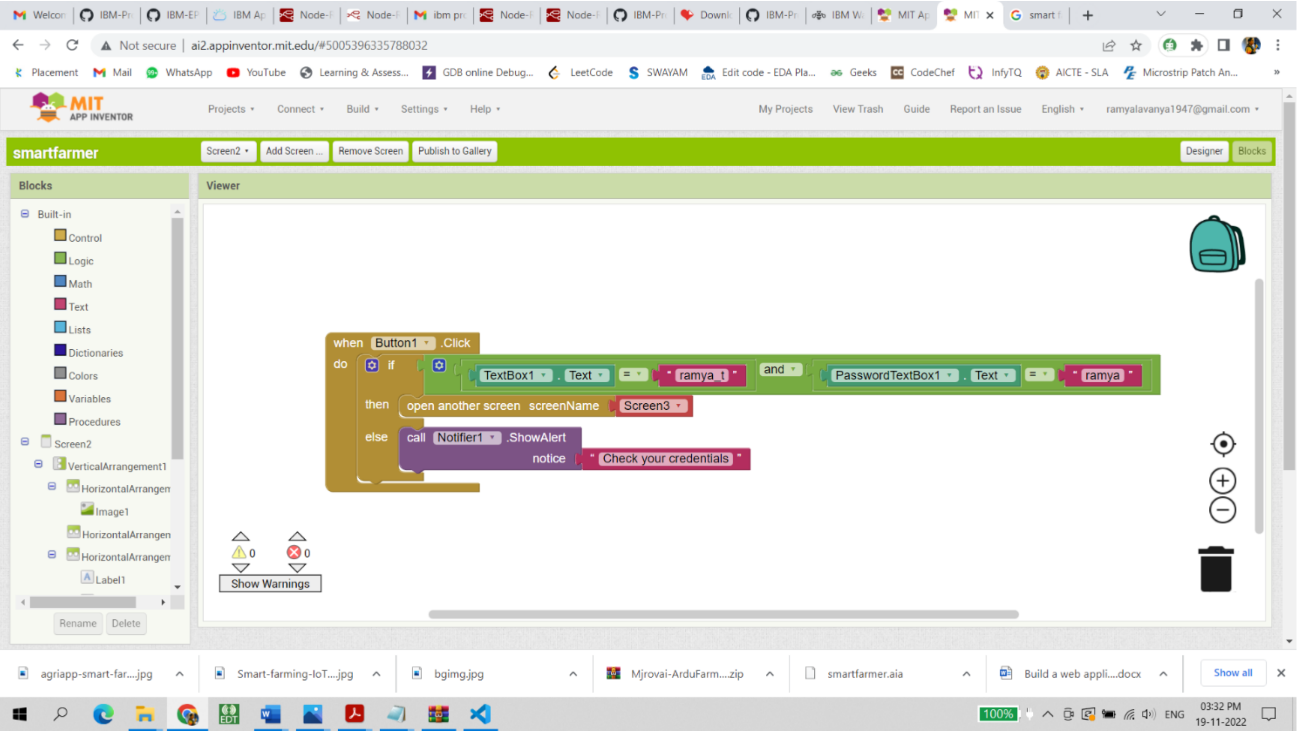
Project Delivery Sprint - 1

Date	20 Oct 2022
Team ID	PNT2022TMID04704
Project Name	Smart Farmer-IOT Enabled Smart Farming Application

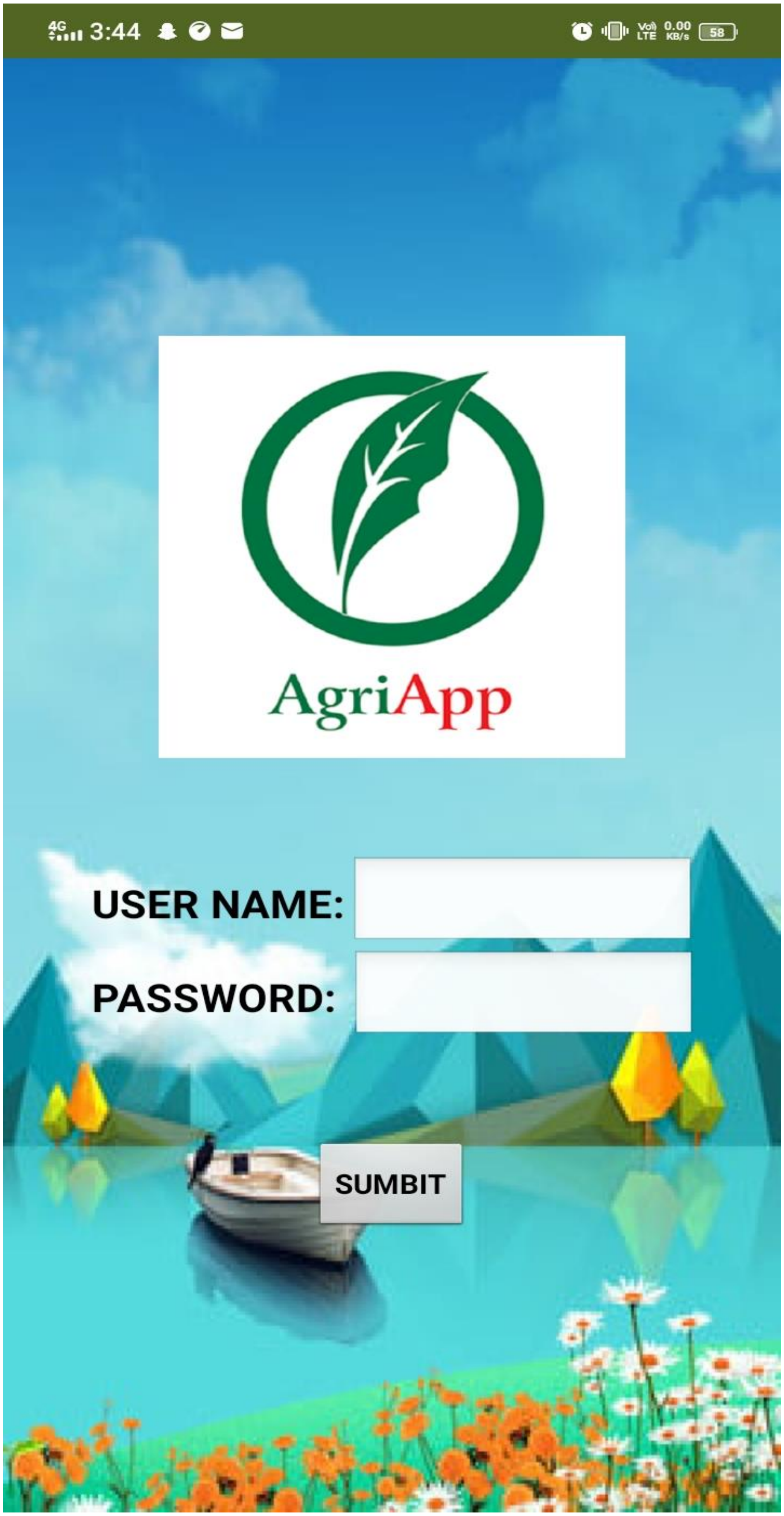
Sprint	Functional Requirement (Epic)	User Story Number	User Story /Task
Sprint-1	Registration (Farmer)	USN-1	As a user, I can registerfor the application by entering my username, password.

Block diagram → Registration (Farmer)



```
when Button1 .Click
do  if  TextBox1 . Text = "ramya_t" and PasswordTextBox1 . Text = "ramya"
    then  open another screen  screenName "Screen3"
    else  call Notifier1 .ShowAlert
                          notice "Check your credentials"
```

Mobile App page


4G 3:44
Vo) LTE 0.00 KB/s 58
AgriApp
USER NAME:
PASSWORD:
SUMBIT

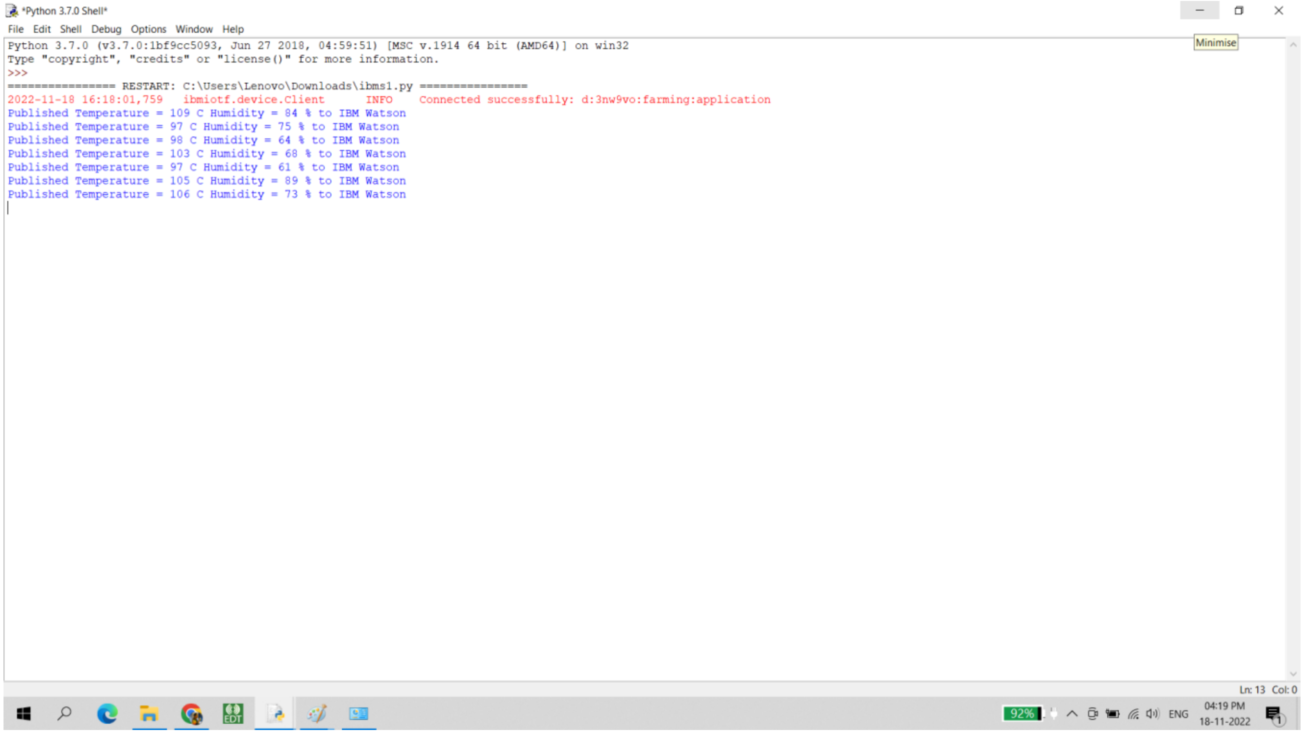
Sprint	Functional Requirement (Epic)	User Story Number	User Story /Task
Sprint-1	IBM IoT cloud Service	USN-2	Publish and subscribe to IBM IoT cloud

Python code Connect With IBM IoT Cloud Service

```
import time
import sys
import ibmiotf.application
import ibmiotf.device
import random

#Provide your IBM Watson Device Credentials
organization = "3nw9vo"
deviceType = "farming"
deviceId = "application"
authMethod = "token"
authToken = "87654321"

# Initialize GPIO
def myCommandCallback(cmd):
    print("Command received: %s" % cmd.data['command'])
    status=cmd.data['command']
    if status=="motoron":
        print ("Motor is on")
    elif status == "motoroff":
        print ("Motor is off")
    else :
        print ("please send proper command")

try:
        deviceOptions = {"org": organization, "type": deviceType, "id": deviceId, "auth-method":
authMethod, "auth-token": authToken}
        deviceCli = ibmiotf.device.Client(deviceOptions)
```

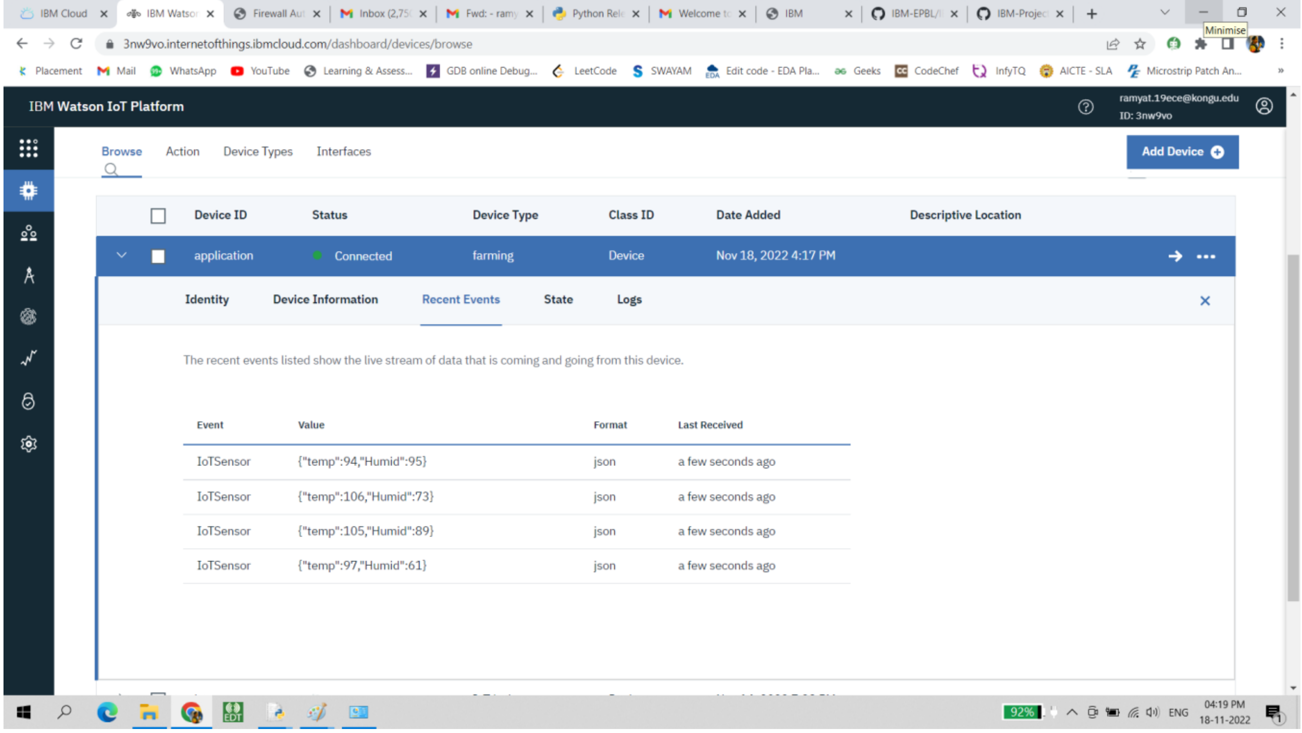
```
         #.............................................

 except Exception as e:
         print("Caught exception connecting device: %s" % str(e))
         sys.exit()

 # Connect and send a datapoint "hello" with value "world" into the cloud as an event of type
 "greeting" 10 times
 deviceCli.connect()

 while True:
      #Get Sensor Data from DHT11

      temp=random.randint(90,110)
      Humid=random.randint(60,100)

      data = { 'temp' : temp, 'Humid': Humid }
      #print data
      def myOnPublishCallback():
         print ("Published Temperature = %s C" % temp, "Humidity = %s %%" % Humid, "to IBM
 Watson")

      success = deviceCli.publishEvent("IoTSensor", "json", data, qos=0,
 on_publish=myOnPublishCallback)
      if not success:
         print("Not connected to IoTF")
      time.sleep(10)

      deviceCli.commandCallback = myCommandCallback

 # Disconnect the device and application from the cloud
deviceCli.disconnect()
```

OUTPUT:



```
Python 3.7.0 (v3.7.0:1bf9cc5093, Jun 27 2018, 04:59:51) [MSC v.1914 64 bit (AMD64)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> 
================ RESTART: C:\Users\Lenovo\Downloads\ibms1.py ================
2022-11-18 16:18:01,759   ibmiotf.device.Client      INFO    Connected successfully: d:3nw9vo:farming:application
Published Temperature = 109 C Humidity = 84 % to IBM Watson
Published Temperature = 97 C Humidity = 75 % to IBM Watson
Published Temperature = 98 C Humidity = 64 % to IBM Watson
Published Temperature = 103 C Humidity = 68 % to IBM Watson
Published Temperature = 97 C Humidity = 61 % to IBM Watson
Published Temperature = 105 C Humidity = 89 % to IBM Watson
Published Temperature = 106 C Humidity = 73 % to IBM Watson
```

Data received



Device ID	Status	Device Type	Class ID	Date Added	Descriptive Location
application	Connected	farming	Device	Nov 18, 2022 4:17 PM	

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

Event	Value	Format	Last Received
IoTSensor	{"temp":94,"Humid":95}	json	a few seconds ago
IoTSensor	{"temp":106,"Humid":73}	json	a few seconds ago
IoTSensor	{"temp":105,"Humid":89}	json	a few seconds ago
IoTSensor	{"temp":97,"Humid":61}	json	a few seconds ago

Project Delivery Sprint - 2

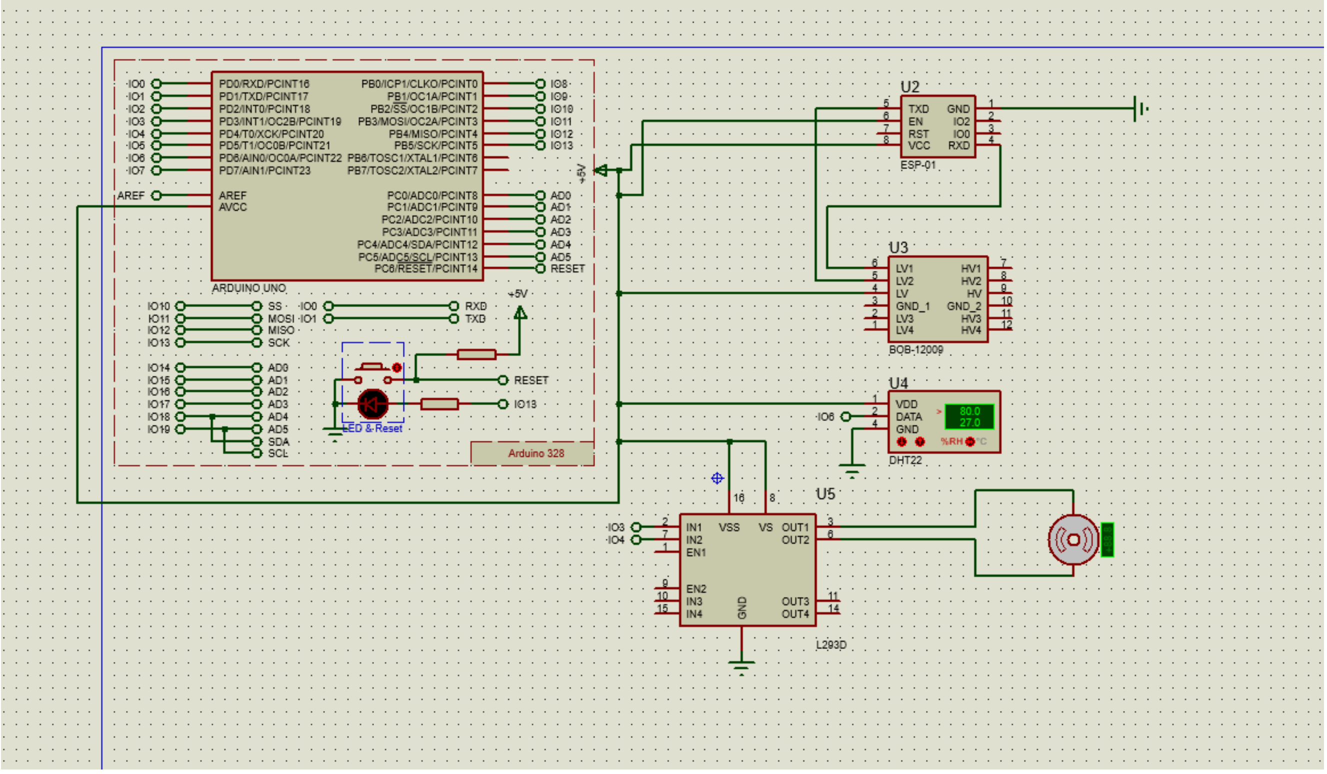
Date	28 Oct 2022
Team ID	PNT2022TMID04704
Project Name	Smart Farmer - IoT Enabled Smart Farming Application

Sprint	Functional Requirement (Epic)	User Story Number	User Story /Task
Sprint-2	I/O interface for Sensors.	USN-3	As a user, I can connect the various sensors like temperature, moisture sensor with Arduino board.

CODE:

```
#include<iWre.h>
#include <DHT.h>;


#define DHTPIN 6
#define m1 3
#define m2 4
#define DHTTYPE DHT22
DHT dht(DHTPIN, DHTTYPE);


Variables
int chk;
float hum;
float temp;

void setup()
{
   pinMode(m1, OUTPUT);
   pinMode(m2, OUTPUT);
  Serial.begin(9600);
  dht.begin();
}

void loop()
{
    delay(2000);
    hum = 80;
    temp= 27;
    Serial.print("Humidity: ");
    Serial.print(hum);
    Serial.print(" %, Temp: ");
    Serial.print(temp);
    Serial.println(" Celsius");
    delay(5000);
    temp=35;

    if (temp>30){
      digitalWrite (m1, HIGH);
      delay(5000);
    }


}
```

Circuit Diagram:



Python code To Connect Sensors

```
import time
import sys
import ibmiotf.application
import ibmiotf.device
import random

#Provide your IBM Watson Device Credentials
organization = "3nw9vo"
deviceType = "farming"
deviceId = "application"
authMethod = "token"
authToken = "87654321"

# Initialize GPIO
def myCommandCallback(cmd):
    print("Command received: %s" % cmd.data['command'])
    status=cmd.data['command']
    if status=="motoron":
        print ("Motor is on")
    elif status == "motoroff":
        print ("Motor is off")
    else :
        print ("please send proper command")

try:
          deviceOptions = {"org": organization, "type": deviceType, "id": deviceId, "auth-
method": authMethod, "auth-token": authToken}
          deviceCli = ibmiotf.device.Client(deviceOptions)
          #..............................................

except Exception as e:
          print("Caught exception connecting device: %s" % str(e))
          sys.exit()

# Connect and send a datapoint "hello" with value "world" into the cloud as an event of
type "greeting" 10 times
deviceCli.connect()

while True:
        #Get Sensor Data from DHT11

        temp=random.randint(90,110)
        Humid=random.randint(60,100)

        data = { 'temp' : temp, 'Humid': Humid }
        #print data
        def myOnPublishCallback():
            print ("Published Temperature = %s C" % temp, "Humidity = %s %%" %
Humid, "to IBM Watson")
```

```
        success = deviceCli.publishEvent("IoTSensor", "json", data, qos=0, on_publish=myOnPublishCallback)
        if not success:
            print("Not connected to IoTF")
        time.sleep(10)

        deviceCli.commandCallback = myCommandCallback

# Disconnect the device and application from the cloud
deviceCli.disconnect()
```

Project Delivery Sprint - 3

Date	17 Nov 2022
Team ID	PNT2022TMID04704
Project Name	Smart Farmer - IoT Enabled Smart Farming Application

Sprint	Functional Requirement(Epic)	User Story Number	User Story /Task
Sprint-3	Interface for connecting to IBM IoT cloud.	USN-4	Temperature and soil moisture sensor sends the data to the cloud via IBM Watson service.

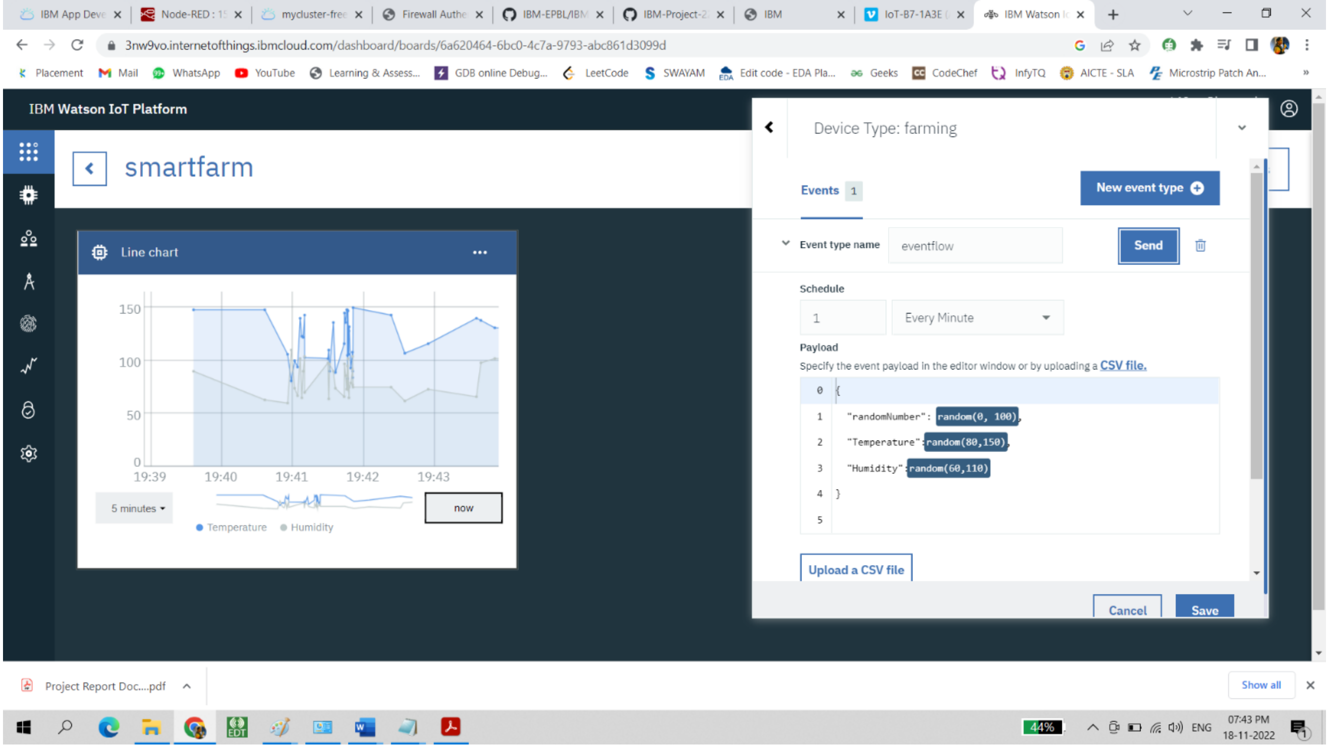
Connecting IOT Simulator to IBM Watson IOT Platform

Give the credentials of your device in IBM

Watson Mycredentials given to simulator are:

```
organization = "3nw9vo"
deviceType = "farming"
deviceId = "application"
authMethod = "token"
authToken = "87654321"
```

- You can see the received data in graphs by creating cards in Boards tab
- You will receive the simulator data in cloud



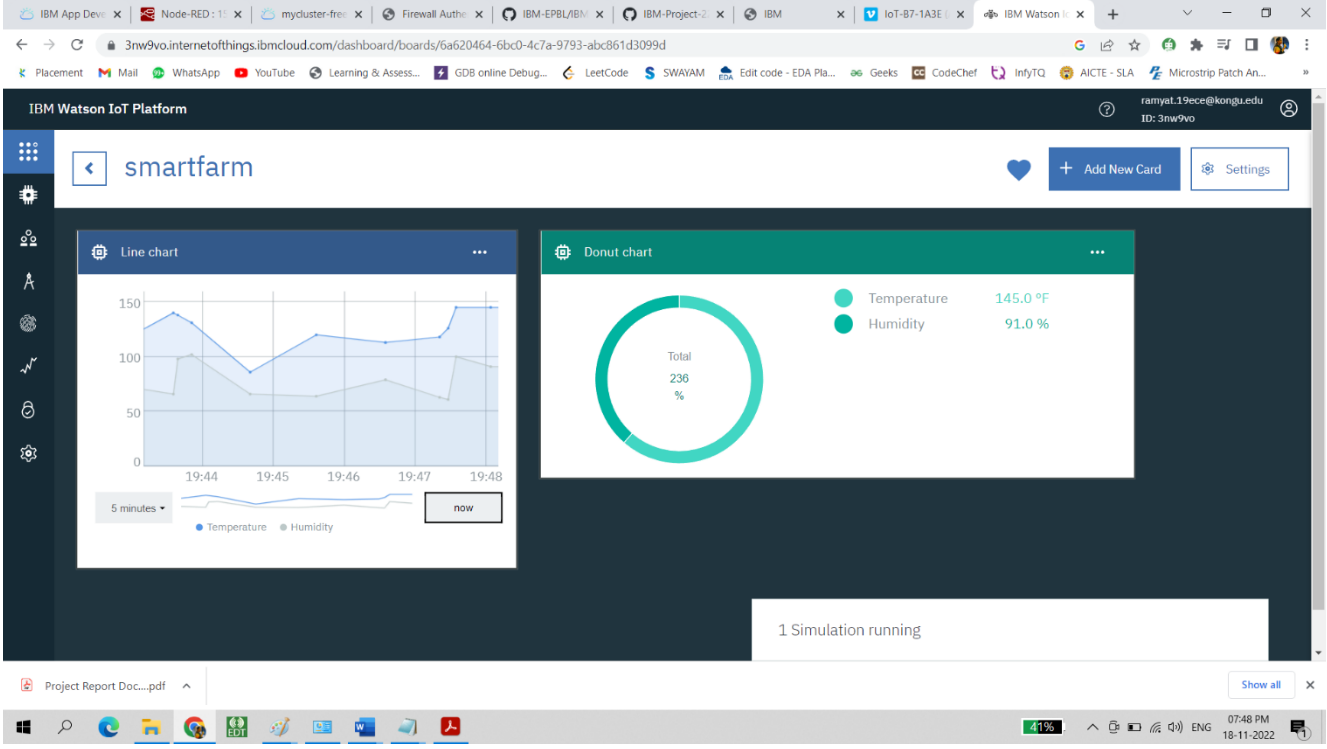
Event type name: eventflow

Schedule: 1 Every Minute

Payload

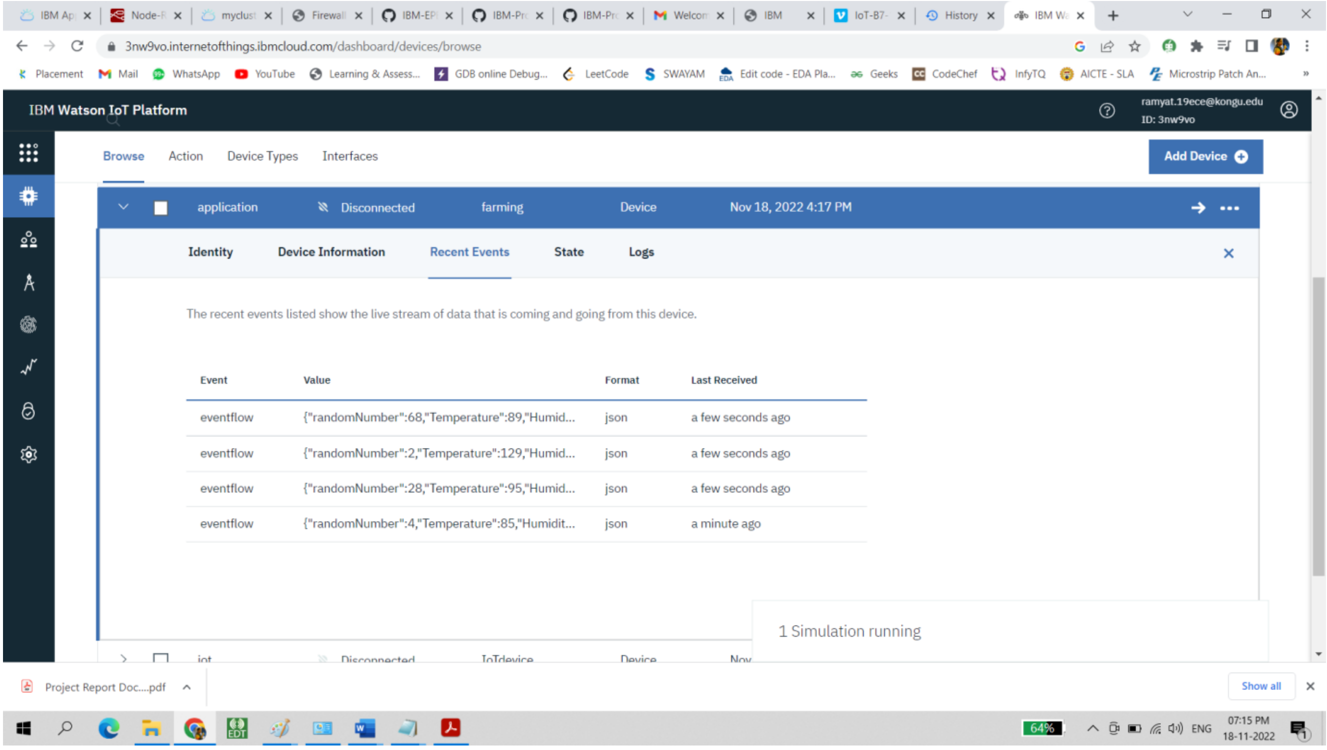
Specify the event payload in the editor window or by uploading a CSV file.

```
{
  "randomNumber": random(0, 100),
  "Temperature":random(80,150),
  "Humidity":random(60,110)
}
```



- You can see the received data in Recent Events under your device
- Data received in this format (json)

```
{
"Moisture":89,
"temp":96.0,
"Humid":89
}
```



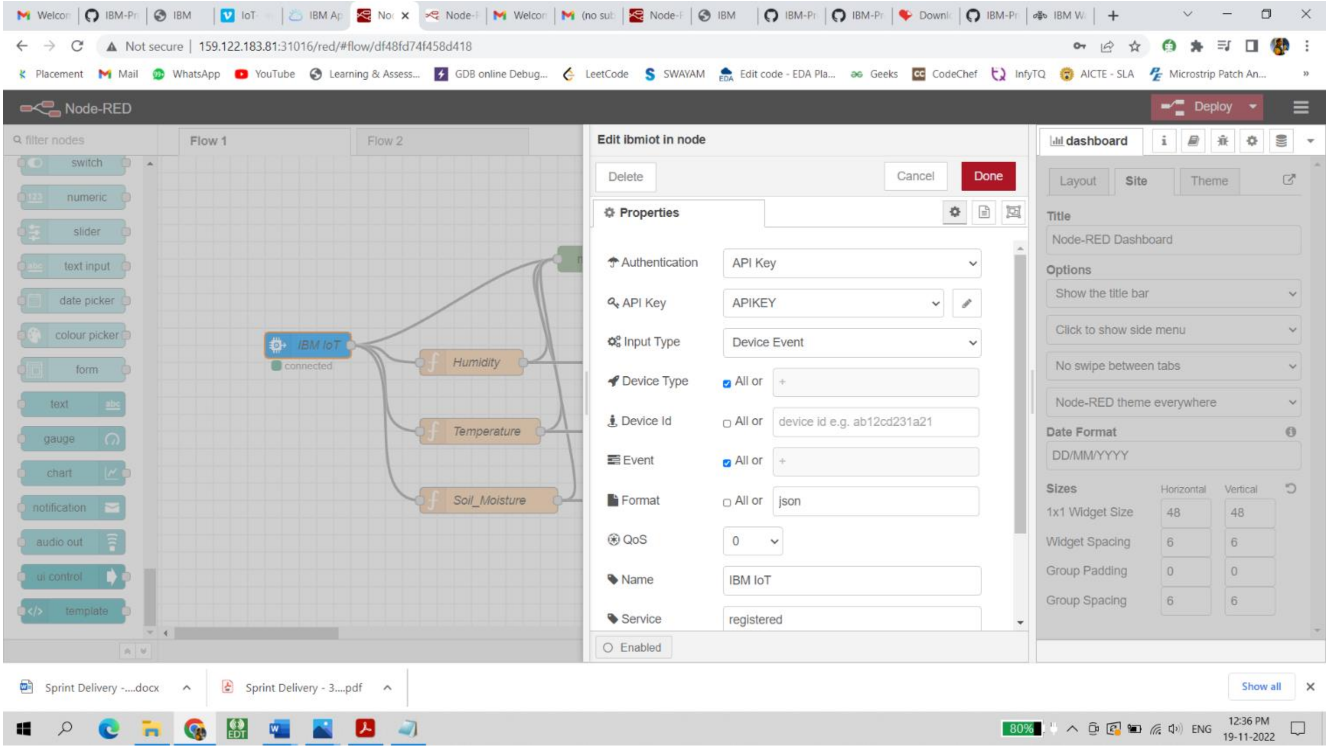
The recent events listed show the live stream of data that is coming and going from this device.

Event	Value	Format	Last Received
eventflow	{"randomNumber":68,"Temperature":89,"Humid...	json	a few seconds ago
eventflow	{"randomNumber":2,"Temperature":129,"Humid...	json	a few seconds ago
eventflow	{"randomNumber":28,"Temperature":95,"Humid...	json	a few seconds ago
eventflow	{"randomNumber":4,"Temperature":85,"Humidit...	json	a minute ago

Sprint	Functional Requirement(Epic)	User Story Number	User Story /Task
Sprint-3	Create Node Red Simulator	USN - 5	Create Node-Red Service and create a web application

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.



Edit ibmiot in node

Delete | Cancel | Done

Properties

Authentication: API Key

API Key: APIKEY

Input Type: Device Event

Device Type: ☑ All or +

Device Id: □ All or device id e.g. ab12cd231a21

Event: ☑ All or +

Format: □ All or json

QoS: 0

Name: IBM IoT

Service: registered

○ Enabled

Title: Node-RED Dashboard

Options: Show the title bar; Click to show side menu; No swipe between tabs; Node-RED theme everywhere

Date Format: DD/MM/YYYY

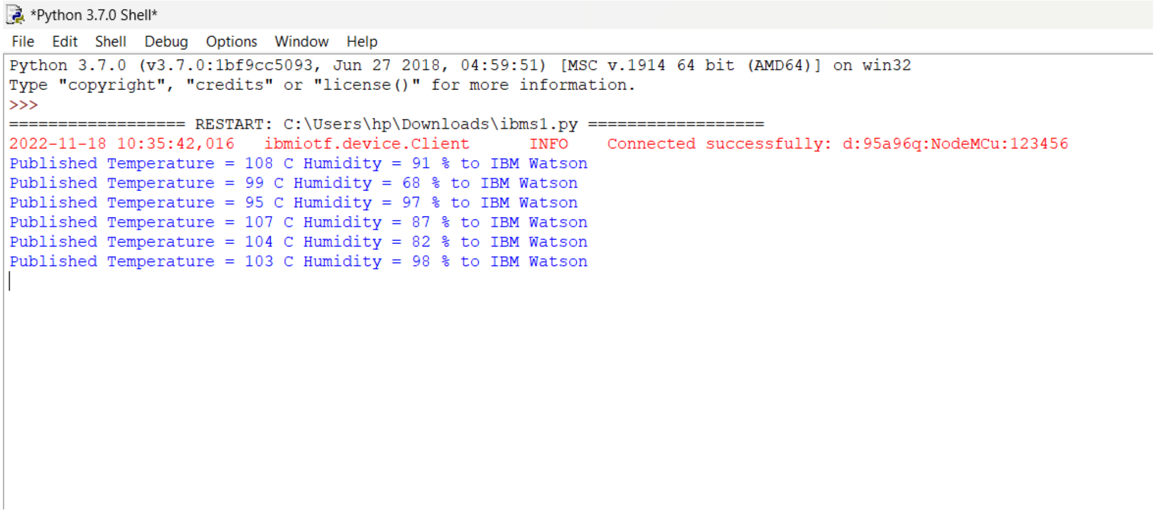
Sizes	Horizontal	Vertical
1x1 Widget Size	48	48
Widget Spacing	6	6
Group Padding	0	0
Group Spacing	6	6

- 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.Temperature**

```
return msg;
```

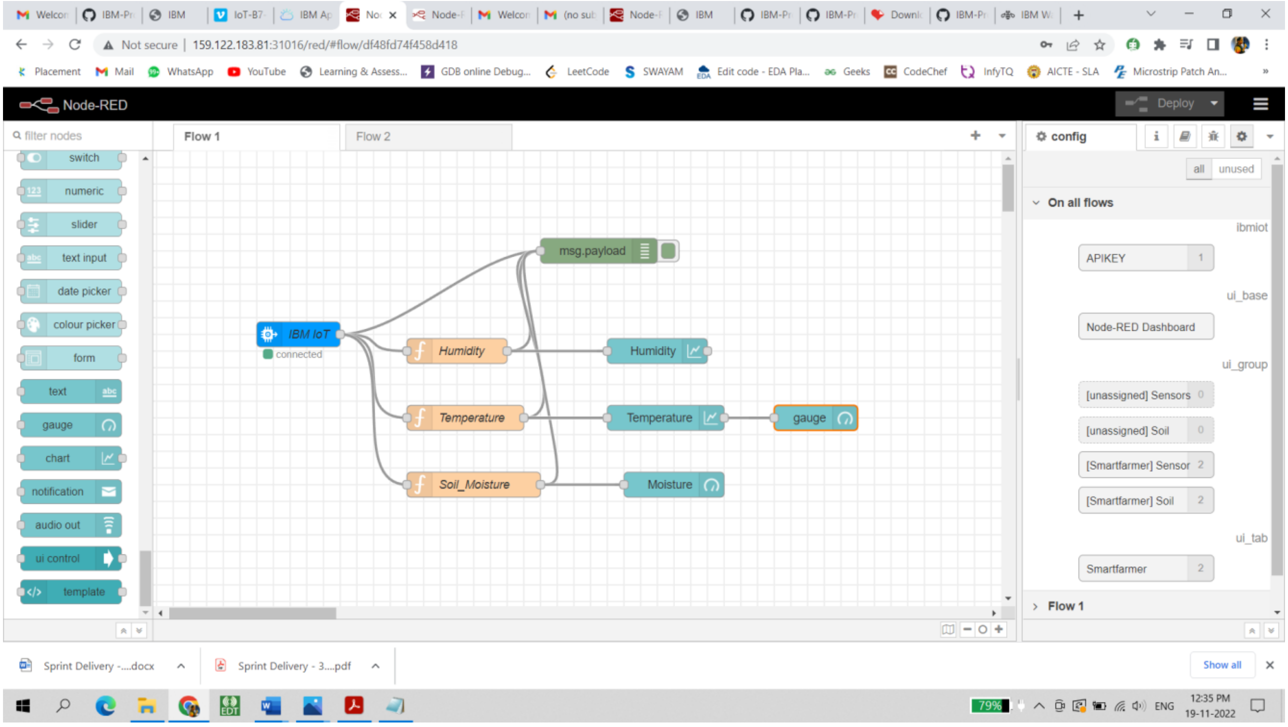
- Finally connect Gauge nodes from dashboard to see the data in UI.

- Data send by the python code



```
Python 3.7.0 (v3.7.0:1bf9cc5093, Jun 27 2018, 04:59:51) [MSC v.1914 64 bit (AMD64)] on win32
Type "copyright", "credits" or "license()" for more information.
>>>
================= RESTART: C:\Users\hp\Downloads\ibms1.py =================
2022-11-18 10:35:42,016  ibmiotf.device.Client      INFO    Connected successfully: d:95a96q:NodeMCu:123456
Published Temperature = 108 C Humidity = 91 % to IBM Watson
Published Temperature = 99 C Humidity = 68 % to IBM Watson
Published Temperature = 95 C Humidity = 97 % to IBM Watson
Published Temperature = 107 C Humidity = 87 % to IBM Watson
Published Temperature = 104 C Humidity = 82 % to IBM Watson
Published Temperature = 103 C Humidity = 98 % to IBM Watson
```

- Data received from the cloud in Node-Red console



- Nodes connected in following manner to get each reading separately.

Configuration of Node-Red to collect data from Open Weather

- The Node-Red also receive data from the Open Weather API by HTTPGET request. An inject trigger is added to perform HTTP request for every certain interval.

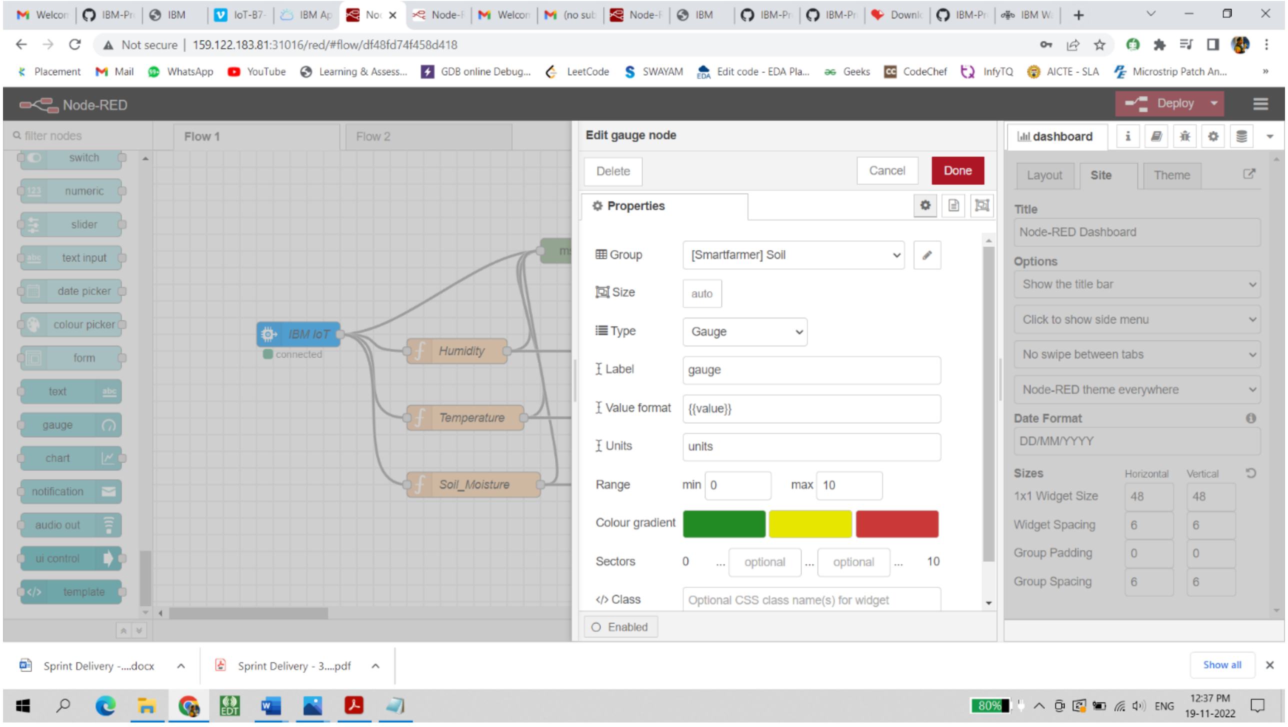
- The link to get open weather API : https://api.openweathermap.org/data/2.5/weather?lat=11.4383197&lon=77.5402674&appid=124d808d2039542453a0b1b05f37e900

- The data we receive from Open Weather after request is in below JSONformat.
- {"coord":{"lon":77.5403,"lat":11.4383},"weather":[{"id":804,"main":"Clouds","description":"overcast clouds","icon":"04d"}],"base":"stations","main":{"temp":300.33,"feels_like":303.19,"temp_min":300.33,"temp_max":300.33,"pressure":1009,"humidity":79,"sea_level":1009,"grnd_level":986},"visibility":10000,"wind":{"speed":2.3,"deg":113,"gust":3.05},"clouds":{"all":97},"dt":1668332957,"sys":{"country":"IN","sunrise":1668300334,"sunset":1668342165},"timezone":19800,"id":1270947,"name":"Gobichettipalayam","cod":200}

- In order to parse the JSON string we use Java script functions and geteach parameters

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

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



Edit gauge node

Delete Cancel Done

Properties

Group: [Smartfarmer] Soil

Size: auto

Type: Gauge

Label: gauge

Value format: {{value}}

Units: units

Range: min 0 max 10

Colour gradient: 🟩 🟨 🟥

Sectors: 0 ... optional ... optional ... 10

Class: Optional CSS class name(s) for widget

○ Enabled

dashboard — Layout | Site | Theme

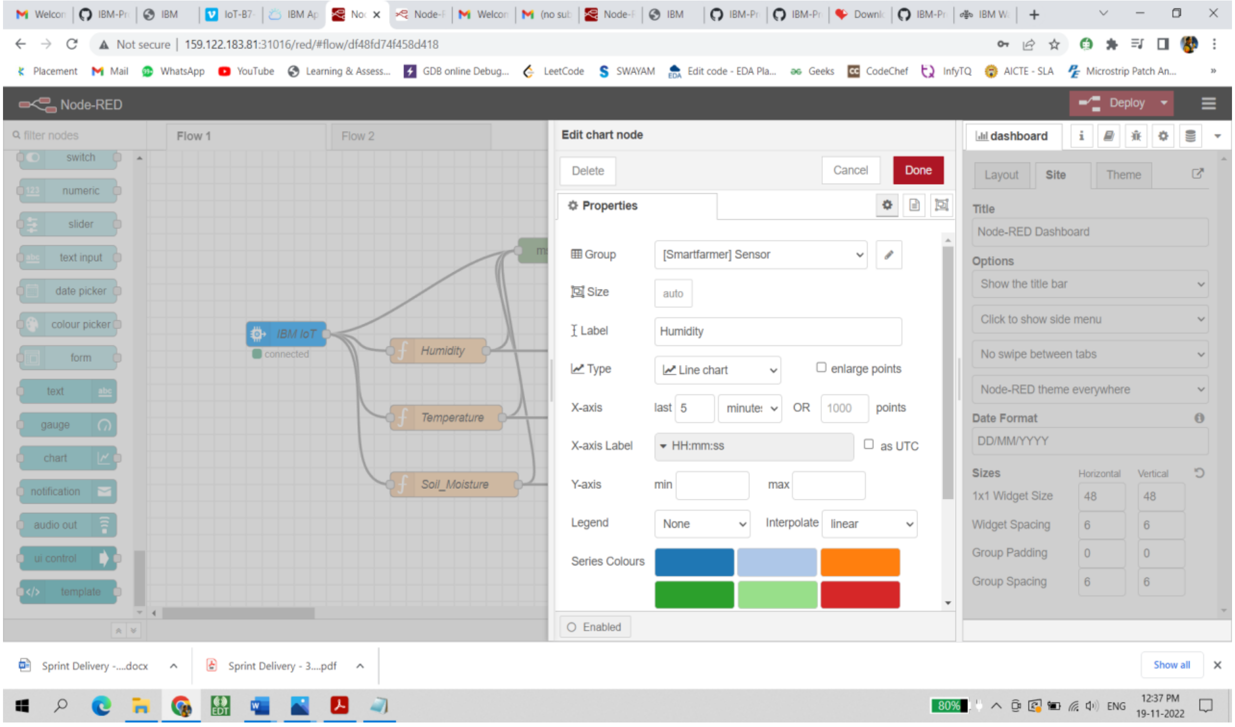
Title: Node-RED Dashboard

Options: Show the title bar; Click to show side menu; No swipe between tabs; Node-RED theme everywhere

Date Format: DD/MM/YYYY

Sizes	Horizontal	Vertical
1x1 Widget Size	48	48
Widget Spacing	6	6
Group Padding	0	0
Group Spacing	6	6

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



Edit chart node

Delete Cancel Done

Properties

Group: [Smartfarmer] Sensor

Size: auto

Label: Humidity

Type: Line chart □ enlarge points

X-axis: last 5 minute: OR 1000 points

X-axis Label: HH:mm:ss □ as UTC

Y-axis: min max

Legend: None Interpolate: linear

Series Colours

○ Enabled

dashboard — Layout | Site | Theme

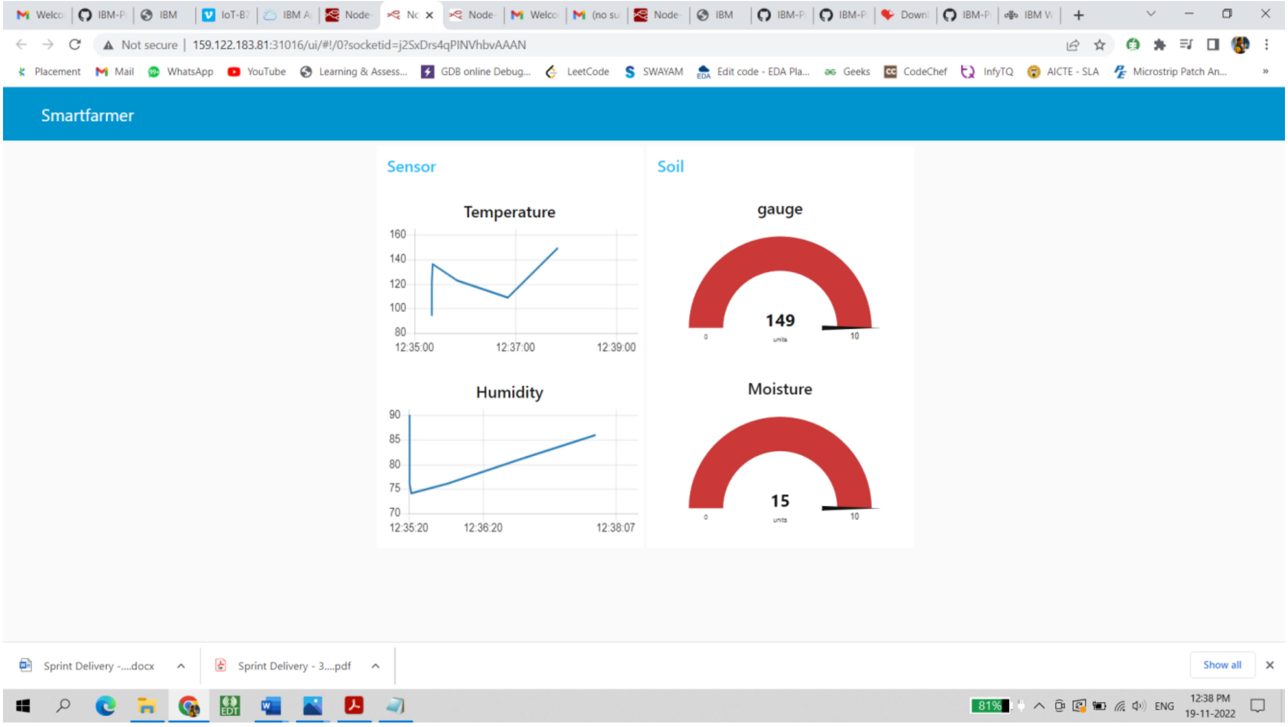
Title: Node-RED Dashboard

Options: Show the title bar; Click to show side menu; No swipe between tabs; Node-RED theme everywhere

Date Format: DD/MM/YYYY

Sizes	Horizontal	Vertical
1x1 Widget Size	48	48
Widget Spacing	6	6
Group Padding	0	0
Group Spacing	6	6

- You can the data in the node-red dashboard

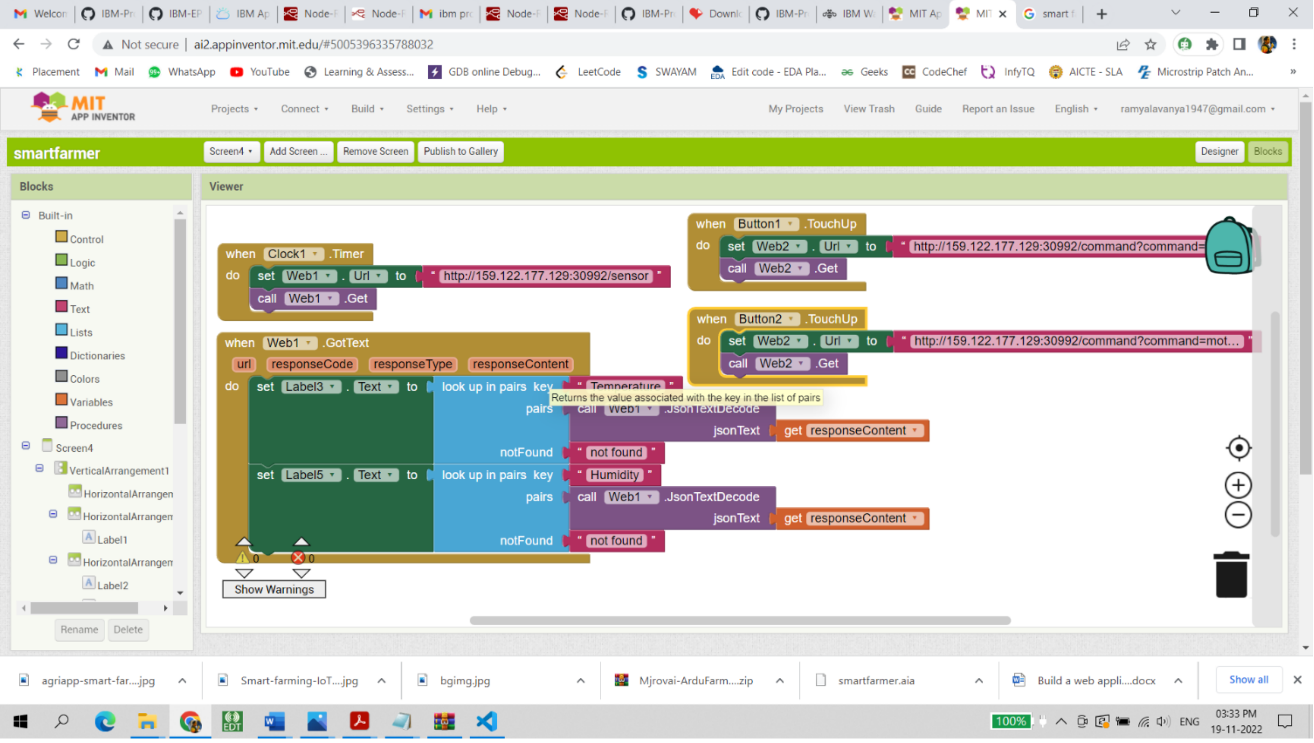

Not secure | 159.122.183.81:31016/ui/#!/0?socketid=j2SxDrs4qPINVhbvAAAN
Placement
Mail
WhatsApp
YouTube
Learning & Assess...
GDB online Debug...
LeetCode
SWAYAM
Edit code - EDA Pla...
Geeks
CodeChef
InfyTQ
AICTE - SLA
Microstrip Patch An...
Smartfarmer
Sensor
Temperature
160
140
120
100
80
12:35:00
12:37:00
12:39:00
Humidity
90
85
80
75
70
12:35:20
12:36:20
12:38:07
Soil
gauge
149
units
0
10
Moisture
15
units
0
10
Sprint Delivery -....docx
Sprint Delivery - 3....pdf
Show all
81%
ENG
12:38 PM
19-11-2022

Project Delivery Sprint 4

Date	17 Nov 2022
Team ID	PNT2022TMID04704
Project Name	Smart Farmer - IoT Enabled Smart Farming Application

Sprint	Functional Requirement (Epic)	User Story Number	User Story /Task
Sprint - 4	App Development	USN - 6	Add a user interface in a mobile app to monitor temperature, humidity and control the motor.

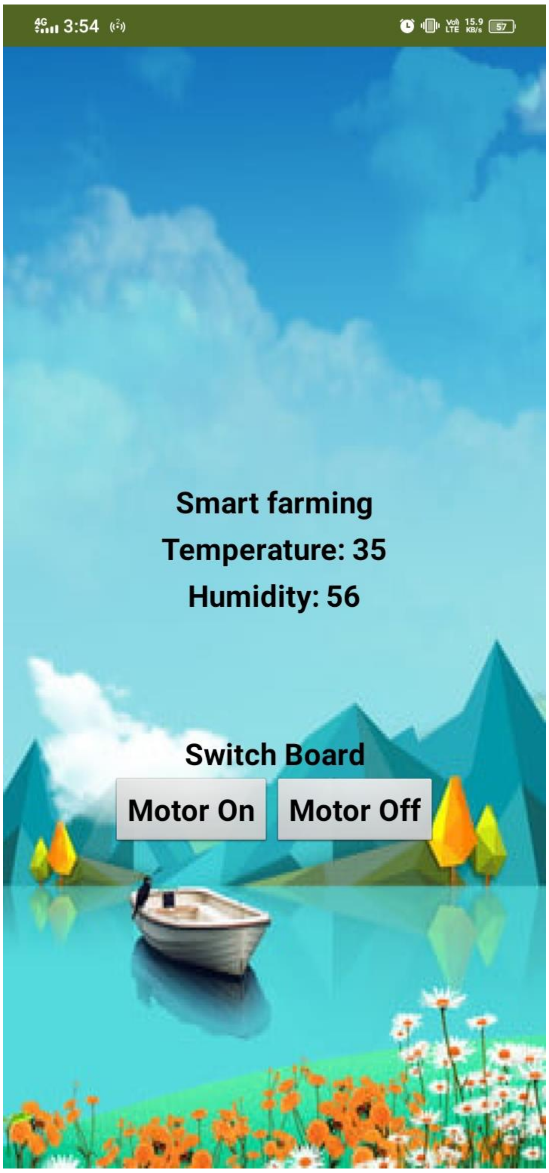
MOBILE APP



```
when Clock1 .Timer
do  set Web1 . Url to " http://159.122.177.129:30992/sensor "
    call Web1 .Get

when Web1 .GotText
  url  responseCode  responseType  responseContent
do  set Label3 . Text to look up in pairs key " Temperature "
                                      pairs call Web1 .JsonTextDecode
                                                 jsonText get responseContent
                                      notFound " not found "
    set Label5 . Text to look up in pairs key " Humidity "
                                      pairs call Web1 .JsonTextDecode
                                                 jsonText get responseContent
                                      notFound " not found "

when Button1 .TouchUp
do  set Web2 . Url to " http://159.122.177.129:30992/command?command=
    call Web2 .Get

when Button2 .TouchUp
do  set Web2 . Url to " http://159.122.177.129:30992/command?command=mot... "
    call Web2 .Get
```

Returns the value associated with the key in the list of pairs

- COMMAND RECEIVED FROM WEB UI AND MOBILE APP
  - MOTOR ON/OFF COMMAND

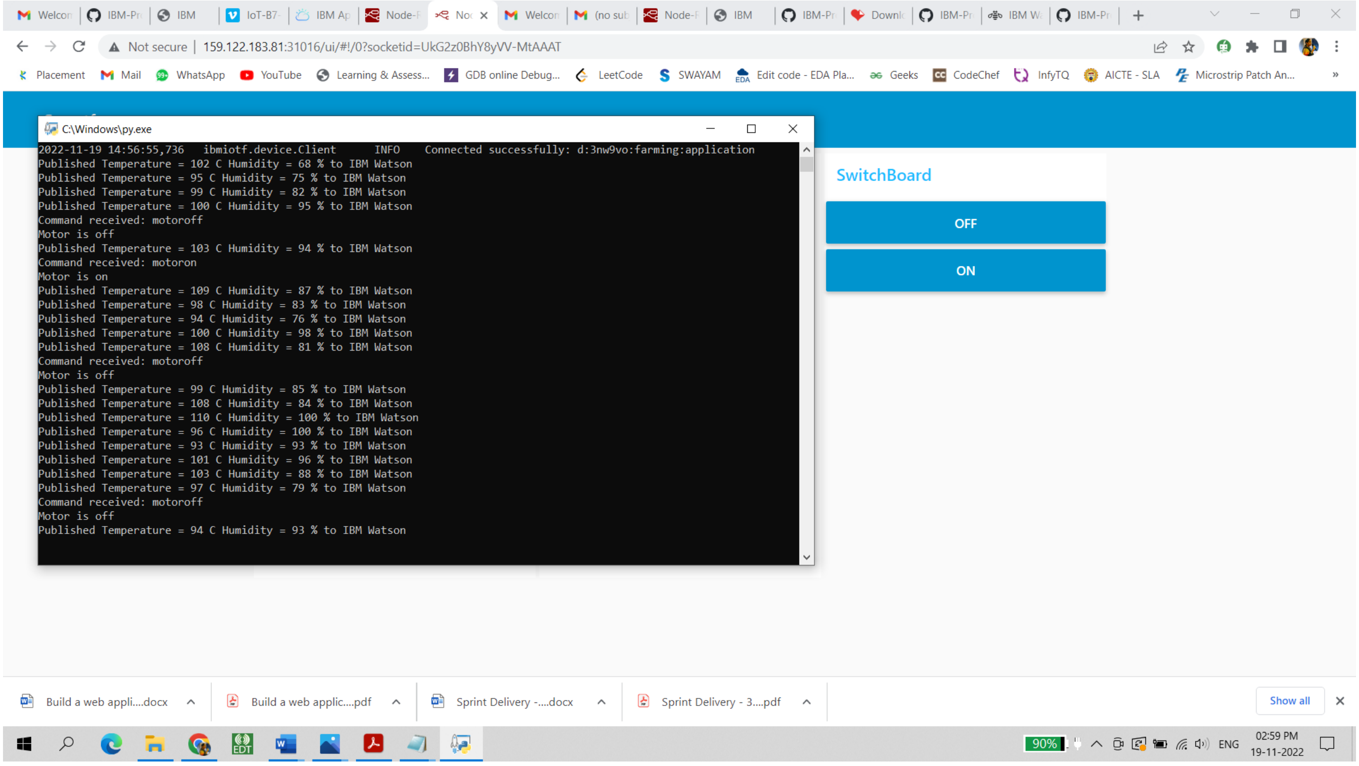


Smart farming

Temperature: 35

Humidity: 56

Switch Board



Not secure | 159.122.183.81:31016/ui/#!/0?socketid=UkG2z0BhY8yVV-MtAAAT
C:\Windows\py.exe
SwitchBoard
OFF
ON
Build a web appli....docx
Build a web applic....pdf
Sprint Delivery -....docx
Sprint Delivery - 3....pdf
Show all
90%
ENG
02:59 PM
19-11-2022

```
2022-11-19 14:56:55,736   ibmiotf.device.Client      INFO    Connected successfully: d:3nw9vo:farming:application
Published Temperature = 102 C Humidity = 68 % to IBM Watson
Published Temperature = 95 C Humidity = 75 % to IBM Watson
Published Temperature = 99 C Humidity = 82 % to IBM Watson
Published Temperature = 100 C Humidity = 95 % to IBM Watson
Command received: motoroff
Motor is off
Published Temperature = 103 C Humidity = 94 % to IBM Watson
Command received: motoron
Motor is on
Published Temperature = 109 C Humidity = 87 % to IBM Watson
Published Temperature = 98 C Humidity = 83 % to IBM Watson
Published Temperature = 94 C Humidity = 76 % to IBM Watson
Published Temperature = 100 C Humidity = 98 % to IBM Watson
Published Temperature = 108 C Humidity = 81 % to IBM Watson
Command received: motoroff
Motor is off
Published Temperature = 99 C Humidity = 85 % to IBM Watson
Published Temperature = 108 C Humidity = 84 % to IBM Watson
Published Temperature = 110 C Humidity = 100 % to IBM Watson
Published Temperature = 96 C Humidity = 100 % to IBM Watson
Published Temperature = 93 C Humidity = 93 % to IBM Watson
Published Temperature = 101 C Humidity = 96 % to IBM Watson
Published Temperature = 103 C Humidity = 88 % to IBM Watson
Published Temperature = 97 C Humidity = 79 % to IBM Watson
Command received: motoroff
Motor is off
Published Temperature = 94 C Humidity = 93 % to IBM Watson
```

ADVANTAGES

- Less labour cost.
- Field can be monitored the environment parameters and controlled the motor remotely.
- Better standards of living.
- Farmers can also monitor and control the farm field by Web UI.
- Increase in convenience to farmers.

DISADVANTAGES

- Farmers wanted to adapt the use of Mobile App.
- Lack of internet/connectivity issues.
- Added cost of internet and internet gateway infrastructure.

CONCLUSION

Thus, the objective of the project is to implement an IOT system in order to help farmers to control the motor function and monitor the environment parameters like temperature, humidity and soil moisture of their farms has been implemented successfully.