

FINAL CODE

Code 1:

- ✓ To view temperature, oxygen level and humidity

```
import time
```

```
import sys
```

```
import ibmiotf.application
```

```
import ibmiotf.device
```

```
import random
```

```
#Provide your IBM Watson Device Credentials
```

```
organization = "nym2po"
```

```
deviceType = "b5ibm"
```

```
deviceId = "b5device"
```

```
authMethod = "token"
```

```
authToken = "b55m1eibm"
```

```
#Initialize GPIO
```

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

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

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

```
lat= 17
```

```
lon=18
```

```
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:

    #Get Sensor Data from DHT11

    data = {"d":{"temp": temp, 'pulse': pulse, 'oxygen': oxygen,"lat":lat,"lon":lon}}

    #print data

    def myonPublishCallback():

        print ("Published Temperature = %s C" % temp, "Humidity = %s %" % pulse, "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()

```

CODE 2 :

- ✓ To control sprinklers by present temperature and to send alert message to IBM Watson
- ✓ To send alert message if farmer uses the unsafe fertilizer to crops or animal attack is detected
- ✓ To send alert message if Moisture level is LOW and to Turn ON Motor-1 for irrigation.
- ✓ To send alert message if Water level is HIGH and to Turn ON Motor-2 to take water out.

```
import random
import ibmiotf.application
import ibmiotf.device

from time import sleep
import sys

#IBM Watson Device Credentials.
organization = "nym2po "
deviceType = "b5ibm"
deviceId = " b5device "
authMethod = "token"
authToken = "b55m1eibm"

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 and 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,
sprinklerlers are turned ON" %temp_sensor }
, qos=0)

    sleep(1)

    if success:

        print( 'Published alert1 : ', "Temperature(%s) is high, sprinklerlers 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")

#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("")

#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("")  
  
#command recived by farmer  
deviceCli.commandCallback = myCommandCallback  
# Disconnect the device and application from the cloud  
deviceCli.disconnect()
```


Code 3:

- ✓ Connect the device and application to cloud

Provide your IBM Watson Device Credentials

```
def myCommandCallback(cmd): # function for Callback if cm.data['command'] == 'motoron':
```

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

```
elif cmd.data['command'] == 'motoroff': print("MOTOR OFF IS RECEIVED")
```

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

```
else:
```

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

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

```
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, "authmethod":  
authMethod,
```

```

"auth-token": authToken}          deviceCli
= ibmiotf.device.Client(deviceOptions) # .....

exceptException 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()

```

SENSOR.PY

```

import time import sysimport ibmiotf.application importibmiotf.device
import random

# Provide your IBM Watson Device Credentials
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)
#.....

exceptException 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,"Soil Moisture =  
%s %" % soil,"to IBM Watson")
```

```
success = deviceCli.publishEvent("IoTSensor", "json", data, qos=0,  
on_publish=myOnPublishCallback)          if not success:
```

```
print("Not connected to IoT") time.sleep(1)
```

```
deviceCli.commandCallback = myCommandCallback
```