SPRINT - 2

Date	14 November 2022
Team ID	PNT2022TMID04642
Project Name	Smart Farmer-IoT Enabled Smart Farming Application

Build Project

1) Connecting IOT Simulator to IBM Watson IOTPlatform

Give the credentials of your device in IBM Watson IOTPlatform

Click on connect

My credentials given to simulator are:

OrgID: 3wc7ia

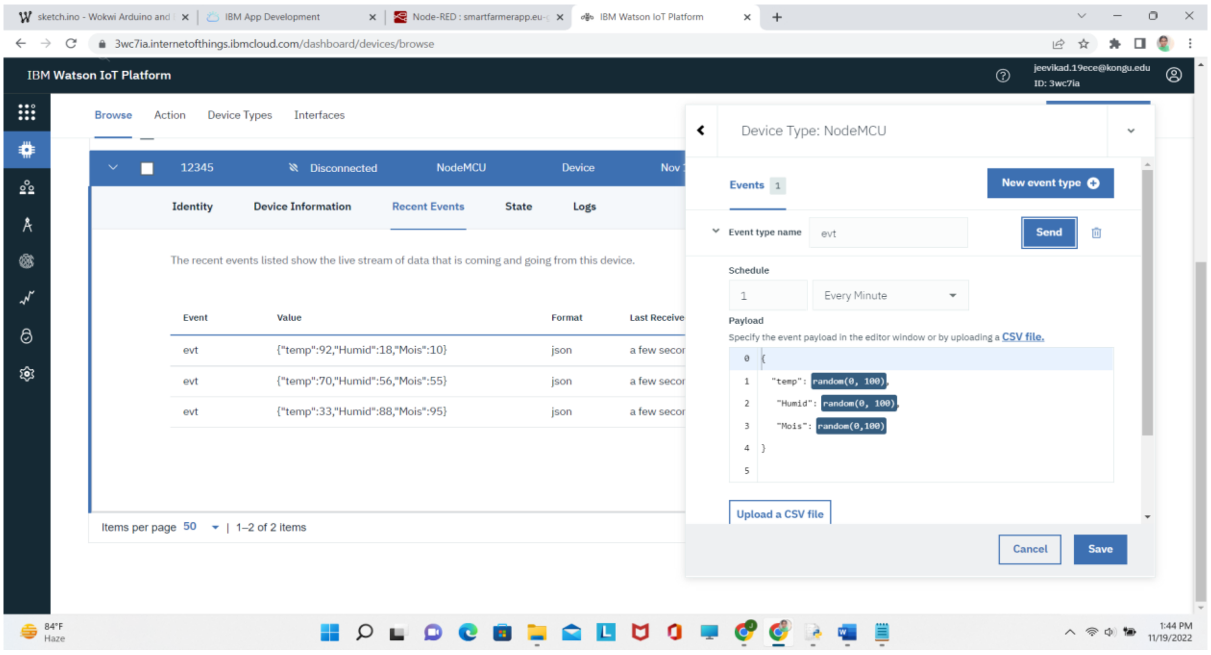
api: a-3wc7ia-gqztwdvblr

Device type:NodeMCU

Token: 8cdIkQot_LAzMFmzAE

Device ID : 12345

Device Token : 12345678



3wc7ia.internetofthings.ibmcloud.com/dashboard/devices/browse

IBM Watson IoT Platform

jeevikad.19ece@kongu.edu
ID: 3wc7ia

Browse | Action | Device Types | Interfaces

12345 | Disconnected | NodeMCU | Device | Nov

Identity | Device Information | Recent Events | State | Logs

The recent events listed show the live stream of data that is coming and going from this device.

Event	Value	Format	Last Receive
evt	{"temp":92,"Humid":18,"Mois":10}	json	a few secor
evt	{"temp":70,"Humid":56,"Mois":55}	json	a few secor
evt	{"temp":33,"Humid":88,"Mois":95}	json	a few secor

Items per page 50 | 1–2 of 2 items

Device Type: NodeMCU

Events 1 | New event type

Event type name: evt | Send

Schedule: 1 | Every Minute

Payload

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

```
{
  "temp": random(0, 100),
   "Humid": random(0, 100),
   "Mois": random(0,100)
}
```

Upload a CSV file

Cancel | Save

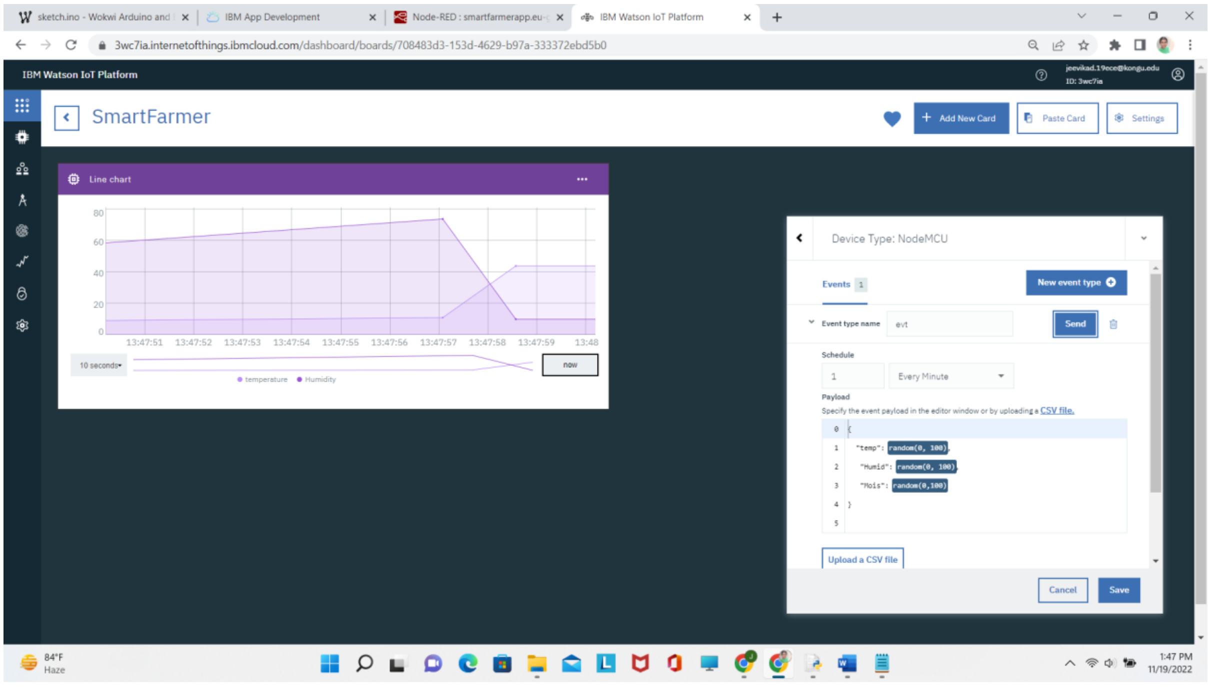
You can see the received data in graphs by creating cards in Boards tab

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": "NodeMCU", ▪ "temperature": 17, ▪ "humidity": 76, ▪ "Moisture ": 25 } }



Device Type: NodeMCU

Events 1

Event type name: evt

Schedule: 1 Every Minute

Payload

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

```
{
  "temp": random(0, 100),
   "Humid": random(0, 100),
   "Mois": random(0,100)
}
```

2) Configure Node-Red to IBM Cloud

The node IBM IOT App In is added to Node-Red workflow.

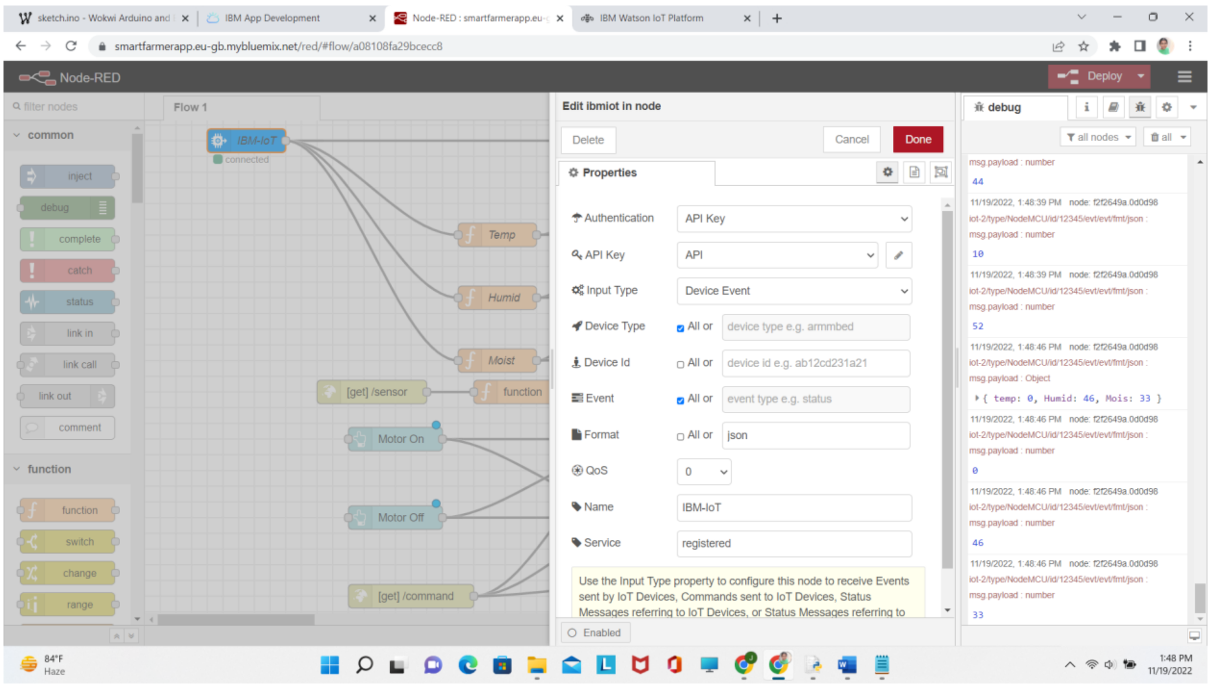
Then the appropriate device credentials obtained earlier are entered into the node to connect and fetch device telemetry to Node-Red.

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



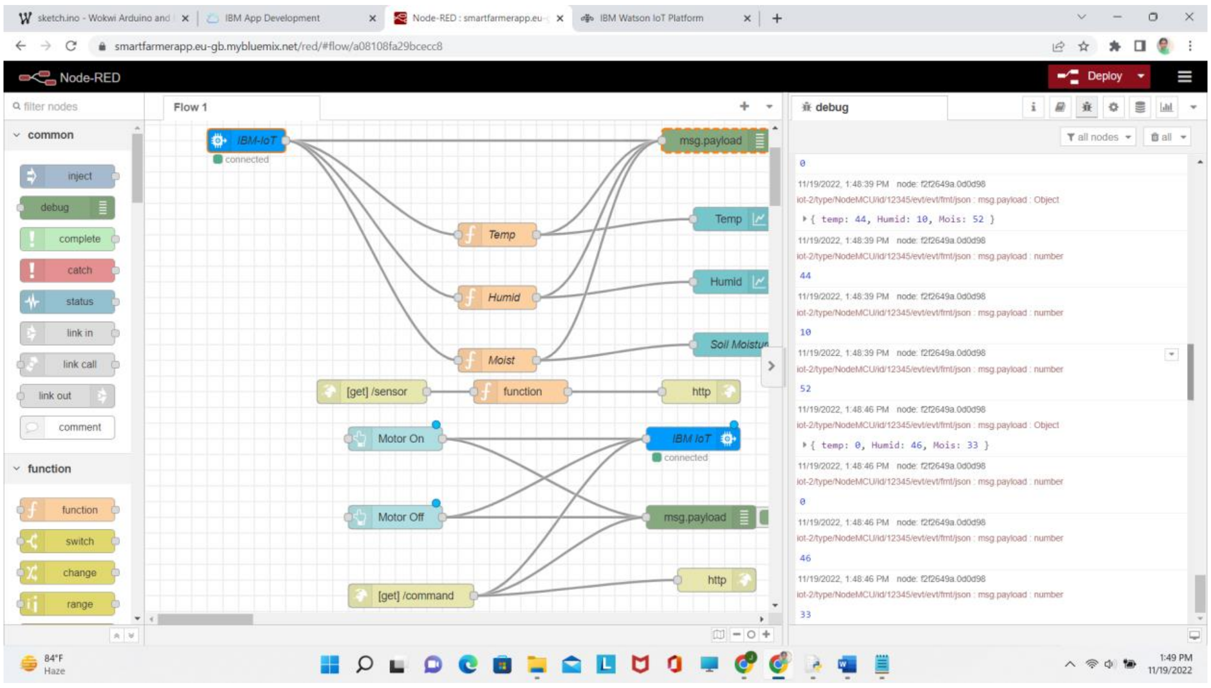
Edit ibmiot in node

Authentication	API Key
API Key	API
Input Type	Device Event
Device Type	☑ All or
Device Id	☐ All or
Event	☑ All or
Format	☐ All or json
QoS	0
Name	IBM-IoT
Service	registered

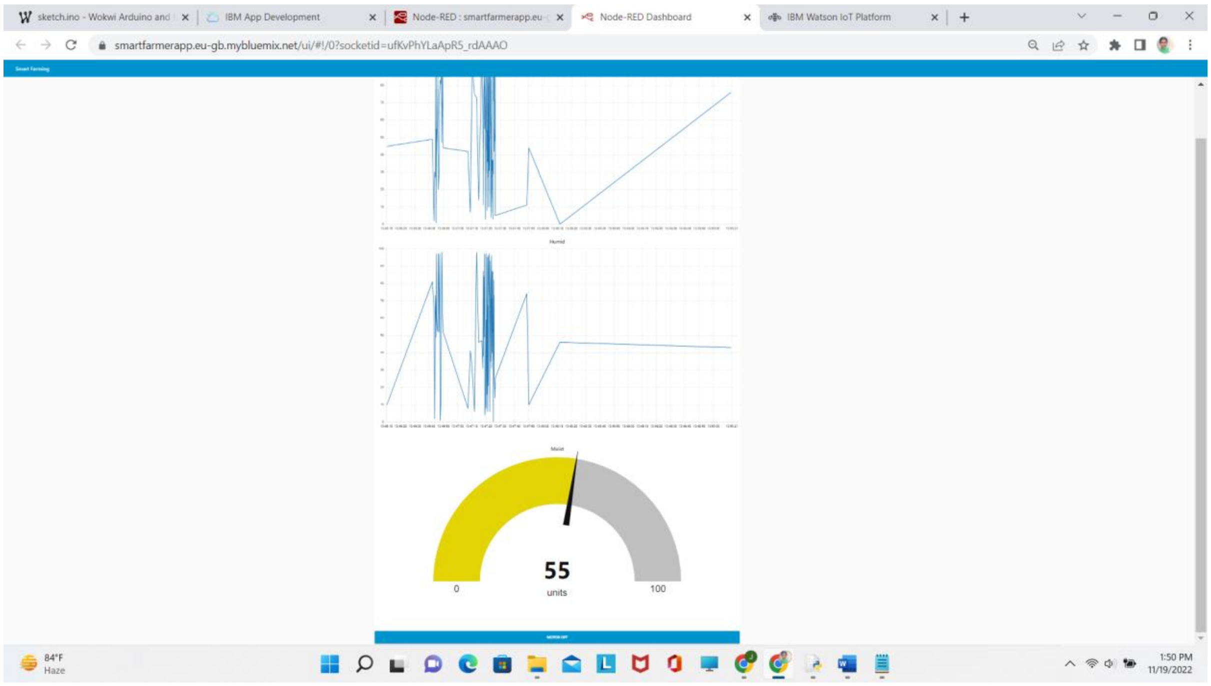
Use the Input Type property to configure this node to receive Events sent by IoT Devices, Commands sent to IoT Devices, Status Messages referring to IoT Devices, or Status Messages referring to

```
msg.payload : number
44
11/19/2022, 1:48:39 PM  node: f2f2649a.0d0d98
iot-2/type/NodeMCU/id/12345/evt/evt/fmt/json :
msg.payload : number
10
11/19/2022, 1:48:39 PM  node: f2f2649a.0d0d98
iot-2/type/NodeMCU/id/12345/evt/evt/fmt/json :
msg.payload : number
52
11/19/2022, 1:48:46 PM  node: f2f2649a.0d0d98
iot-2/type/NodeMCU/id/12345/evt/evt/fmt/json :
msg.payload : Object
▸{ temp: 0, Humid: 46, Mois: 33 }
11/19/2022, 1:48:46 PM  node: f2f2649a.0d0d98
iot-2/type/NodeMCU/id/12345/evt/evt/fmt/json :
msg.payload : number
0
11/19/2022, 1:48:46 PM  node: f2f2649a.0d0d98
iot-2/type/NodeMCU/id/12345/evt/evt/fmt/json :
msg.payload : number
46
11/19/2022, 1:48:46 PM  node: f2f2649a.0d0d98
iot-2/type/NodeMCU/id/12345/evt/evt/fmt/json :
msg.payload : number
33
```

Data received from the cloud in Node-R



```
0
11/19/2022, 1:48:39 PM  node: f2f2649a.0d0d98
iot-2/type/NodeMCU/id/12345/evt/evt/fmt/json : msg.payload : Object
▸{ temp: 44, Humid: 10, Mois: 52 }
11/19/2022, 1:48:39 PM  node: f2f2649a.0d0d98
iot-2/type/NodeMCU/id/12345/evt/evt/fmt/json : msg.payload : number
44
11/19/2022, 1:48:39 PM  node: f2f2649a.0d0d98
iot-2/type/NodeMCU/id/12345/evt/evt/fmt/json : msg.payload : number
10
11/19/2022, 1:48:39 PM  node: f2f2649a.0d0d98
iot-2/type/NodeMCU/id/12345/evt/evt/fmt/json : msg.payload : number
52
11/19/2022, 1:48:46 PM  node: f2f2649a.0d0d98
iot-2/type/NodeMCU/id/12345/evt/evt/fmt/json : msg.payload : Object
▸{ temp: 0, Humid: 46, Mois: 33 }
11/19/2022, 1:48:46 PM  node: f2f2649a.0d0d98
iot-2/type/NodeMCU/id/12345/evt/evt/fmt/json : msg.payload : number
0
11/19/2022, 1:48:46 PM  node: f2f2649a.0d0d98
iot-2/type/NodeMCU/id/12345/evt/evt/fmt/json : msg.payload : number
46
11/19/2022, 1:48:46 PM  node: f2f2649a.0d0d98
iot-2/type/NodeMCU/id/12345/evt/evt/fmt/json : msg.payload : number
33
```



This is the Java script code I written for the function node to get Temperature separately. Configuration of Node-Red to collect data from Open Weather

The Node-Red also receive data from the Open Weather API by HTTP GET request.

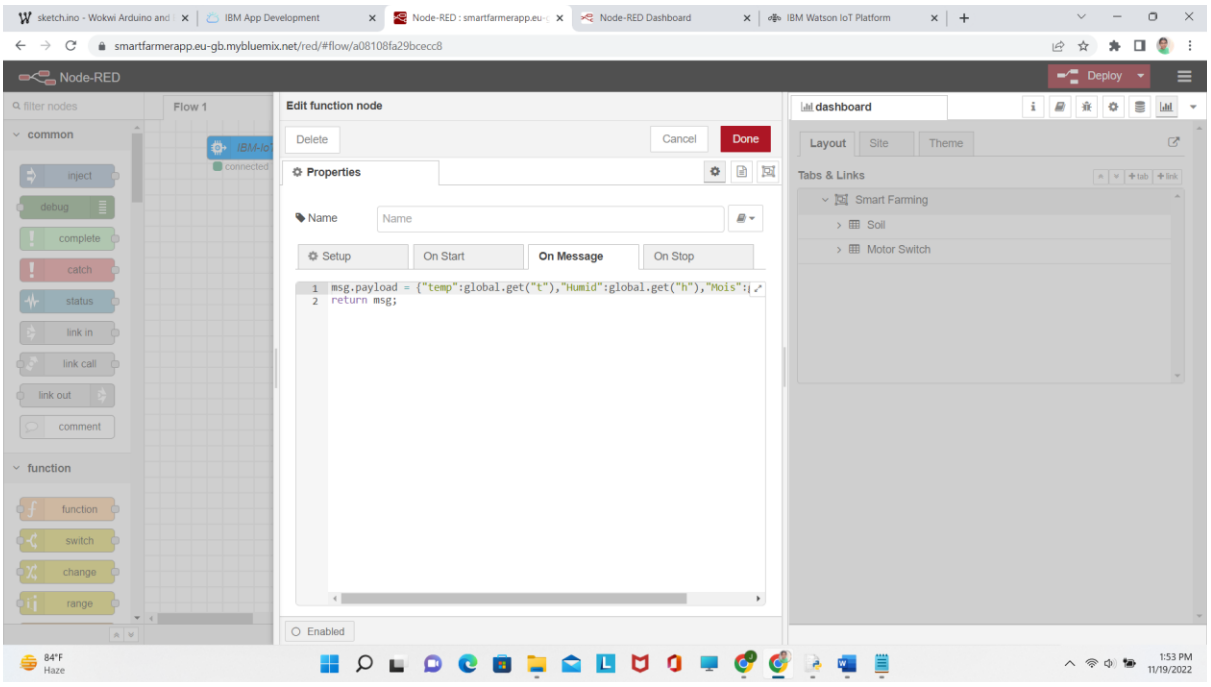
An inject trigger is added to perform HTTP request for every certain interval.

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



```
msg.payload = {"temp":global.get("t"),"Humid":global.get("h"),"Mois":
return msg;
```