

PROJECT DEVELOPMENT PHASE

SPRINT 2

TEAM ID: PNT2022TMID29654

PROJECT NAME: IOT Based Smart Crop Protection System for Agriculture

SOURCE CODE:

```
import random

import ibmiotf.application

import ibmiotf.device

from time import sleep

import sys

#IBM Watson Device Credentials.

organization = "ncgqpp"

deviceType = "raspberrypi"

deviceId = "123"

authMethod = "token"

authToken = "123456789"

def myCommandCallback(cmd):

    print("Command received: %s" % cmd.data['command'])

    status=cmd.data['command']

    if status=="sprinkler_on":

        print ("sprinkler is ON")

    else :
```

```
print ("sprinkler is OFF")
#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()
#Connecting to IBM watson.
deviceCli.connect()
while True:
    #Getting values from sensors.
    temp_sensor = round( random.uniform(0,80),2)
    PH_sensor = round(random.uniform(1,14),3)
    camera = ["Detected","Not Detected","Not Detected","Not
Detected","Not Detected","Not Detected",]
    camera_reading = random.choice(camera)
    flame = ["Detected","Not Detected","Not Detected","Not
Detected","Not Detected","Not Detected",]
    flame_reading = random.choice(flame)
    moist_level = round(random.uniform(0,100),2)
    water_level = round(random.uniform(0,30),2)
    #storing the sensor data to send in json format to cloud.
    temp_data = { 'Temperature' : temp_sensor }
```

```
PH_data = { 'PH Level' : PH_sensor }
camera_data = { 'Animal attack' : camera_reading}
flame_data = { 'Flame' : flame_reading }
moist_data = { 'Moisture Level' : moist_level}
water_data = { 'Water Level' : water_level}

# publishing Sensor data to IBM Watson for every 5-10 seconds.

success = deviceCli.publishEvent("Temperature sensor","json",
temp_data, qos=0)

sleep(1)

if success:

    print (" .....publish ok..... ")

    print ("Published Temperature = %s C" % temp_sensor, "to IBM
Watson")

success = deviceCli.publishEvent("PH sensor", "json",PH_data, qos=0)

sleep(1)

if success:

    print ("Published PH Level = %s" % PH_sensor, "to IBM Watson")

success = deviceCli.publishEvent("camera", "json",camera_data,
qos=0)

sleep(1)

if success:

    print ("Published Animal attack %s " % camera_reading, "to IBM
Watson")

success = deviceCli.publishEvent("Flame sensor", "json",flame_data,
qos=0)
```

```
sleep(1)
if success:
    print ("Published Flame %s " % flame_reading, "to IBM Watson")
    success = deviceCli.publishEvent("Moisture sensor",
    "json",moist_data, qos=0)
    sleep(1)
    if success:
        print ("Published Moisture Level = %s " % moist_level, "to IBM
        Watson")
        success = deviceCli.publishEvent("Water sensor", "json",water_data,
        qos=0)
        sleep(1)
        if success:
            print ("Published Water Level = %s cm" % water_level, "to IBM
            Watson")
            print ("")
            #Automation to control sprinklers by present temperature an to
            send alert message to IBM Watson.
            if (temp_sensor > 35):
                print("sprinkler-1 is ON")
                success = deviceCli.publishEvent("Alert1", "json",{ 'alert1' :
                "Temperature(%s) is high, sprinkerlers are turned ON"
                %temp_sensor }, qos=0)
                sleep(1)
                if success:
```

```
print( 'Published alert1 : ', "Temperature(%s) is high, sprinklers are  
turned ON" %temp_sensor,"to IBM Watson")
```

```
print("")
```

```
else:
```

```
print("sprinkler-1 is OFF")
```

```
print("")
```

```
#To send alert message if farmer uses the unsafe fertilizer to crops.
```

```
if (PH_sensor > 7.5 or PH_sensor < 5.5):
```

```
    success = deviceCli.publishEvent("Alert2", "json",{ 'alert2': "Fertilizer  
PH level(%s) is not safe,use other fertilizer"%PH_sensor },qos=0)
```

```
    sleep(1)
```

```
    if success:
```

```
        print('Published alert2 : ', "Fertilizer PH level(%s) is not safe,use  
other fertilizer" %PH_sensor,"to IBM Watson")
```

```
        print ("")
```

```
# To send alert message to farmer that animal attack on crops.
```

```
if (camera_reading == "Detected"):
```

```
    success = deviceCli.publishEvent("Alert3", "json", {'alert3' : "Animal  
attack on crops detected" }, qos=0)
```

```
    sleep(1)
```

```
    if success:
```

```
        print('Published alert3 : ', "Animal attack on crops detected", "to  
IBM Watson", "to IBM Watson")
```

```
        print("")
```

#To send alert message if flame detected on crop land and turn ON the splinkers to take immediate action.

```
if (flame_reading == "Detected"):
```

```
    print("sprinkler-2 is ON")
```

```
    success = deviceCli.publishEvent("Alert4", "json", { 'alert4': "Flame is detected crops are in danger,sprinklers turned ON"}, qos=0)
```

```
    sleep(1)
```

```
if success:
```

```
    print( 'Published alert4 : ' , "Flame is detected crops are in danger,sprinklers turned ON","to IBM Watson")
```

```
    print("")
```

```
else:
```

```
    print("sprinkler-2 is OFF")
```

```
    print("")
```

#To send alert message if Moisture level is LOW and to Turn ON Motor-1 for irrigation.

```
if (moist_level < 20):
```

```
    print("Motor-1 is ON")
```

```
    success = deviceCli.publishEvent("Alert5", "json", { 'alert5': "Moisture level(%s) is low, Irrigation started" %moist_level}, qos=0)
```

```
    sleep(1)
```

```
if success:
```

```
    print('Published alert5 : ' , "Moisture level(%s) is low, Irrigation started" %moist_level,"to IBM Watson" )
```

```
    print("")
```

else:

```
print("Motor-1 is OFF")
```

```
print("")
```

#To send alert message if Water level is HIGH and to Turn ON Motor-2 to take water out.

```
if (water_level > 20):
```

```
print("Motor-2 is ON")
```

```
success = deviceCli.publishEvent("Alert6", "json", { 'alert6': "Water level(%s) is high, so motor is ON to take water out "%water_level }, qos=0)
```

```
sleep(1)
```

```
if success:
```

```
print('Published alert6 : ' , "water level(%s) is high, so motor is ON to take water out " %water_level,"to IBM Watson" )
```

```
print("")
```

else:

```
print("Motor-2 of OFF")
```

```
print("")
```

#command recived by farmer

```
deviceCli.commandCallback = myCommandCallback
```

Disconnect the device and application from the cloud

```
deviceCli.disconnect()
```

Feature code:

```
def myCommandCallback(cmd):
```

```
    print("Command received: %s" % cmd.data['command'])
```

```
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:
```

```
    temp=random.randint(0,100)
```

```
    pulse=random.randint(0,100)
```

```
    soil=random.randint(0,100)
```

```
    data = { 'temp' : temp, 'pulse': pulse , 'soil':soil}
```

```
    print (data)
```

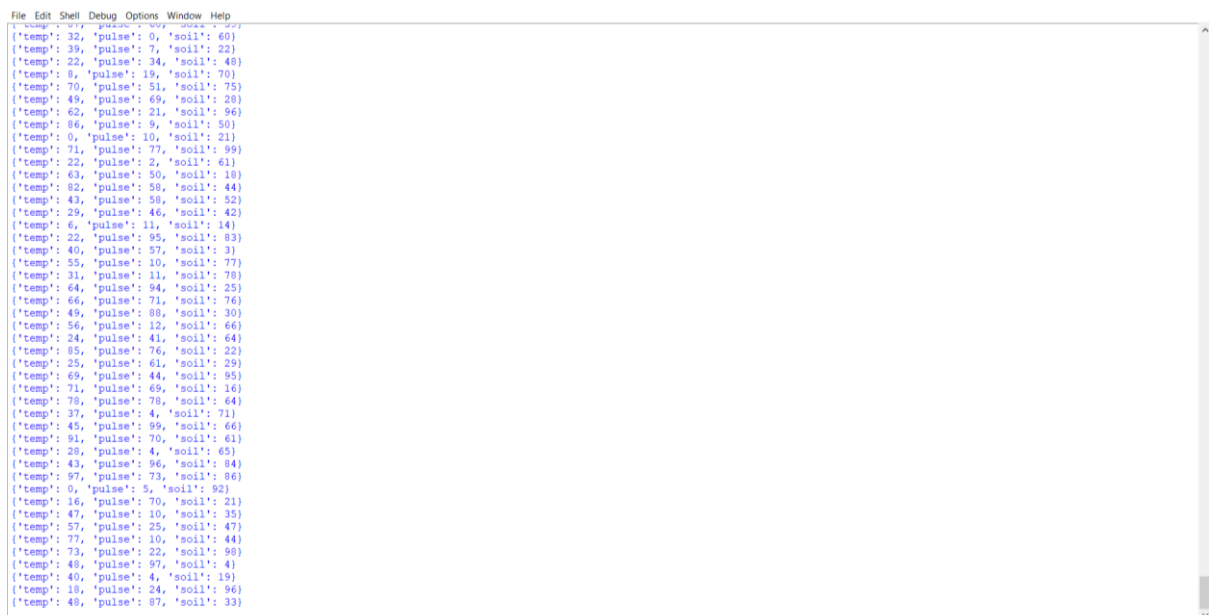


```
def myOnPublishCallback():
    print ("Published Temperature = %s C" % temp, "Humidity = %s
    %% " % pulse, "SoilMoisture = %s %% " % soil, "to IBM Watson")

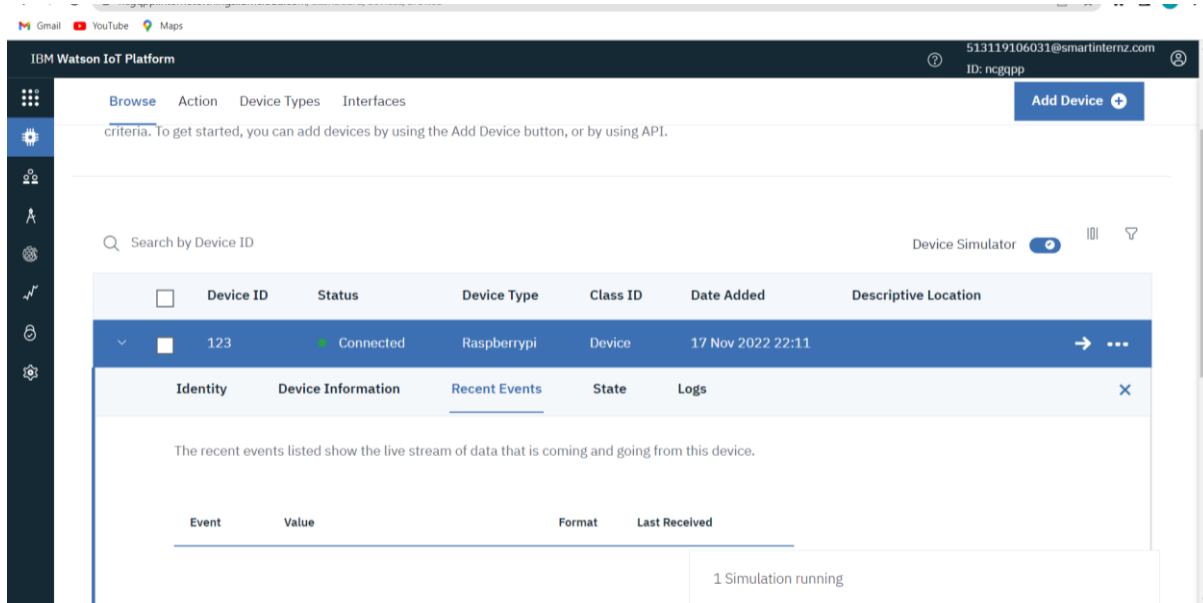
    success = deviceCli.publishEvent("IoTSensor", "json", data,
    qos=0, on_publish=myOnPublishCallback)

if not success:
    print("Not connected to IoTF")
    time.sleep(1)

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



```
File Edit Shell Debug Options Window Help
('temp': 32, 'pulse': 0, 'soil': 60)
('temp': 39, 'pulse': 7, 'soil': 22)
('temp': 22, 'pulse': 34, 'soil': 48)
('temp': 8, 'pulse': 18, 'soil': 70)
('temp': 70, 'pulse': 51, 'soil': 75)
('temp': 49, 'pulse': 69, 'soil': 28)
('temp': 62, 'pulse': 21, 'soil': 96)
('temp': 86, 'pulse': 9, 'soil': 50)
('temp': 0, 'pulse': 10, 'soil': 21)
('temp': 71, 'pulse': 77, 'soil': 99)
('temp': 22, 'pulse': 2, 'soil': 61)
('temp': 63, 'pulse': 50, 'soil': 18)
('temp': 82, 'pulse': 58, 'soil': 44)
('temp': 43, 'pulse': 58, 'soil': 52)
('temp': 29, 'pulse': 46, 'soil': 42)
('temp': 6, 'pulse': 11, 'soil': 14)
('temp': 22, 'pulse': 95, 'soil': 83)
('temp': 40, 'pulse': 57, 'soil': 31)
('temp': 55, 'pulse': 10, 'soil': 77)
('temp': 31, 'pulse': 11, 'soil': 78)
('temp': 64, 'pulse': 94, 'soil': 25)
('temp': 66, 'pulse': 71, 'soil': 76)
('temp': 49, 'pulse': 88, 'soil': 30)
('temp': 56, 'pulse': 12, 'soil': 66)
('temp': 24, 'pulse': 41, 'soil': 64)
('temp': 85, 'pulse': 76, 'soil': 22)
('temp': 25, 'pulse': 61, 'soil': 29)
('temp': 69, 'pulse': 44, 'soil': 95)
('temp': 71, 'pulse': 69, 'soil': 16)
('temp': 78, 'pulse': 78, 'soil': 64)
('temp': 37, 'pulse': 4, 'soil': 71)
('temp': 45, 'pulse': 99, 'soil': 66)
('temp': 91, 'pulse': 70, 'soil': 61)
('temp': 28, 'pulse': 4, 'soil': 65)
('temp': 43, 'pulse': 96, 'soil': 84)
('temp': 97, 'pulse': 73, 'soil': 66)
('temp': 0, 'pulse': 5, 'soil': 92)
('temp': 16, 'pulse': 70, 'soil': 21)
('temp': 47, 'pulse': 10, 'soil': 35)
('temp': 57, 'pulse': 25, 'soil': 47)
('temp': 77, 'pulse': 10, 'soil': 44)
('temp': 73, 'pulse': 22, 'soil': 98)
('temp': 48, 'pulse': 97, 'soil': 4)
('temp': 40, 'pulse': 4, 'soil': 19)
('temp': 18, 'pulse': 24, 'soil': 96)
('temp': 48, 'pulse': 87, 'soil': 33)
```



Feature code:

```
def myCommandCallback(cmd):
```

```
    if cmd.data['command'] == 'motoron':
```

```
        print("MOTOR ON IS RECEIVED")
```

```
    elif cmd.data['command'] == 'motoroff':
```

```
        print("MOTOR OFF IS RECEIVED")
```

```
    if cmd.command == "setInterval":
```

```
        if 'interval' not in cmd.data:
```

```
            print("Error - command is missing required information:
'interval'")
```

```
        else:
```

```
            interval = cmd.data['interval']
```

```

elif cmd.command == "print":

    if 'message' not in cmd.data:

        print("Error - command is missing required information:
'message'")

    else:

        output = cmd.data['message']

        print(output)

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:

```

```
deviceCli.commandCallback = myCommandCallback
```

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

```
deviceCli.disconnect()
```

```
File Edit Shell Debug Options Window Help
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\dhosh\AppData\Local\Programs\Python\Python37\motor.py ==
2022-11-17 23:37:29,315 ibmiotf.device.Client INFO Connected successfully: d:ncgqpp:Raspberrypi:123
```

