

SPRINT-3
PROJECT DEVELOPMENT PHASE

Date	8 Nov 2022
Team Id	PNT2022TMID49537
Project Name	SmartFarmer -IoT Enabled Smart Farming Application
Maximum Mark	8 Marks

Coding:

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

```
#Provide your IBM Watson Device Credentials
organization = "puubdh"
deviceType = "raspberrypi"
deviceId = "demo123"
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 ison")
    else :
        print ("motor isoff")
```

```
#print(cmd)
```

```
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(0,100)
    hum=random.randint(0,100)
    moist=random.randint(0,100
    )

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

        success = deviceCli.publishEvent("IoTSensor", "json", data, qos=0,
on_publish=myOnPublishCallback)
        if not success:
            print("Not connected to IoTF")
            time.sleep(5)

    deviceCli.commandCallback = myCommandCallback

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

```


TEST CASE

Recent Event:

The screenshot displays the IBM Watson IoT Platform dashboard. The top navigation bar includes tabs for 'Browse', 'Action', 'Device Types', and 'Interfaces'. A search bar labeled 'Search by Device ID' is present. The main content area shows a table of devices, with 'demo123' selected. Below the device list, the 'Recent Events' tab is active, displaying a table of live data events.

Event	Value	Format	Last Received
IoTSensor	{"temp":56,"hum":95,"moist":54}	json	a few seconds ago
IoTSensor	{"temp":59,"hum":35,"moist":77}	json	a few seconds ago
IoTSensor	{"temp":45,"hum":97,"moist":5}	json	a few seconds ago

Node-Red Output:

Smart Agriculture:

The screenshot shows the Node-RED web interface in a browser. The active flow is named "smart agriculture". It features an "IBM IoT" node connected to three sensor nodes: "Soil Moisture", "Humidity", and "Temperature". Each sensor node is connected to a corresponding "msg.payload" node, which then connects to an "http" node. Below this, there are "Motor ON" and "Motor OFF" nodes connected to an "IBM IoT" node, which also connects to a "msg.payload" node and an "http" node. A "GET /data" node is connected to a "data" node, which then connects to an "http" node. The "debug" console on the right shows a series of messages with payloads: 72, 40, 92, and 60. The browser tabs include IBM, Application Details, Node-RED, IBM Watson IoT Platform, WhatsApp, and Python Release Python.

OpenWeather:

The screenshot shows the Node-RED web interface in a browser. The active flow is named "Open Weather". It starts with a "timestamp" node connected to a "Madurai" node, which then connects to a "function" node. This "function" node is connected to five other "function" nodes, each of which is connected to a "msg.payload" node. These "msg.payload" nodes are then connected to "Location", "Weather", "Temperature", "Humidity", and "Wind Speed" nodes. A "GET /details" node is connected to a "function" node, which then connects to an "http" node. The "debug" console on the right shows a series of messages with payloads: "Madurai", "Haze", 30, 66, and 3.6. The browser tabs include IBM, Application Details, Node-RED, IBM Watson IoT Platform, WhatsApp, and Python Release Python.

Output:

```
ibmiotpublishsubscribe (2).py - F:\cse4\Downloads\ibmiotpublishsubscribe (2).py (3.7.0)
File Edit Format Run Options Window Help

import time
import sys
import ibmiotf.appl
import ibmiotf.device
import random

#Provide your IBM Watson organization = "puubdh:raspberrypi:demo123"
deviceType = "raspberrypi"
deviceId = "demo123"
authMethod = "token"
authToken = "1234567890"

# Initialize GPIO

def myCommandCallback(status=cmd.data):
    print("Command received: " + status)
    if status=="motor on":
        print ("motor is on")
    elif status=="motor off":
        print ("motor is off")
    else:
        print ("motor is off")
    #print(cmd)

try:
    deviceOptions = {}
    deviceCli = ibmiotf.device.Client(organization, deviceId, authMethod, authToken)
    #.....
except Exception as e:
    print("Caught exception: " + str(e))
    sys.exit()

# Connect and send data
deviceCli.connect()
```

```
Python 3.7.0 Shell
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: F:\cse4\Downloads\ibmiotpublishsubscribe (2).py =====
2022-11-08 11:43:53,821 ibmiotf.device.Client INFO Connected successfully: d:puubdh:raspberrypi:demo123
Published Temperature = 85 C Humidity = 80 % Soil Moisture = 57 to IBM Watson
Published Temperature = 41 C Humidity = 80 % Soil Moisture = 80 to IBM Watson
Command received: motor on
motor is on
Published Temperature = 20 C Humidity = 5 % Soil Moisture = 34 to IBM Watson
Command received: motor off
motor is off
Published Temperature = 7 C Humidity = 29 % Soil Moisture = 18 to IBM Watson

10 times
```

Ln: 24 Col: 21

11:44
08-11-2022