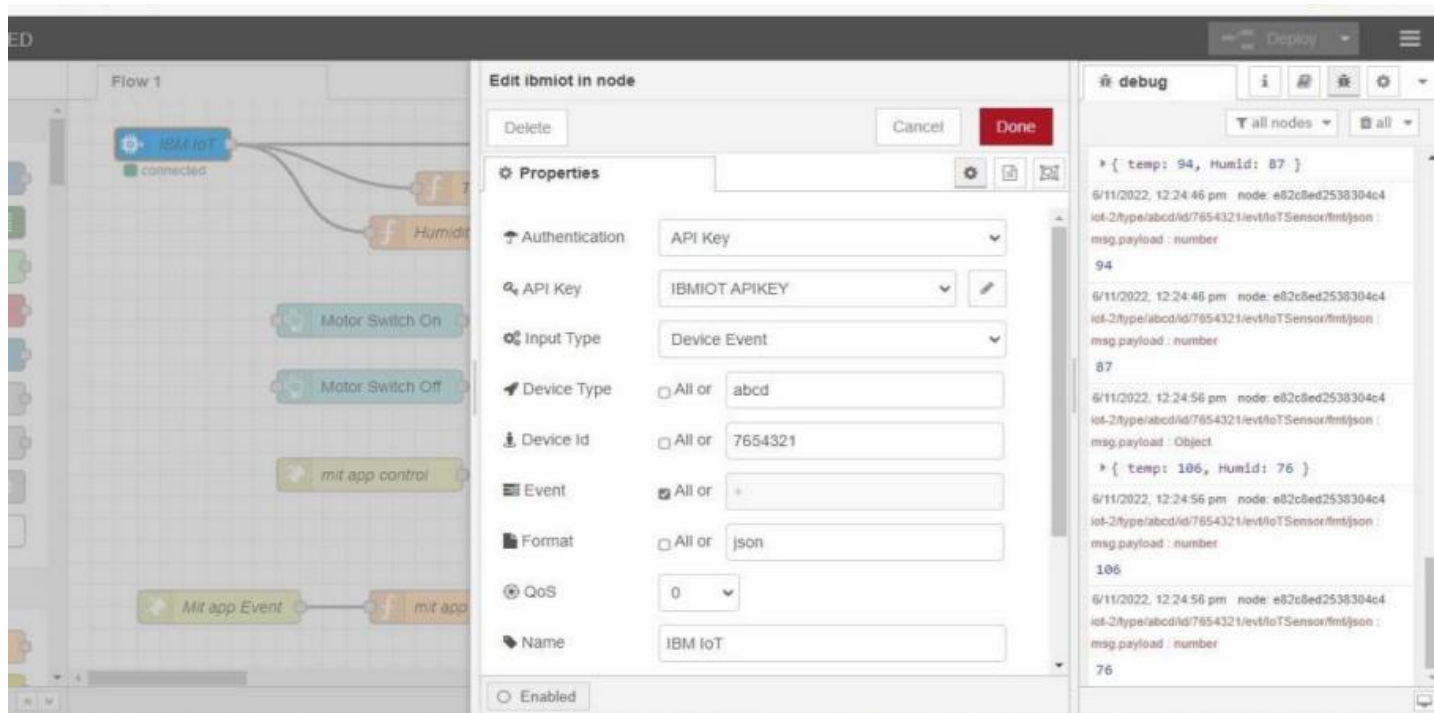
Event type name: eventflow

Schedule: 1 Every Minute

Payload:

```
{  
  "randomNumber": random(0, 100),  
  "temp": random(90, 110),  
  "hum": random(60, 100)  
}
```

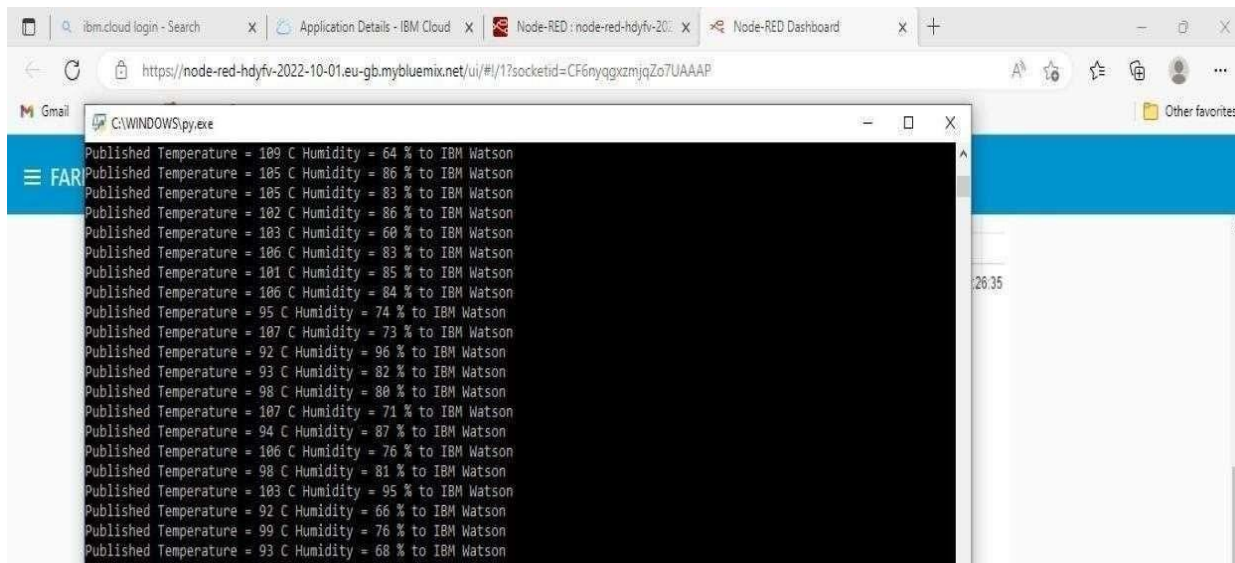
Buttons: Send, Upload a CSV file, Cancel, Save



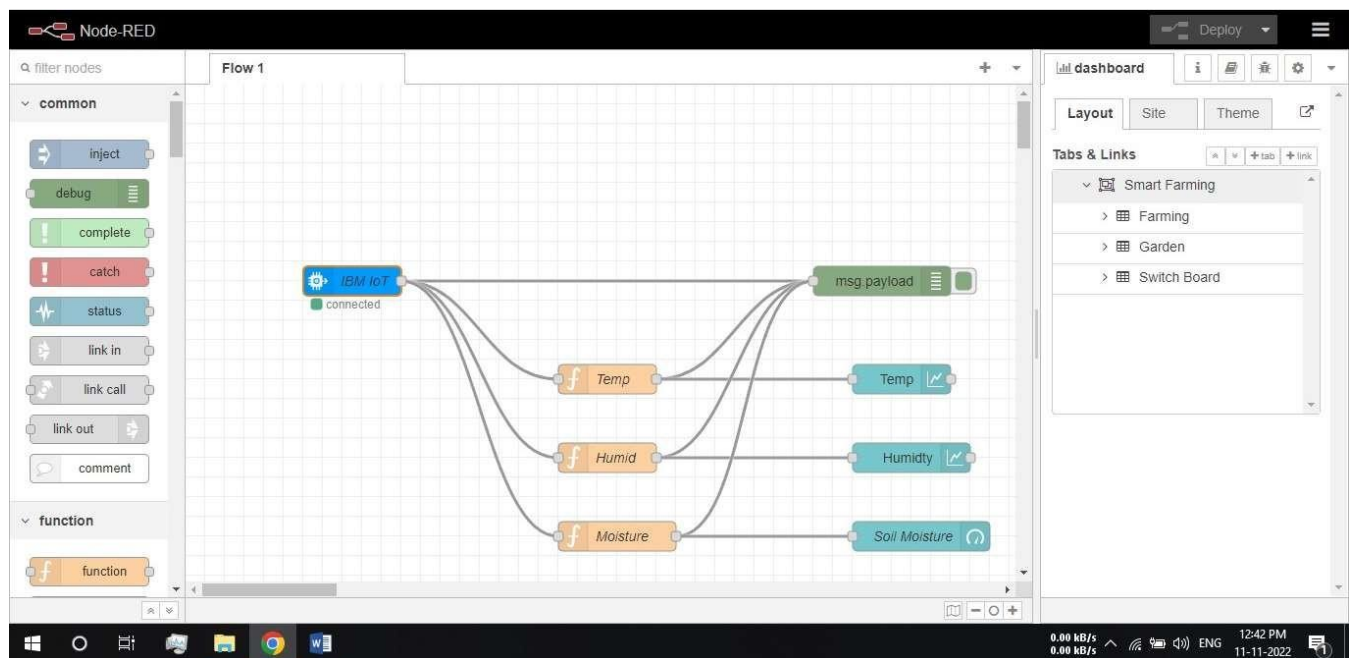
Configuration of Node-Red to collect IBM cloud data

- 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:

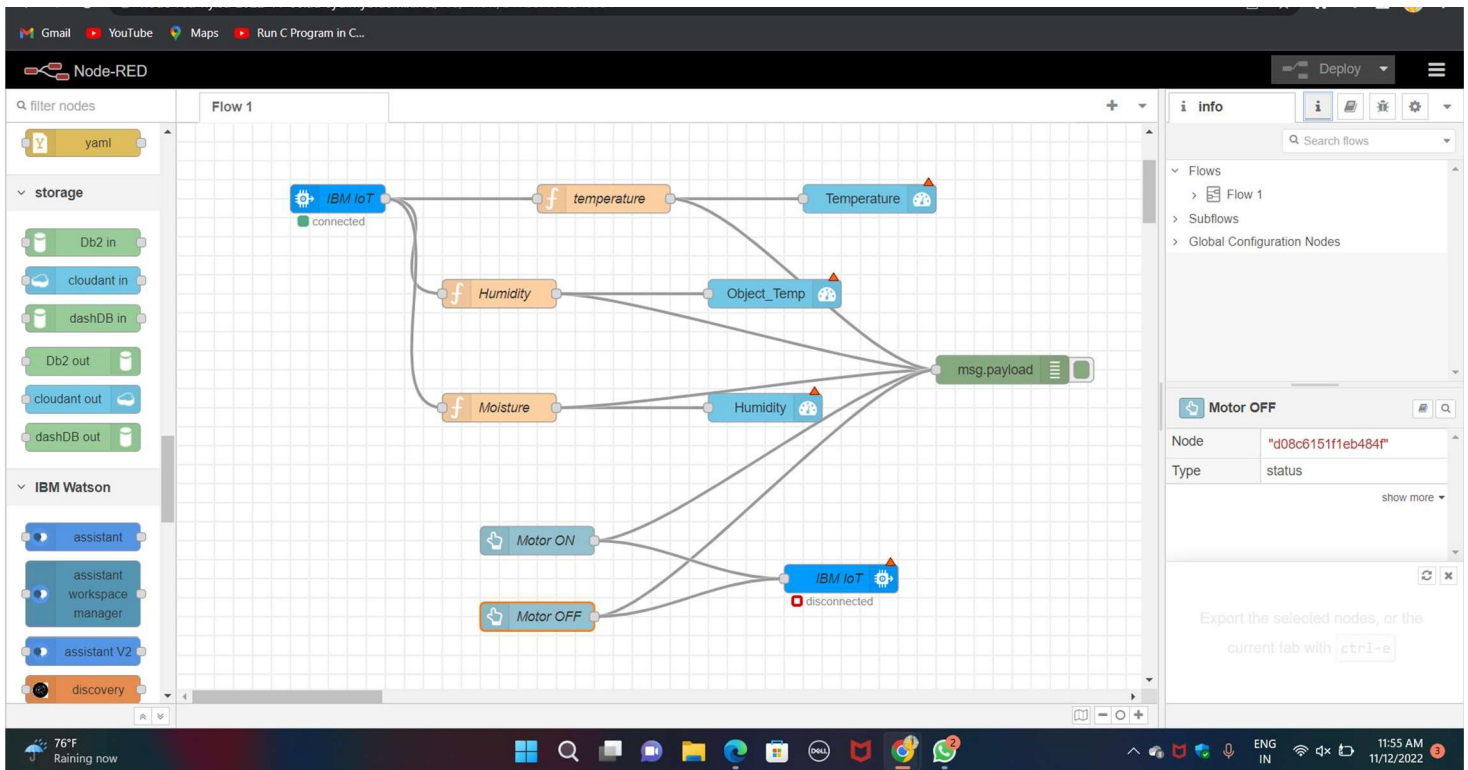
```
msg.payload=msg.payload.d.temperature
return msg;
```
- Finally connect Gauge nodes from dashboard to see the data in UI



➤ 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 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

