```
import random

import ibmiotf.application

import ibmiotf.device

from time import sleep

import sys

#IBM Watson Device Credentials.

organization = "66gns9"

deviceType = "abcd"

deviceId = "123"

authMethod = "token"

authToken = "23456789"

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, sprinkerlers are
turned ON" %temp_sensor }

, qos=0)

 sleep(1)

 if success:

    print( 'Published alert1 : ', "Temperature(%s) is high, sprinkerlers 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 received at farmer end**

```
deviceCli.commandCallback = myCommandCallback
```

# Disconnecting the device and application from the cloud

```
deviceCli.disconnect()import random

import ibmiotf.application

import ibmiotf.device

from time import sleep

import sys
```

#IBM Watson Device Credentials.

```
organization = "66gns9"

deviceType = "abcd"

deviceId = "123"

authMethod = "token"

authToken = "23456789"

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, sprinkerlers are
turned ON" %temp_sensor }
, qos=0)
 sleep(1)
 if success:
    print( 'Published alert1 : ', "Temperature(%s) is high, sprinkerlers are turned ON" %temp_sensor,"to
IBM Watson")
 print("")
else:
 print("sprinkler-1 is OFF")
 print("")
# Sending 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("")
 # Sending 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("")

 # Sending 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")

 #Sending 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("")
 #Sending 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 received by farmer

deviceCli.commandCallback = myCommandCallback

# Disconnect the device and application from the cloud

deviceCli.disconnect()
```