SPRINT 4 – To receive commands from IBM Cloud using Python program and testing of user interaction

Date:	17th November 2022
Team ID	PNT2022TMID27964
Project Name	Project – Smart Farmer- IoT basedSmartFarmingApplication

AIM:

To receive the data from mobile app to IBM Cloud and to develop a python program to receive commands from cloud.

SOFTWARES USED:

- IBM Cloud
- IBM Watson for IoT
- Node RED
- MIT App Inventer
- Python

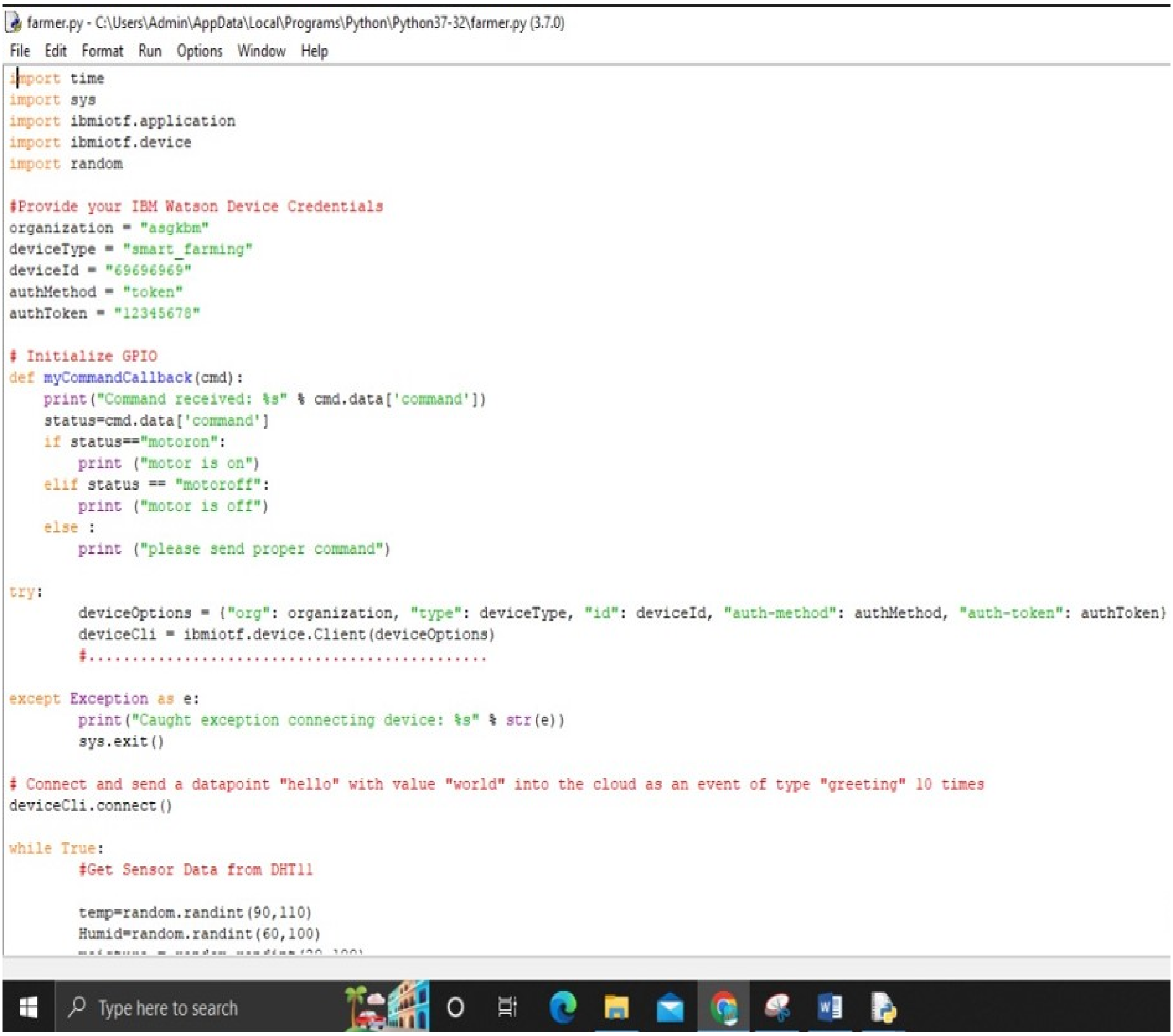
PROCEDURE :

- A python code is developed to receive the data from IBM Cloud regarding motor operation.
- The app is linked with Node RED and has got buttons regarding motor operation.
- Once the sensor data is received and if motor on button is pressed on app then the motor in the field connected to IoT platform should get turned on.
- This is done by processing the data back to IBM Cloud from the mobile app through the Node RED platform.
- Once the data is received in the IBM Cloud, the data is received by the IoT device which is done using the previously developed python code.
- The cloud and IoT device credentials are given in the code which then receives the data.
- Thus, the motor can be turned ON and OFF through this.

PYTHON PROGRAM :

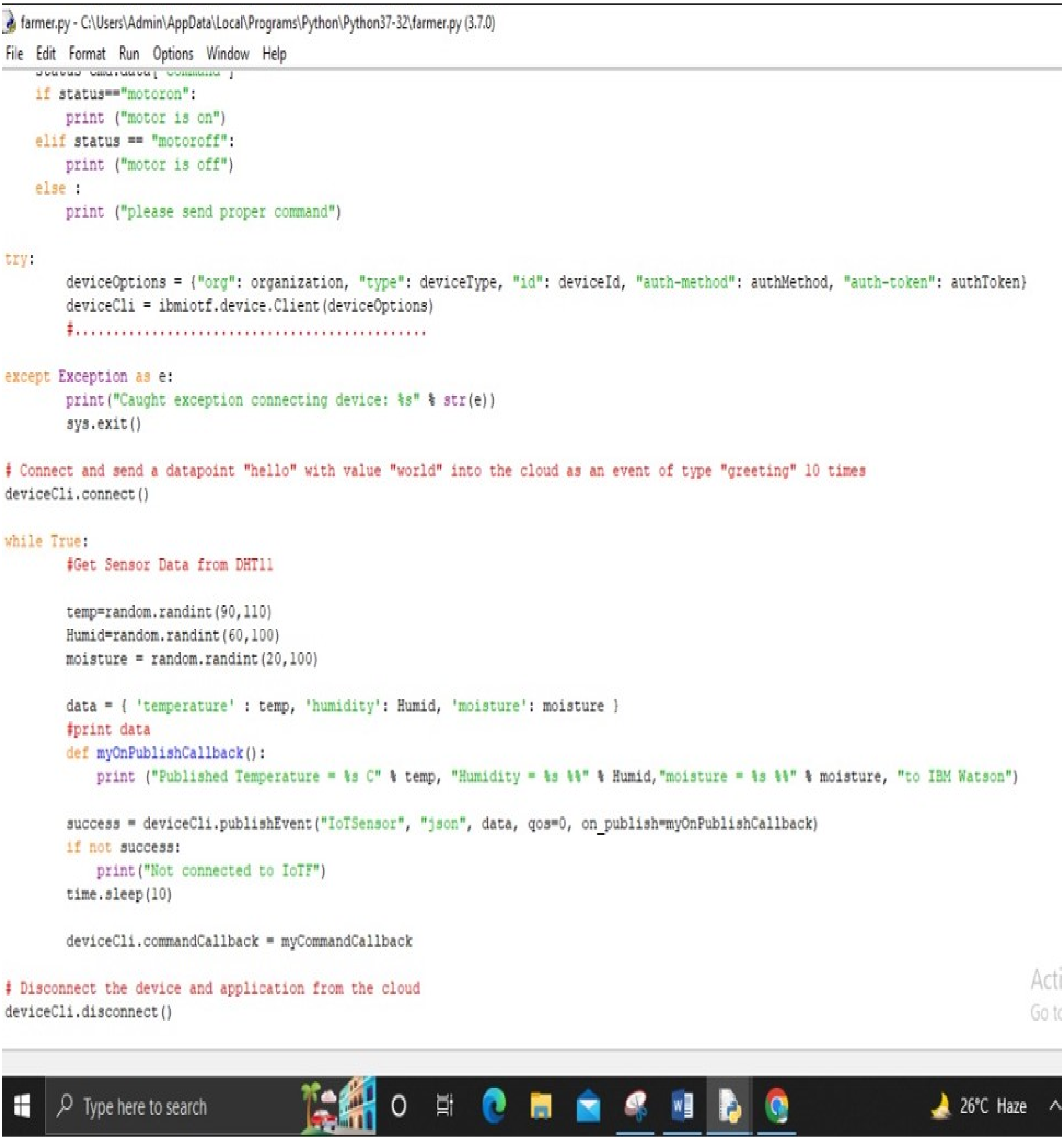
```
import time importsys
import ibmiotf.application import
ibmiotf.device importrandom
#Provide your IBM Watson Device Credentials
organization = "asgkbm" deviceType = "smart_farming"
deviceId = "69696969" authMethod = "token"
authToken = "12345678"
# Initialize GPIO
def myCommandCallback(cmd):
print("Command 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 anevent of type
  "greeting" 10 times deviceCli.connect()


  while True:
#Get Sensor Data from DHT11temp=random.randint(90,110) Humid=random.randint(60,100)
Mois=random.
```

```
Randint(20,120)
    data = { 'temp' : temp, 'Humid': Humid ,‘Mois’: Mois}
      #print data
defmyOnPublishCallback():
          print ("Published Temperature = %s C" % temp, "Humidity = %s %%" %Humid, “Moisture =%s
 deg c” % Mois “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()
```



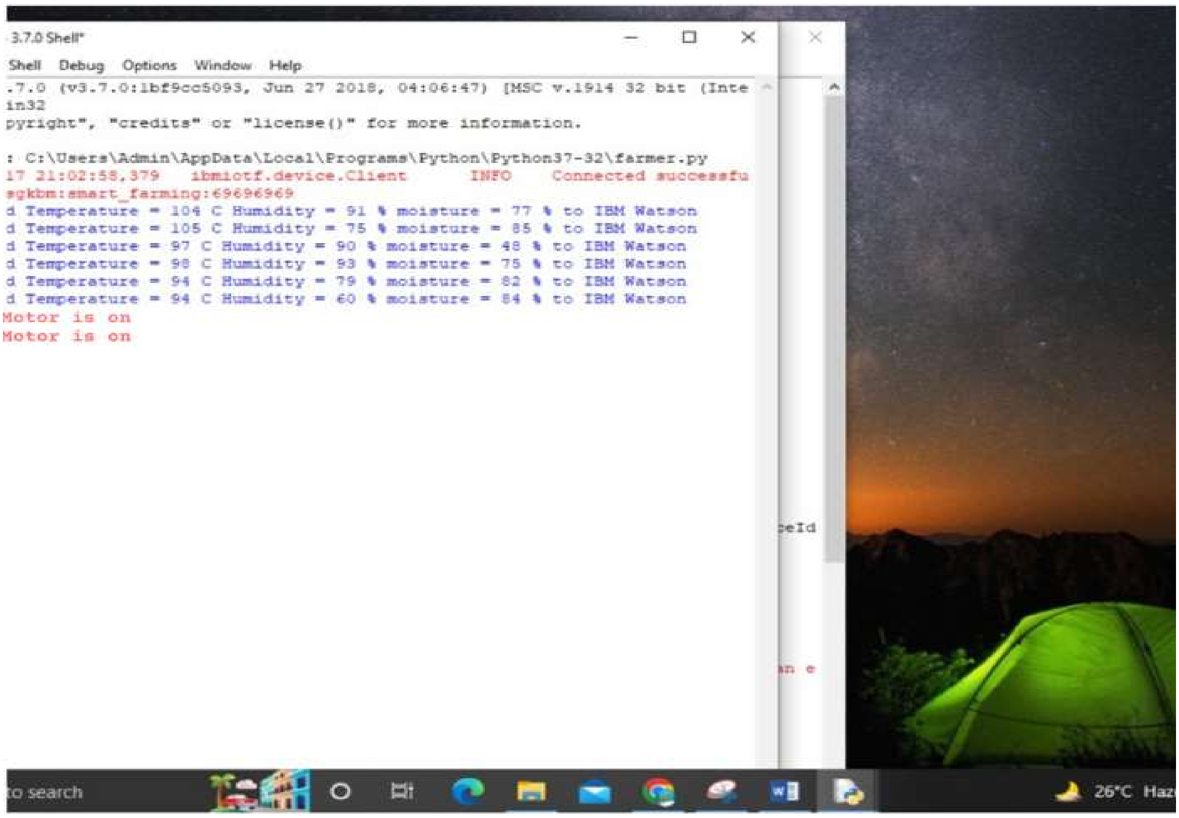
farmer.py - C:\Users\Admin\AppData\Local\Programs\Python\Python37-32\farmer.py (3.7.0)
File Edit Format Run Options Window Help
Type here to search

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

#Provide your IBM Watson Device Credentials
organization = "asgkbm"
deviceType = "smart_farming"
deviceId = "69696969"
authMethod = "token"
authToken = "12345678"

# 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)
```


farmer.py - C:\Users\Admin\AppData\Local\Programs\Python\Python37-32\farmer.py (3.7.0)
File Edit Format Run Options Window Help
Type here to search
26°C Haze
Acti
Go to

```
    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)
        moisture = random.randint(20,100)

        data = { 'temperature' : temp, 'humidity': Humid, 'moisture': moisture }
        #print data
        def myOnPublishCallback():
            print ("Published Temperature = %s C" % temp, "Humidity = %s %%" % Humid,"moisture = %s %%" % moisture, "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()
```

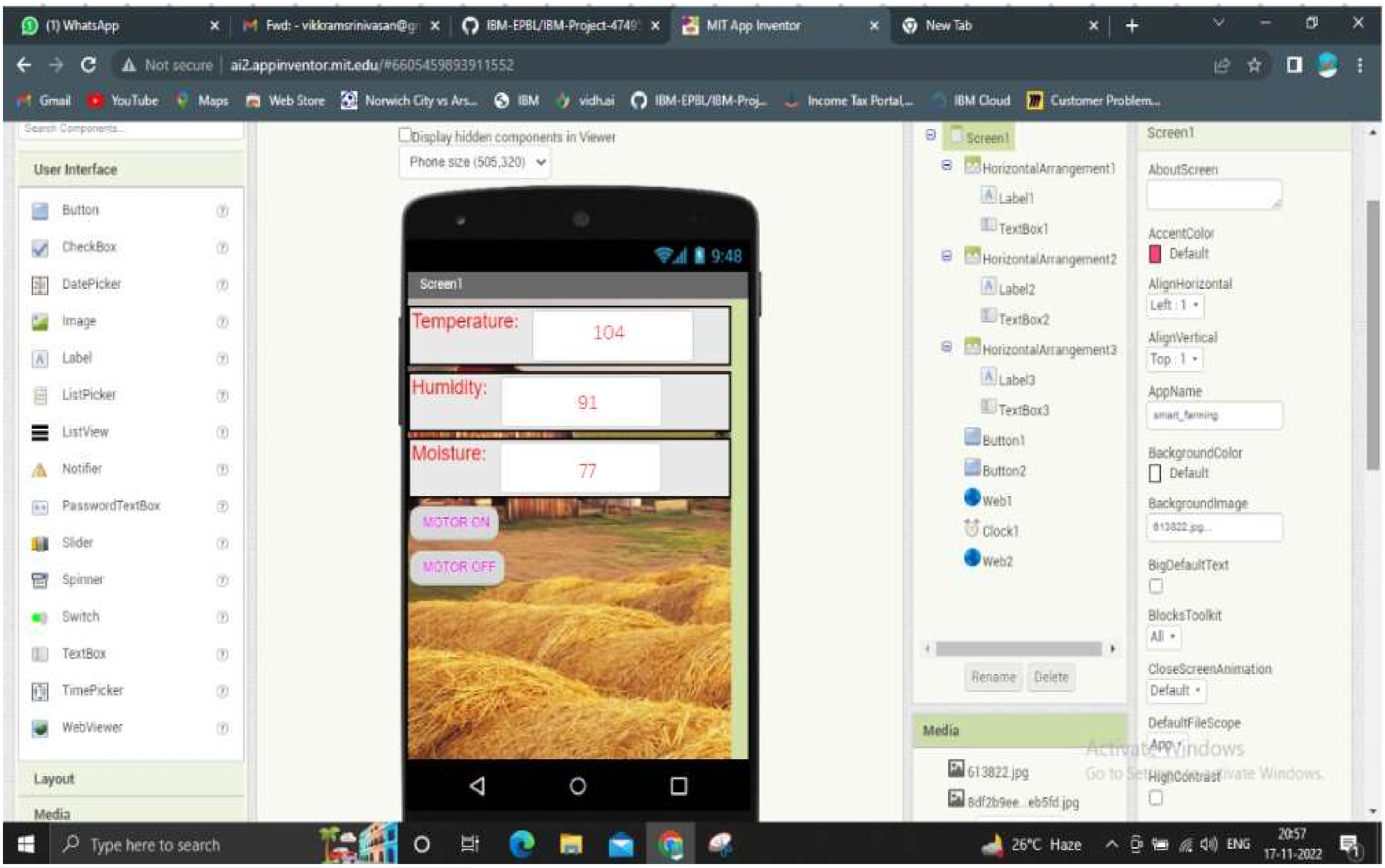
NODE RED PROGRAM FLOW



```
.7.0 (v3.7.0:1bf9cc5093, Jun 27 2018, 04:06:47) [MSC v.1914 32 bit (Inte
in32
pyright", "credits" or "license()" for more information.

: C:\Users\Admin\AppData\Local\Programs\Python\Python37-32\farmer.py
17 21:02:58,379   ibmiotf.device.Client      INFO    Connected successfu
sgkbm:smart_farming:69696969
d Temperature = 104 C Humidity = 91 % moisture = 77 % to IBM Watson
d Temperature = 105 C Humidity = 75 % moisture = 85 % to IBM Watson
d Temperature = 97 C Humidity = 90 % moisture = 48 % to IBM Watson
d Temperature = 98 C Humidity = 93 % moisture = 75 % to IBM Watson
d Temperature = 94 C Humidity = 79 % moisture = 82 % to IBM Watson
d Temperature = 94 C Humidity = 60 % moisture = 84 % to IBM Watson
Motor is on
Motor is on
```

OBSERVATIONS AND RESULTS

- ## MOBILE APPLICATION OUTPUT

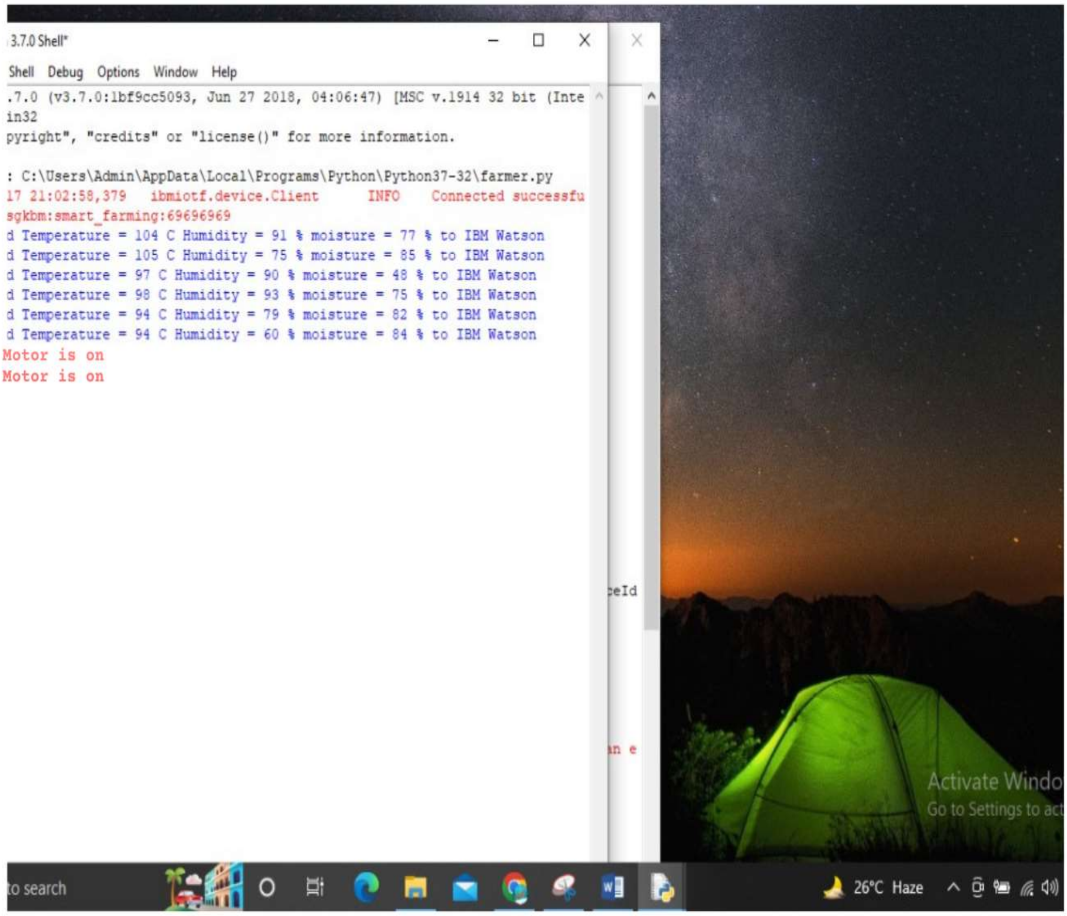


Temperature: 104

Humidity: 91

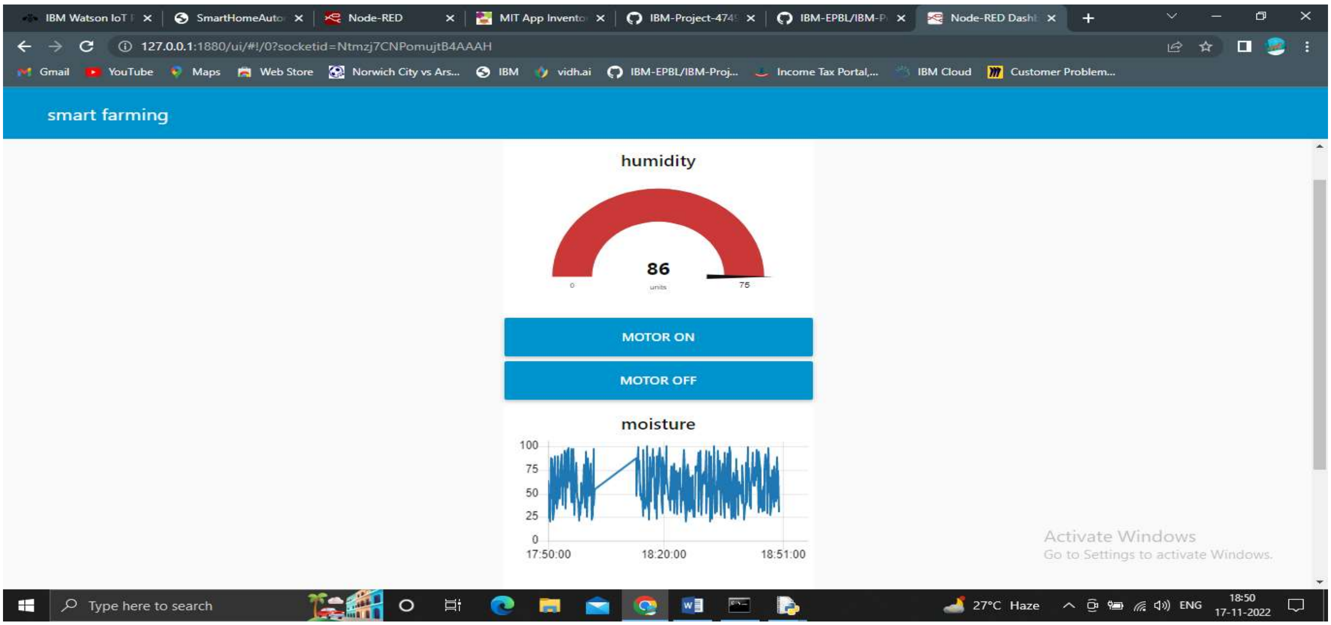
Moisture: 77

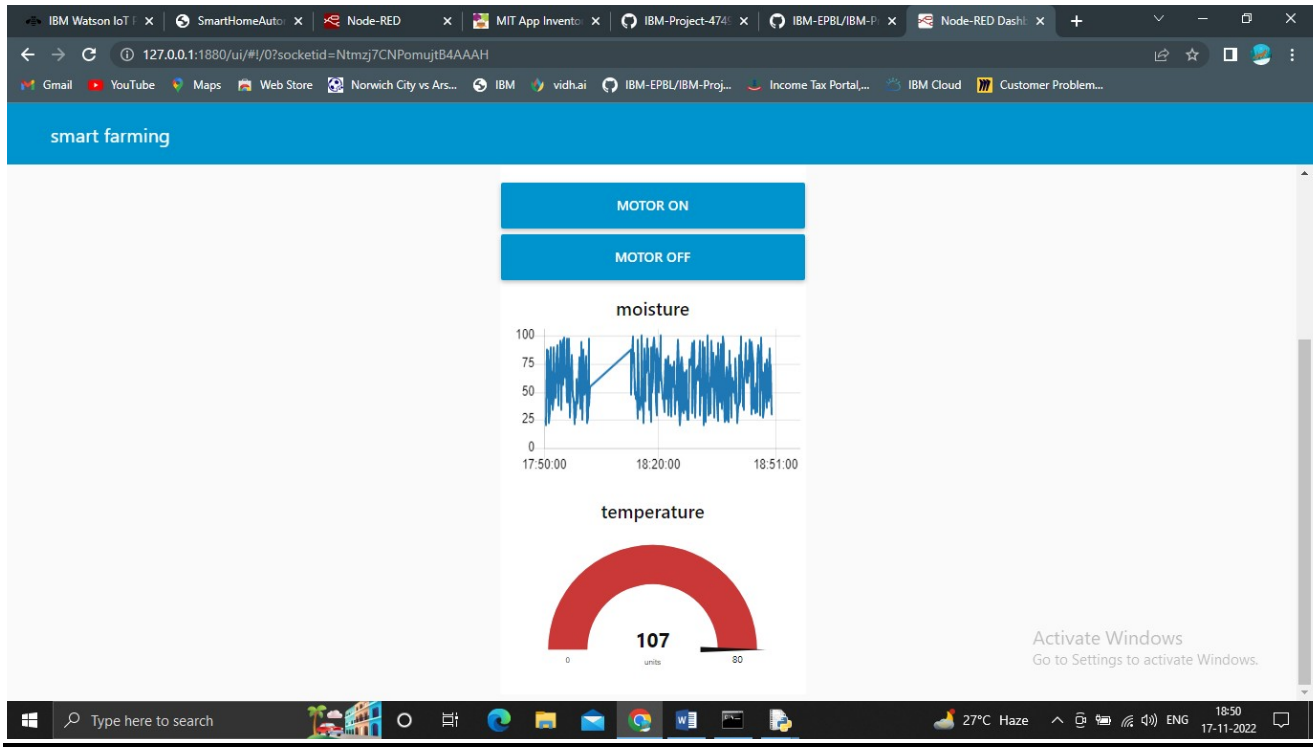
- **PYTHON CODE OUTPUT**



```
.7.0 (v3.7.0:1bf9cc5093, Jun 27 2018, 04:06:47) [MSC v.1914 32 bit (Inte
in32
pyright", "credits" or "license()" for more information.

: C:\Users\Admin\AppData\Local\Programs\Python\Python37-32\farmer.py
17 21:02:58,379   ibmiotf.device.Client      INFO    Connected successfu
sgkbm:smart_farming:69696969
d Temperature = 104 C Humidity = 91 % moisture = 77 % to IBM Watson
d Temperature = 105 C Humidity = 75 % moisture = 85 % to IBM Watson
d Temperature = 97 C Humidity = 90 % moisture = 48 % to IBM Watson
d Temperature = 98 C Humidity = 93 % moisture = 75 % to IBM Watson
d Temperature = 94 C Humidity = 79 % moisture = 82 % to IBM Watson
d Temperature = 94 C Humidity = 60 % moisture = 84 % to IBM Watson
Motor is on
Motor is on
```

WEB UI OUTPUT





ADVANTAGES AND DISADVANTAGES

Advantages:

- Farms can be monitored and controlled remotely from anywhere.
- Increase in convenience to farmers.
- Less labor cost.
- Better standards of living.

Disadvantages:

- Lack of internet/connectivity issues.
- Added cost of internet and internet gateway infrastructure.
- Mobile phone is necessary for using the app.

CONCLUSION

Thus the objective of the project to implement an IoT system in order to helpfarmers to control and monitor their farms has been implemented successfully