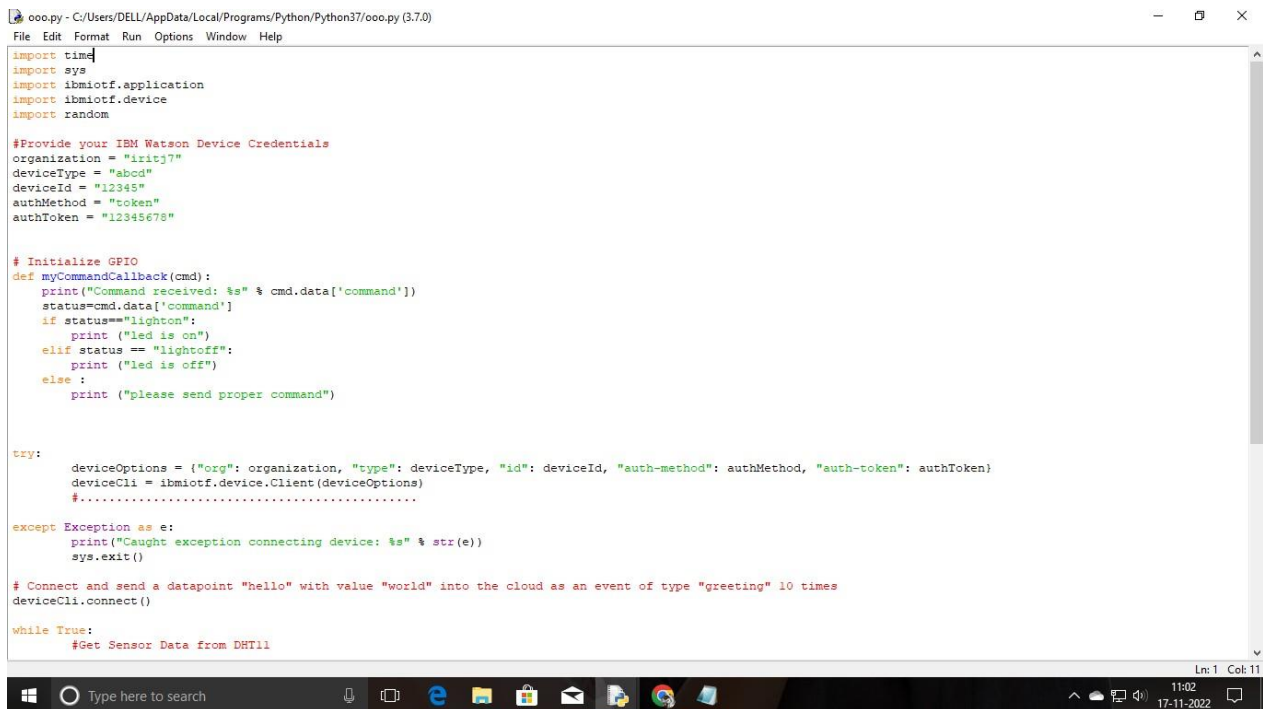


PROJECT DEVELOPMENT PHASE

SPRINT-3

Team ID	PNT2022TMID22100
Project Name	IoT Based Smart Crop Protection System for Agriculture
Date	20-November-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 = "iirtj7"
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 and send data to IBM IoT Watson Platform.

The screenshot shows the IBM IoT Platform console. On the left, a table lists recent events for a device with ID 12345. The events are in JSON format, containing temperature, humidity, and motion data. On the right, a configuration panel for a NODEMCU device is shown, including event type, schedule, and a Python payload for generating random data.

Event	Value	Format	Last Received
event_1	["temp":14,"humid":69,"motionanimal":87]	json	a few seconds ago
event_1	["temp":50,"humid":95,"motionanimal":33]	json	a minute ago
event_1	["temp":81,"humid":2,"motionanimal":81]	json	a minute ago
event_1	["temp":12,"humid":98,"motionanimal":11]	json	a minute ago
event_1	["temp":30,"humid":89,"motionanimal":4]	json	a minute ago

Device Type: NODEMCU

Event type name: event_1

Schedule: 20 Every Minute

Payload:

```
{
  "temp": random(0, 100),
  "humid": random(0, 100),
  "motionanimal": random(0, 100)
}
```

STEP 3: Open Node-RED flow dashboard.

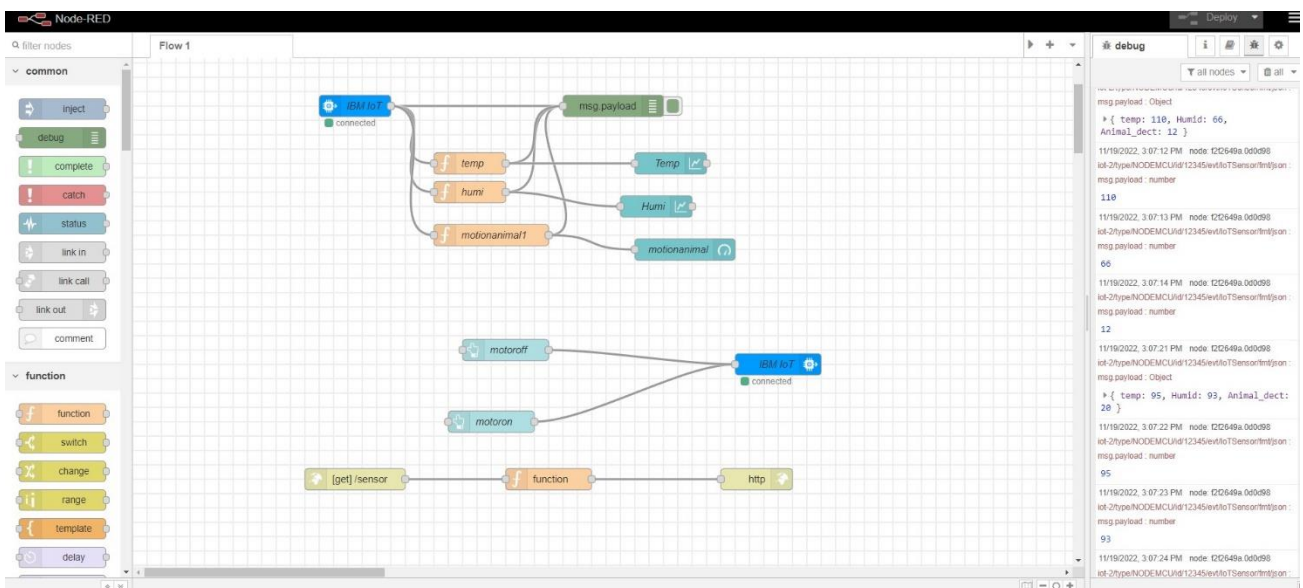
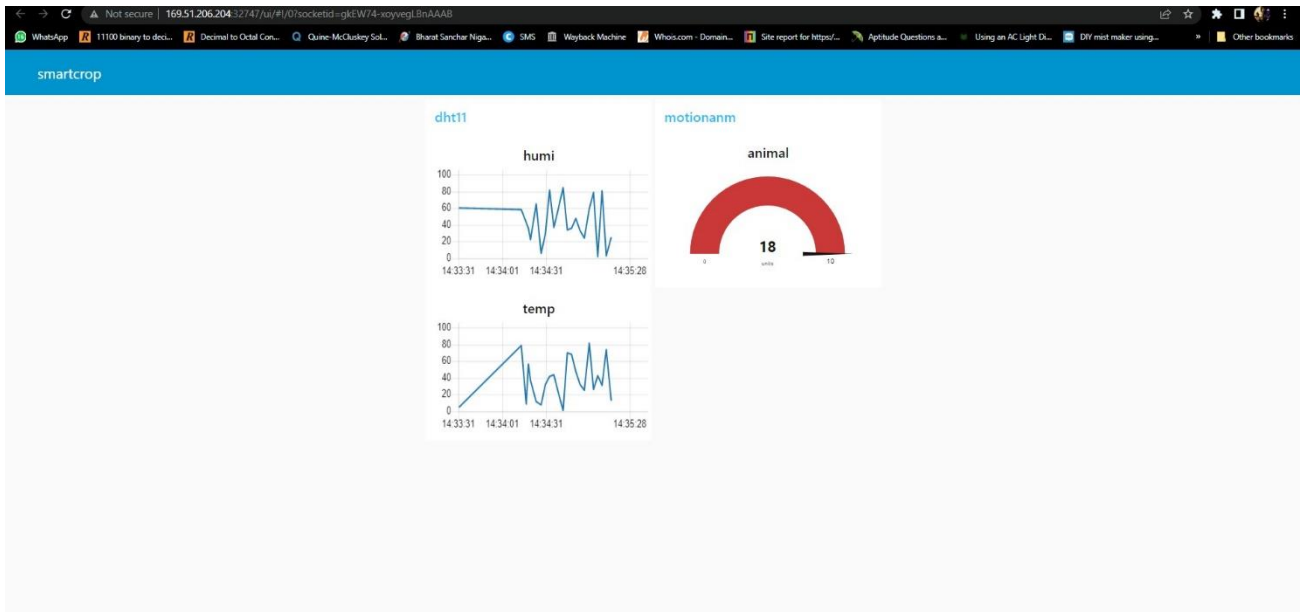
The screenshot shows the Node-RED flow dashboard. A flow is configured to receive data from an IBM IoT device. The data is split into three streams: temperature, humidity, and motion data. Each stream is then processed by a corresponding output node (Temp, Humi, motionanimal) and sent to a msg payload node. The debug console on the right shows the incoming data and the output of the flow.

```
graph LR
    IoT[IBM IoT] --> Split(( ))
    Split --> Temp[temp]
    Split --> Humi[humid]
    Split --> Motion[motionanimal]
    Temp --> TempOut[Temp]
    Humi --> HumiOut[Humi]
    Motion --> MotionOut[motionanimal]
    TempOut --> Payload[msg payload]
    HumiOut --> Payload
    MotionOut --> Payload
```

Debug Console:

```
11/18/2022, 2:49:44 PM node: IoT649a 0d0d58
lat: 27.9561,lon: 86.9411,alt: 12.345,dev: 12345,event: 1,ts: 1671456789
msg.payload: {
  temp: 14,
  humid: 69,
  motionanimal: 87
}
```

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 IoT")
    time.sleep(10)
```

```
deviceCli.commandCallback = myCommandCallback
```

```
# Disconnect the device and application from the cloud
```

```
deviceCli.disconnect()
```