SPRINTDELIVERY–2

TITLE	**Smart Farmer-IOT Enabled Smart Farming Application**
DOMAIN NAME	INTERNET OF THINGS
TEAM ID	PNT2022TMID30465
LEADER NAME	SHRIMATHI.P
TEAM MEMBER NAME	KASTHURI.P SURUTHI.R VASUKI.P

\5,Building Project

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

PlatformClick on connect

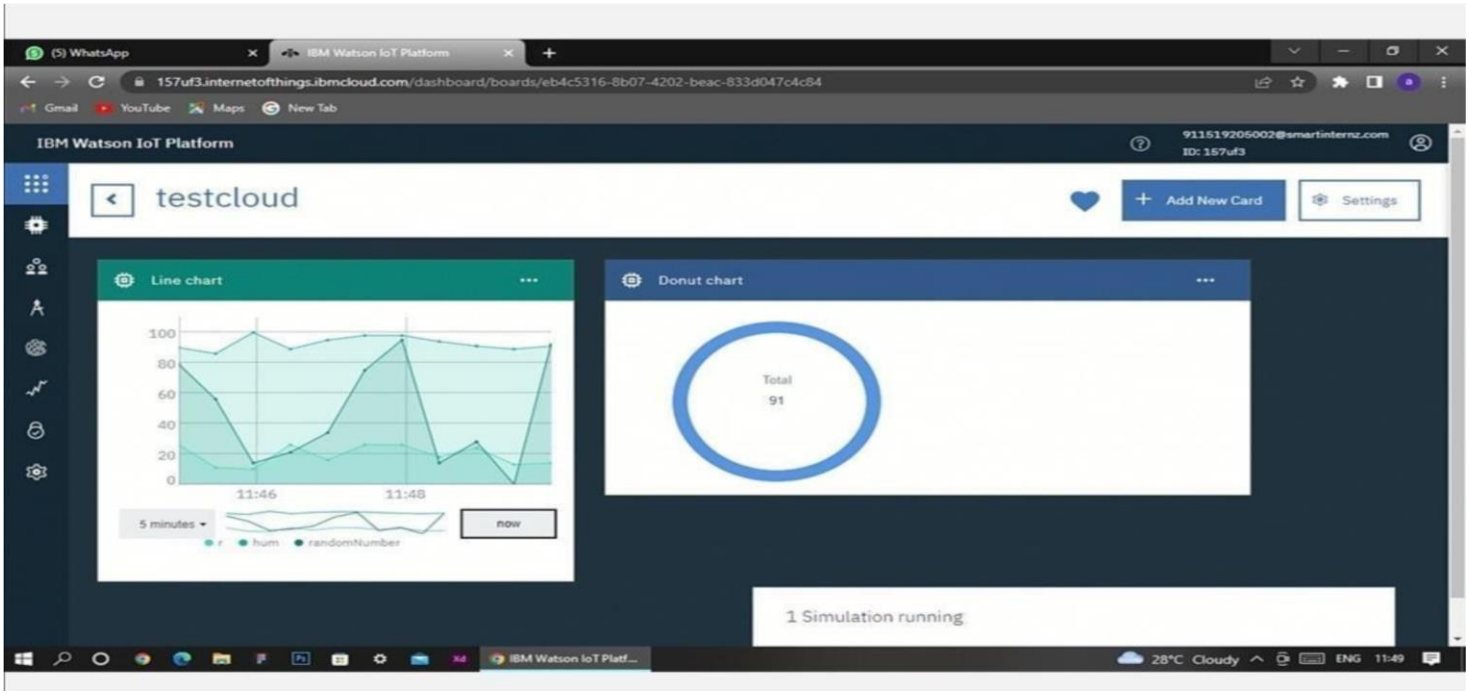
My credentials given to simulator are:

OrgID: **157uf3** api: **a-157uf3f5rg4qxpd3**

Device type: **abcd**

token:**6ogMaaQHNWFEgOD8R?**

Device ID : **7654321**
DeviceToken: **87654321**



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)

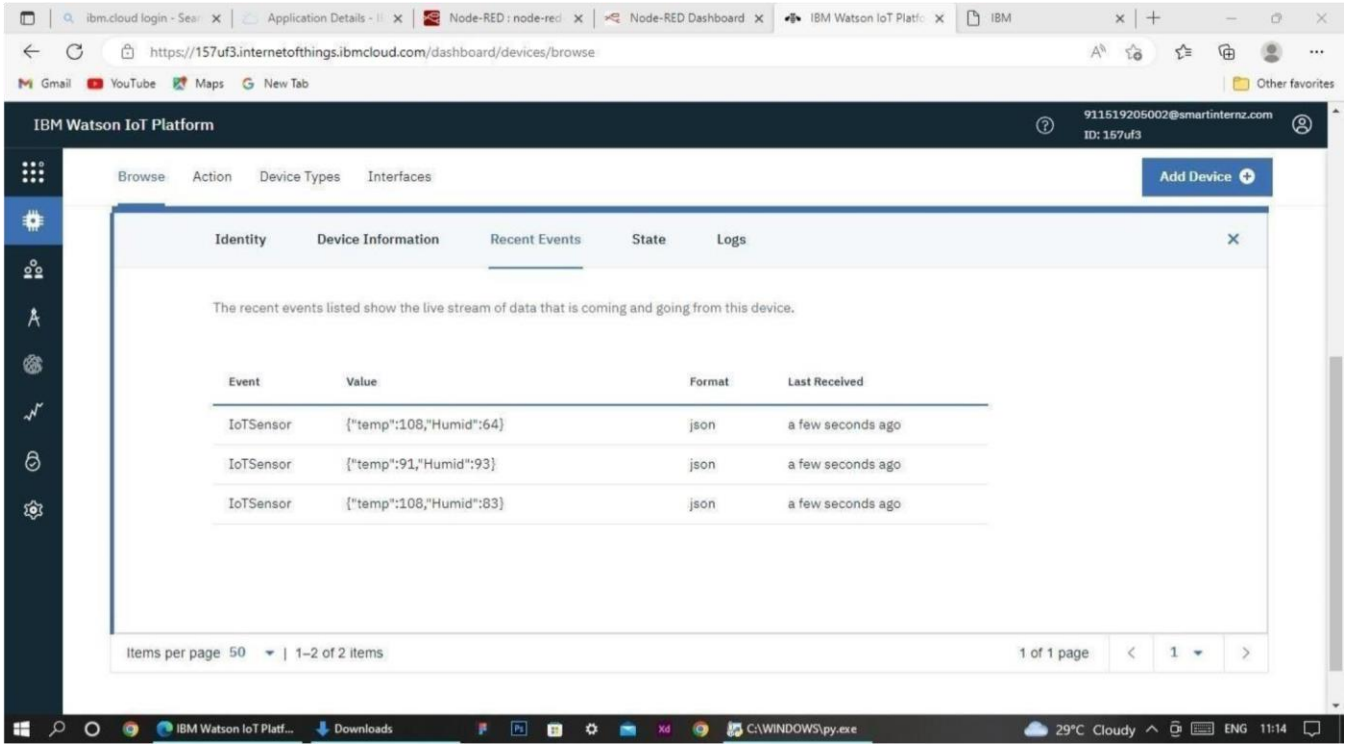
{

"d": {

- "name": "abcd",
- "temperature": 17,
- "humidity": 76,
- "Moisture ": 25

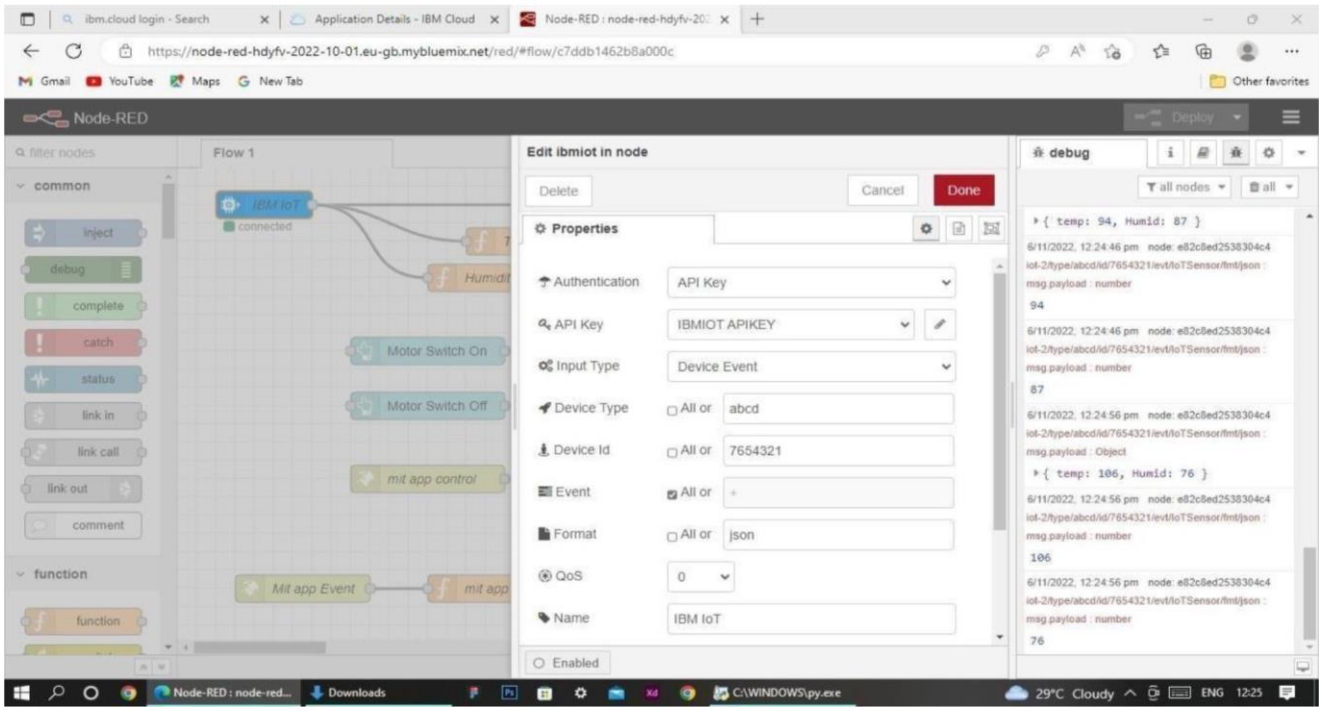
}

}



Configuration of Node-Red to collect IBM cloud data

The node IBM IoT App In is added to Node-Red workflow. Then the appropriatedevice 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

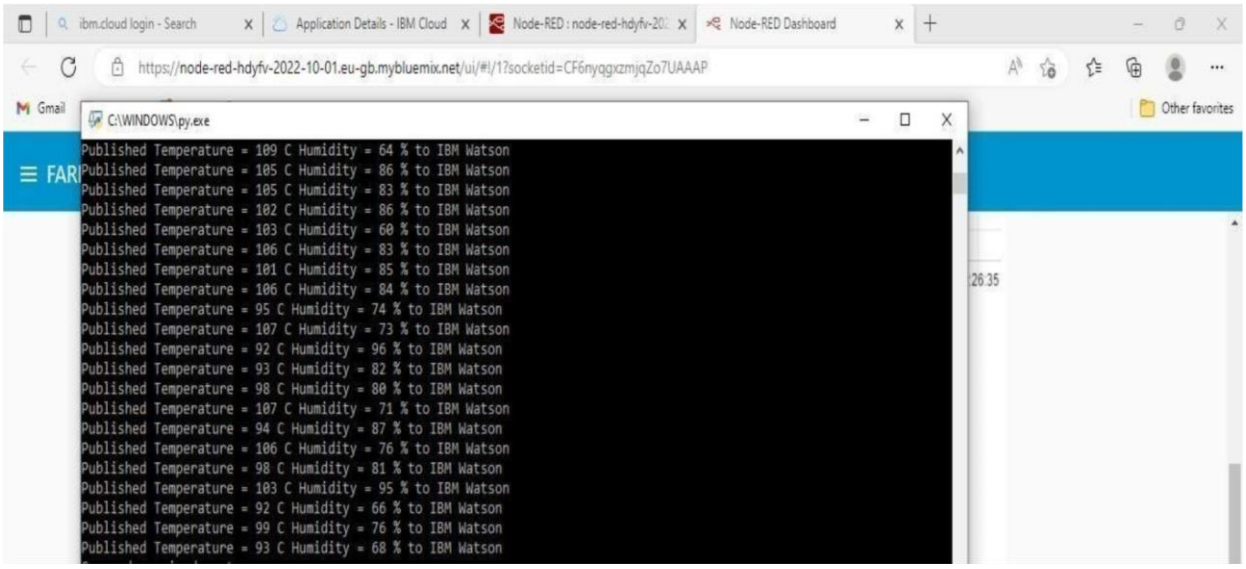
Display the data using debug node for verification

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

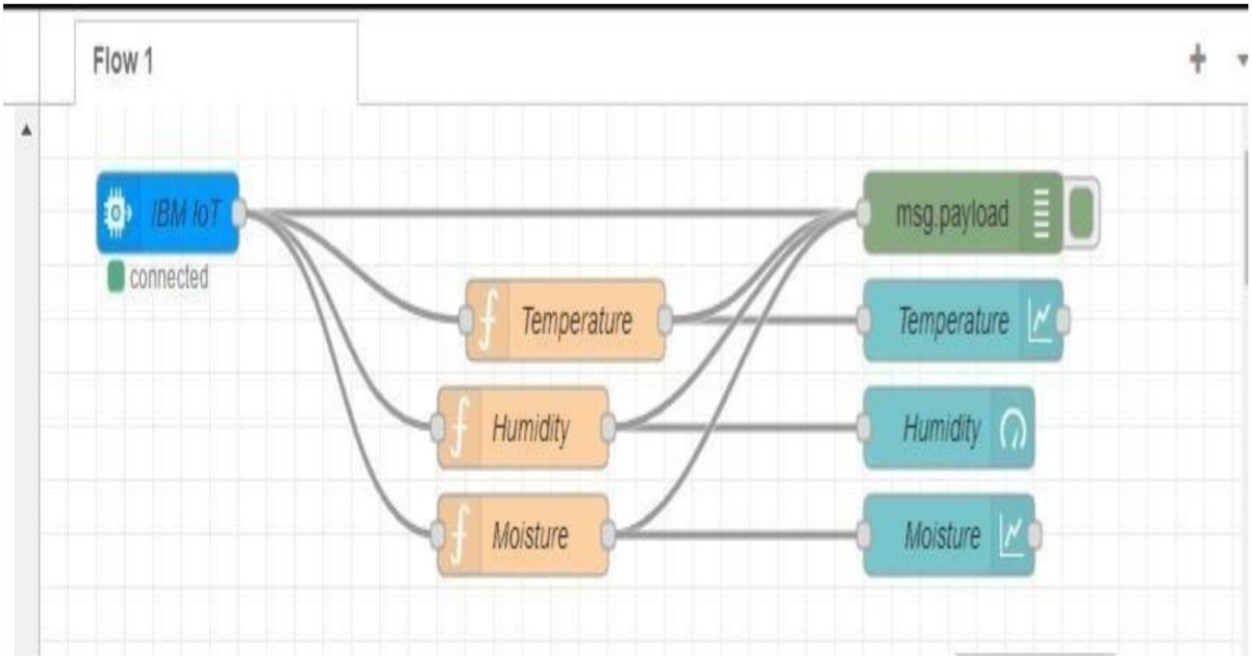
The Java script code for the function node is:

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

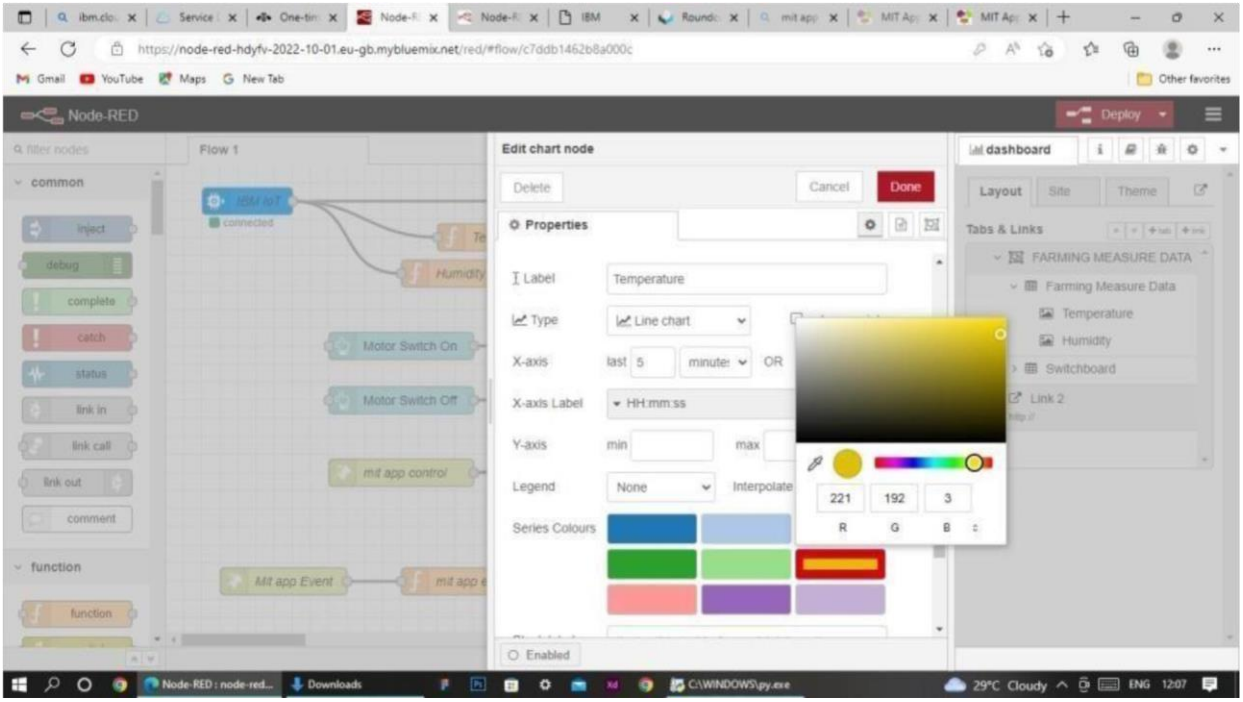
Finally connect Gauge nodes from dashboard to see the data in UI



Data received from the cloud in Node-Red console



Nodes connected in following manner to get each reading separately



This 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":30 7 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":17 0
}
,"clouds":{"all":68},"dt":1589991979,"sys":{"country":"IN","sunrise":1589933553, "sunset":1589979720},"timezone":19800,"id":1270791,"name":"Gūdūr","cod":20 0}

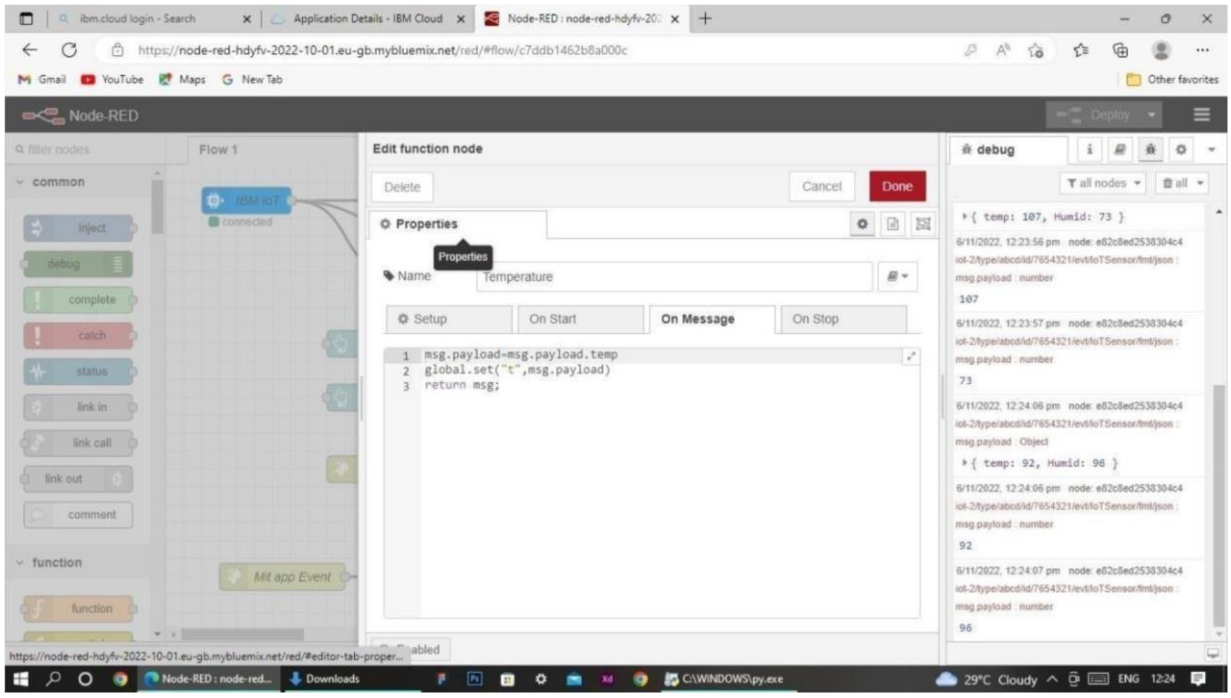
In order to parse the JSON string we use Java script functions and get each parameters var temperature = msg.payload.main.temp;temperature

= temperature-273.15;

return {payload : temperature.toFixed(2)};

In the above Java script code we take temperature parameter into a new variableand convert it from kelvin to Celsius

Then we add Gauge and text nodes to represent data visually in UI


Node-RED
Flow 1
Edit function node
Delete
Cancel
Done
Properties
Name
Temperature
Setup
On Start
On Message
On Stop
msg.payload=msg.payload.temp
global.set("t",msg.payload)
return msg;
debug
{ temp: 107, Humid: 73 }
{ temp: 92, Humid: 96 }
common
function
inject
debug
complete
catch
status
link in
link call
link out
comment
function
Mit app Event