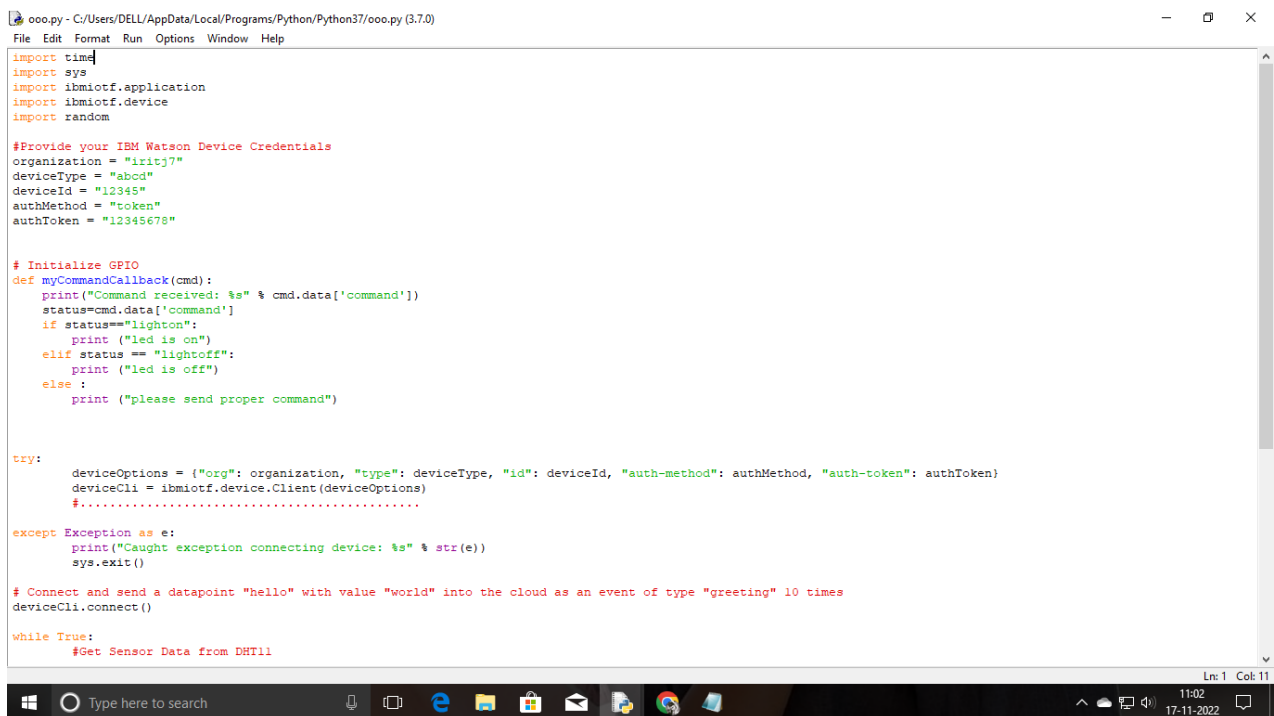


PROJECT DEVELOPMENT PHASE

SPRINT 3

TEAM ID	PNT2022TMID41422
PROJECT NAME	IOT BASESMART CROPPROTECTION SYSTEM FOR AGRICULTURE
DATE	31 OCTOBER 2022

STEP 1: Write a python code for randomize Soil Moisture ,Temperature, Humidity and Animal detection.



```
ooo.py - C:/Users/DELL/AppData/Local/Programs/Python/Python37/ooo.py (3.7.0)
File Edit Format Run Options Window Help

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

#Provide your IBM Watson Device Credentials
organization = "ixitj7"
deviceType = "abod"
deviceId = "12345"
authMethod = "token"
authToken = "12345678"

# Initialize GPIO
def myCommandCallback(cmd):
    print("Command received: %s" % cmd.data['command'])
    status=cmd.data['command']
    if status=="lighton":
        print ("led is on")
    elif status == "lightoff":
        print ("led 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
```

STEP 2: Run the python code it send data to IBM IoT Watson Platform.

The screenshot shows the IBM Watson IoT Platform dashboard. The main view displays a table of recent events for a device named 'abcd'. The table has two columns: 'Event' and 'Value'. The events are from 'IoTSensor' and contain JSON payloads with temperature, humidity, moisture, and animal detection data.

A configuration window for a new event type is open. It shows the following details:

- Device Type: abcd
- Event type name: event_1
- Schedule: 20, Every Minute
- Payload: A JSON object with random values for temperature, humidity, moisture, and animal detection.

```
0 {  
1   "temp": random(90,110)  
2   "Humid": random(60,100)  
3   "Moist": random(0,100)  
4   "Animal_dect": random(0,2)  
5 }  
6 }
```

STEP 3: Open Node-RED flow dashboard.

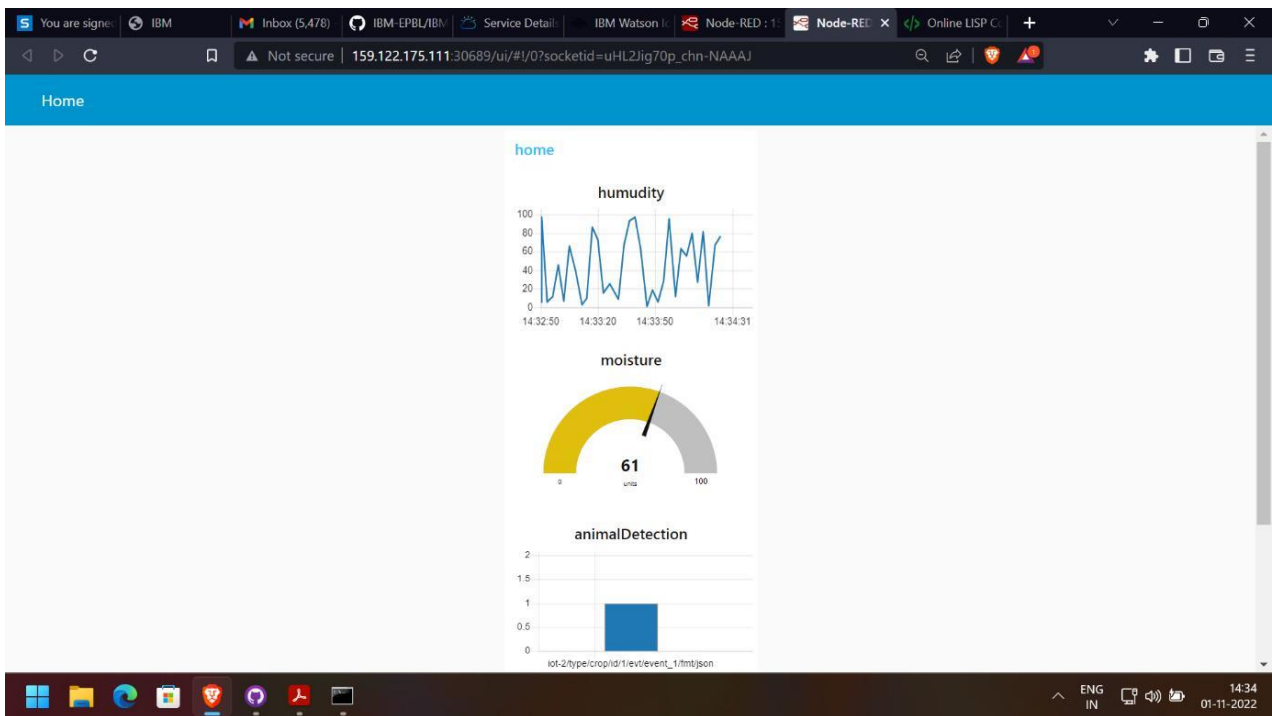
The screenshot shows the Node-RED flow dashboard. The main view displays a flow diagram with the following components:

- Flow 1:** A 'Node red' node connected to a 'msg.payload' node.
- Flow 3:** A 'Node red' node connected to four function nodes: 'temp', 'Humid', 'Moist', and 'Animal_dect'. These function nodes are connected to four output nodes: 'Temperature', 'Humidity', 'Moisture', and 'Animal Detection'.
- Flow 2:** A '[get] /sensor' node connected to a 'function 2' node, which is then connected to an 'http' node.

The right sidebar shows the 'info' panel with the following details:

- Node: "62b90379c54b1921"
- Type: inject

STEP 4: Open Node-RED user interface to show the Soil Moisture, Humidity and Temperature value in gauge.



PYTHON CODE :

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

#Provide your IBM Watson Device Credentials

```
organization = "iritj7"
deviceType = "abcd"
deviceId = "12345"
authMethod = "token"
authToken = "12345678"
```

Initialize GPIO

```
def myCommandCallback(cmd):
    print("Command received: %s" % cmd.data['command'])
    status=cmd.data['command']
    if status=="lighton":
        print ("led is on")
    elif status == "lightoff":
```

```

        print ("led 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)
    Moist=random.randint(20,100)
    Animal_dect=random.randint(1,20)

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

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

    deviceCli.commandCallback = myCommandCallback

# Disconnect the device and application from the cloud

```

deviceCli.disconnect(

