

IOT ENABLED SMART FARMING

APPLICATION

SPRINT – 2

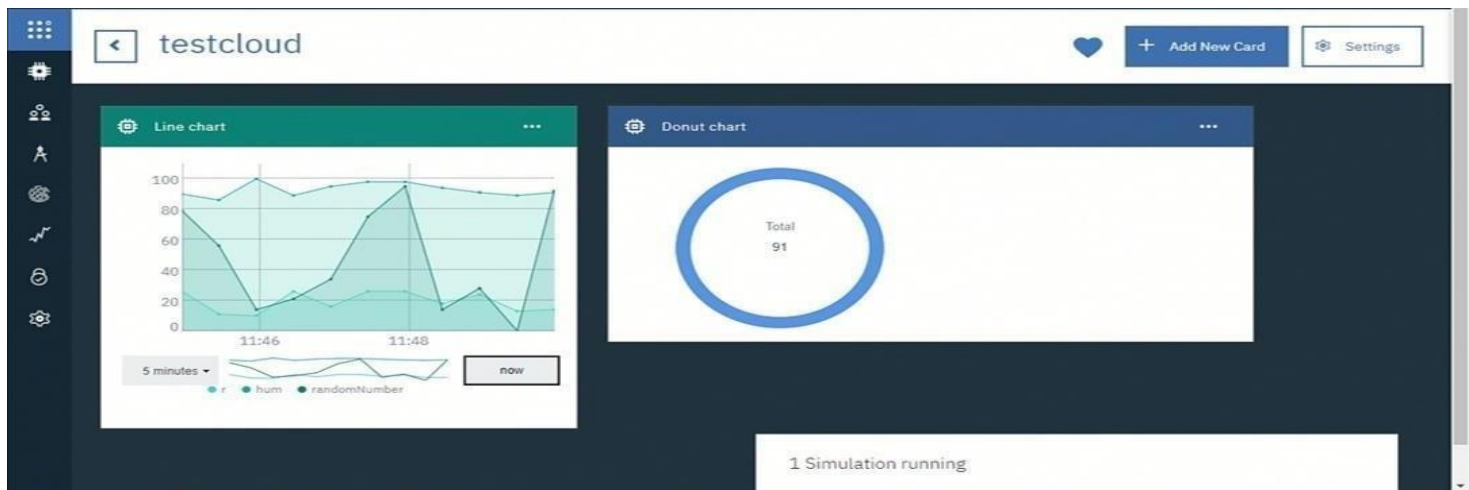
Date	12 NOVEMBER 2022
Team ID	PNT2022TMID52223
Project Name	Project – Smart Farmer-IoT Enabled smart Farming Application

5, Building Project

5.1 Connecting IoT Simulator to IBM Watson IoT Platform

Open link provided in above section 4.3

Give the credentials of your device in IBM Watson IoT Platform



Click on connect

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 shows the IBM IoT Dashboard interface. At the top, there are tabs for 'Browse', 'Action', 'Device Types', and 'Interfaces'. A blue 'Add Device +' button is in the top right. Below these is a modal window with tabs for 'Identity', 'Device Information', 'Recent Events', 'State', and 'Logs'. The 'Recent Events' tab is active, displaying a message: 'The recent events listed show the live stream of data that is coming and going from this device.' Below this is a table with the following data:

Event	Value	Format	Last Received
IoTSensor	{"temp":108,"Humid":64}	json	a few seconds ago
IoTSensor	{"temp":91,"Humid":93}	json	a few seconds ago
IoTSensor	{"temp":108,"Humid":83}	json	a few seconds ago

At the bottom of the modal, it says 'Items per page 50' and '1-2 of 2 items'. The bottom right of the modal shows '1 of 1 page' with navigation arrows.

5.2 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.

The screenshot shows the Node-RED web interface. On the left, the 'common' node palette includes 'inject', 'debug', 'complete', 'catch', 'status', 'link in', 'link call', 'link out', and 'comment'. The 'function' palette includes 'function'. In the center workspace, a flow named 'Flow 1' is shown with an 'IBM IoT' node connected to a 'Humidity' node, which is then connected to a 'Motor Switch On' node. The 'IBM IoT' node is marked as 'connected'. On the right, the 'Edit ibmiot in node' configuration panel is open. It contains the following settings:

- Authentication: API Key
- API Key: IBMIOT APIKEY
- Input Type: Device Event
- Device Type: All or abcd
- Device Id: All or 7654321
- Event: All or +
- Format: All or json
- QoS: 0
- Name: IBM IoT

At the bottom of the configuration panel, there is an 'Enabled' checkbox. To the right of the configuration panel is the 'debug' console, which shows a log of messages received from the IBM IoT node, including temperature and humidity data.

Once it is connected Node-Red receives data from the device

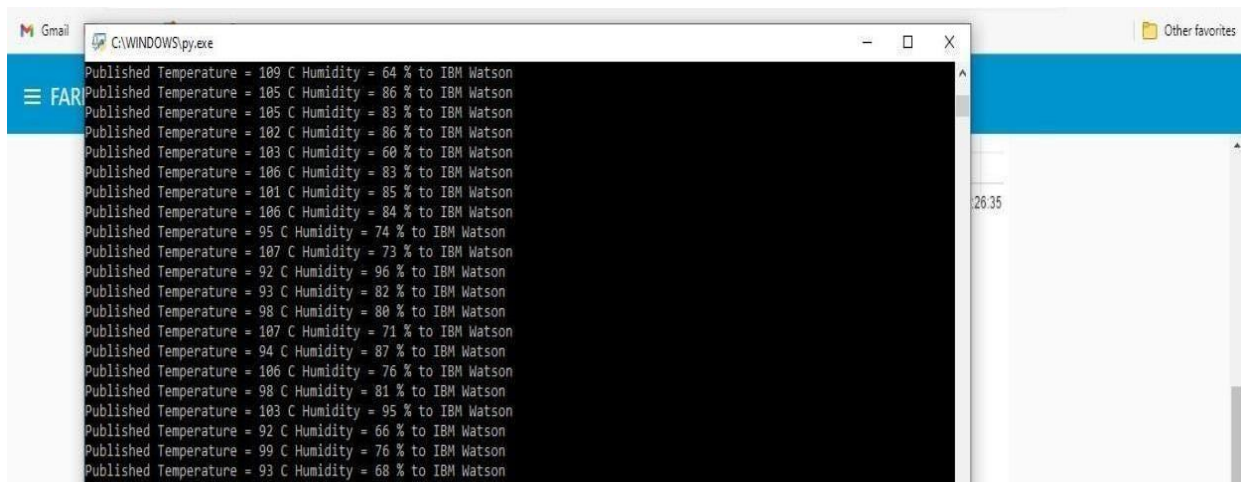
Display the data using debug node for verification

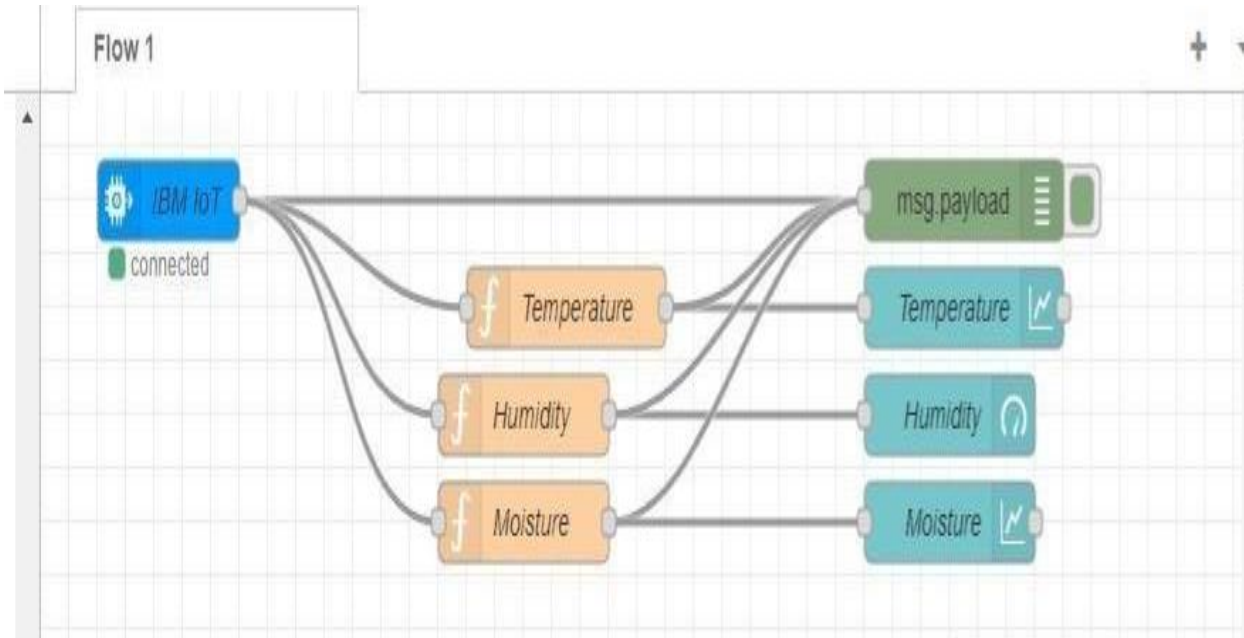
Connect function node and write the Java script code to get each reading separately.

The Java script code for the function node is:

```
msg.payload=msg.payload.d.temperature returnmsg;
```

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 screenshot shows the Node-RED web interface. On the left, the 'common' and 'function' node palettes are visible. The main workspace shows a flow with an 'IBM IoT' node connected to 'Temperature' and 'Humidity' function nodes. The 'Edit chart node' panel is open for the 'Temperature' node. In this panel, the 'I Label' is set to 'Temperature', the 'Type' is 'Line chart', and the 'X-axis' is 'last 5 minute'. A color picker is open over the 'Series Colours' section, showing a color selection interface with RGB values (221, 192, 3). On the right, a 'dashboard' sidebar is visible, showing a tree view of 'FARMING MEASURE DATA' with sub-items for 'Temperature', 'Humidity', and 'Switchboard'.

This is the Java script code I written for the function node to get Temperature separately.

5.3 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

Node-RED interface showing a flow with an IBM IoT node and a function node. The function node is being edited, showing the following code:

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

The function node is named "Temperature". The debug console shows the following output:

```
{ temp: 107, Humid: 73 }
6/11/2022, 12:23:56 pm node: e82c8ed2538304c4
iot-2/type/abcd/id/7654321/ev/IoTSensor/fm/json :
msg.payload : number
107
6/11/2022, 12:23:57 pm node: e82c8ed2538304c4
iot-2/type/abcd/id/7654321/ev/IoTSensor/fm/json :
msg.payload : number
73
6/11/2022, 12:24:06 pm node: e82c8ed2538304c4
iot-2/type/abcd/id/7654321/ev/IoTSensor/fm/json :
msg.payload : Object
{ temp: 92, Humid: 96 }
6/11/2022, 12:24:06 pm node: e82c8ed2538304c4
iot-2/type/abcd/id/7654321/ev/IoTSensor/fm/json :
msg.payload : number
92
6/11/2022, 12:24:07 pm node: e82c8ed2538304c4
iot-2/type/abcd/id/7654321/ev/IoTSensor/fm/json :
msg.payload : number
96
```