

VIVEKANANDHA COLLEGE OF ENGINEERING FOR WOMEN

Department of Computer Science and Engineering

Smart Farmer-IOT Enabled Smart Farming Application

SPRINT DELIVERY – 2

TITLE	Smart Farmer-IOT Enabled Smart Farming Application
DOMAIN NAME	INTERNET OF THINGS
TEAM ID	PNT2022TMID54280
LEADER NAME	JEEVITHRA J
TEAM MEMBER NAME	SIRINITH S HINDHUJA K S KANIKOZHI K
MENTOR NAME	GNANAMURUGAN S

Building Project

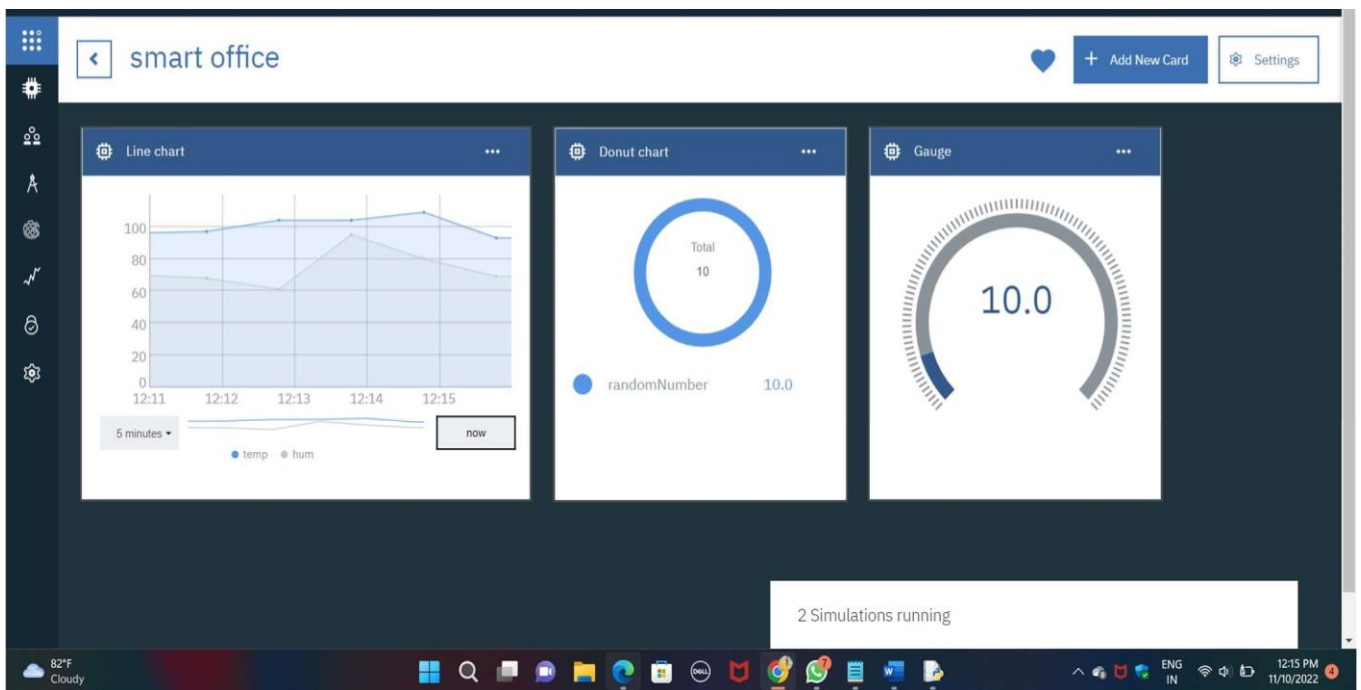
Connecting IoT Simulator to IBM Watson IoT Platform

- ☐ 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:

ORGANIZATION ID: u4ovmu

Device type: 12345

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)

{

```

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

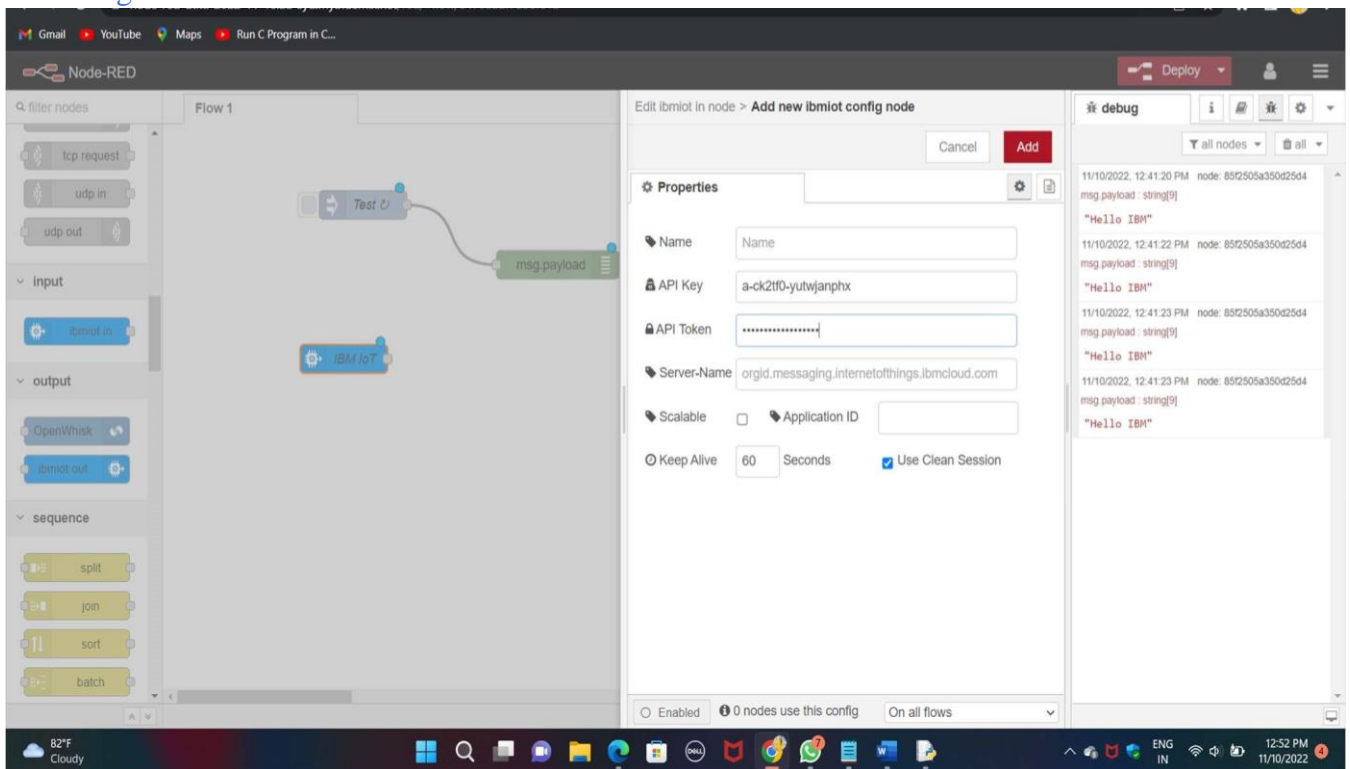
```

The screenshot displays the IBM Watson IoT Platform interface. The main window 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 tabs, a message states: 'The recent events listed show the live stream of data that is coming and going from this device.' A table shows three recent 'eventflow' events, each with a JSON payload containing 'randomNumber', 'temp', and 'hum' values.

An overlay window titled 'Device Type: NodeMCU' is open, showing the 'EVENTS' configuration. The 'Event type name' is 'eventflow'. The 'Schedule' is set to '1' event 'Every Minute'. The 'Payload' section shows a JSON template with random values for 'randomNumber', 'temp', and 'hum'. The 'Send' button is visible. At the bottom of the overlay, there is an 'Upload a CSV file' button and 'Cancel' and 'Save' buttons.

The bottom of the screen shows a Windows taskbar with the date '11/10/2022' and time '11:52 AM'.

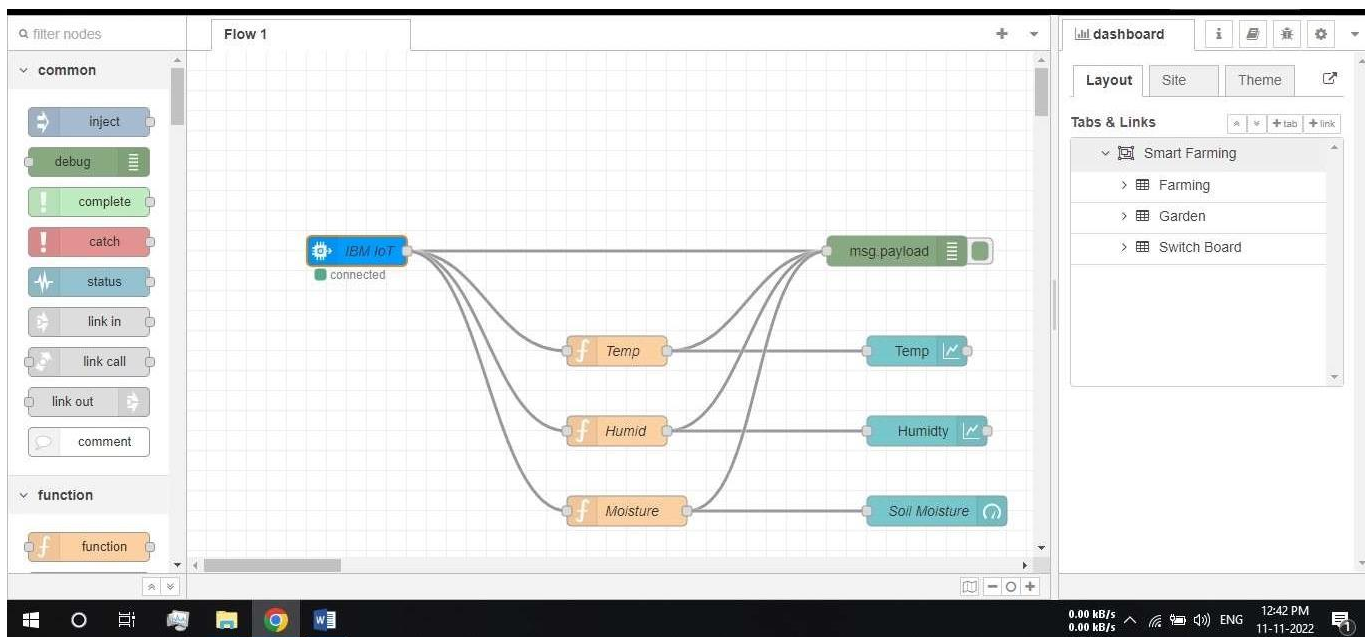
Configuration of Node-Red to collect IBM cloud data



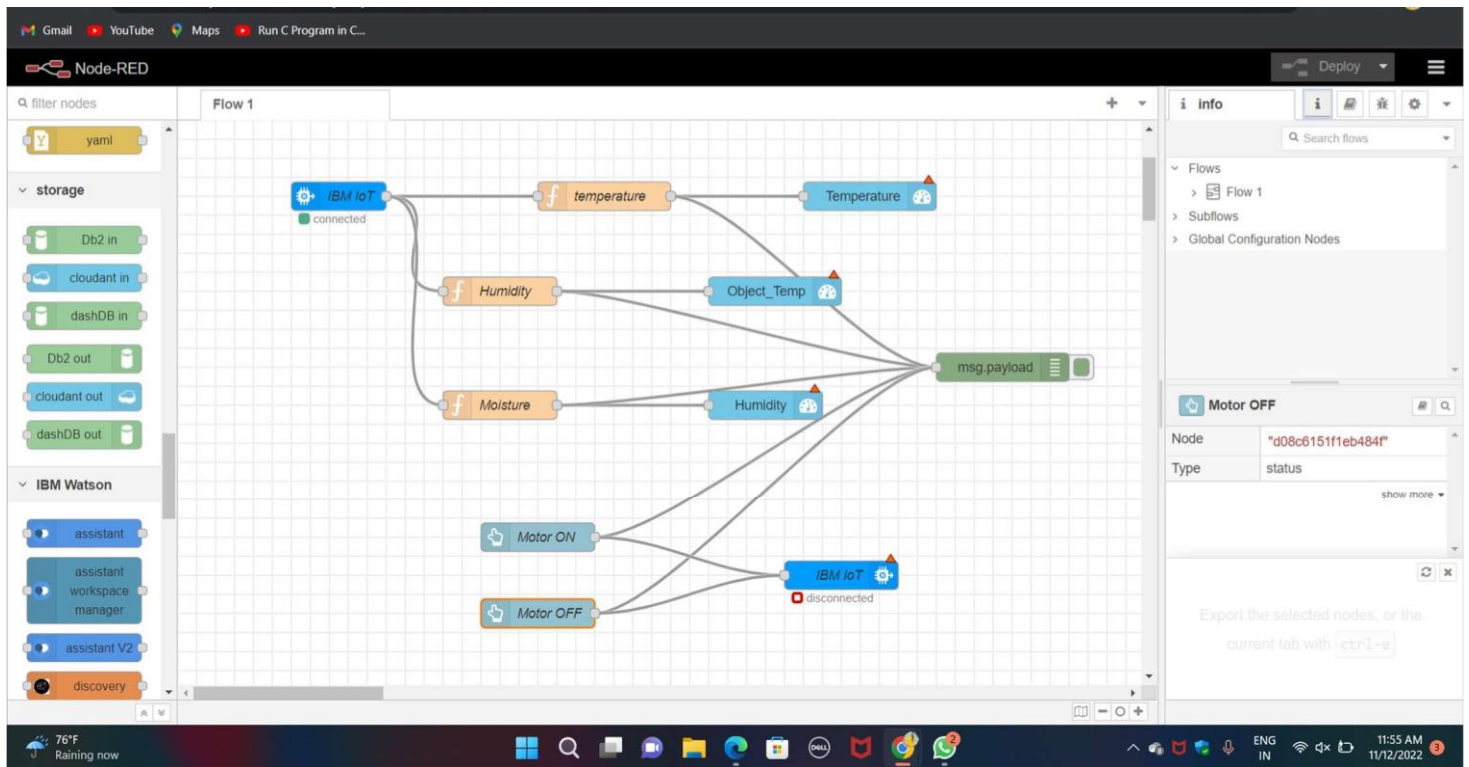
- ☐ The node IBM IoT App In is added to Node-Red workflow. Then the appropriate device credentials obtained earlier are entered into the node to connect and fetch device telemetry to Node-Red.
- ☐ Once it is connected Node-Red receives data from the device
Display the data using debug node for verification
- ☐ 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

```
C:\WINDOWS\system32\cmd.exe
Published Temperature = 109 C Humidity = 64 % to IBM Watson
Published Temperature = 105 C Humidity = 86 % to IBM Watson
Published Temperature = 105 C Humidity = 83 % to IBM Watson
Published Temperature = 102 C Humidity = 86 % to IBM Watson
Published Temperature = 103 C Humidity = 60 % to IBM Watson
Published Temperature = 106 C Humidity = 83 % to IBM Watson
Published Temperature = 101 C Humidity = 85 % to IBM Watson
Published Temperature = 106 C Humidity = 84 % to IBM Watson
Published Temperature = 95 C Humidity = 74 % to IBM Watson
Published Temperature = 107 C Humidity = 73 % to IBM Watson
Published Temperature = 92 C Humidity = 96 % to IBM Watson
Published Temperature = 93 C Humidity = 82 % to IBM Watson
Published Temperature = 98 C Humidity = 80 % to IBM Watson
Published Temperature = 107 C Humidity = 71 % to IBM Watson
Published Temperature = 94 C Humidity = 87 % to IBM Watson
Published Temperature = 106 C Humidity = 76 % to IBM Watson
Published Temperature = 98 C Humidity = 81 % to IBM Watson
Published Temperature = 103 C Humidity = 95 % to IBM Watson
Published Temperature = 92 C Humidity = 66 % to IBM Watson
Published Temperature = 99 C Humidity = 76 % to IBM Watson
Published Temperature = 93 C Humidity = 68 % to IBM Watson
```

□ 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,"humidity":35,"sea_level":1002,"grnd_level":1000},"wind":{"speed":6.23,"deg":170},"clouds":{"all":68},"dt":1589991979,"sys":{"country":"IN","sunrise":1589933553,"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

