Develop a python script

Team ID	PNT2022TMID30928
Project Title	Iot Based Smart Crop Protection System for Agriculture

PYTHON CODE:

```
import cv2

import numpy as np

import wiot.sdk.device

import playsound

import random

import time

import datetime

import ibm_boto3

from ibm_botocore.client import Config, ClientError


#CloudantDB

from cloudant.client import Cloudant

from cloudant.error import CloudantException

from cloudant.result import Result, ResultByKey
from clarifai_grpc.channel.clarifai_channel import ClarifaiChannel
from clarifai_grpc.grpc.api import service_pb2_grpc
stub = service_pb2_grpc.V2Stub(clarifaiChannel.get.grpc_channel())
from clarifai_grpc.grpc.api import service_pb2, resource_pb2
from clarifai_grpc.grpc.api.status import status_code_pb2

#This is how you authenticate
metadata = (('authorization', 'key 0620e202302b4508b90eab7efe7475e4'),)
COS_ENDPOINT="https://s3.jp-tok.cloud-object-storage.appdomain.cloud"
COS_API_KEY_ID="g5d4qO8EIgv4TWUCJj4hfEzgaIqEjrDbE82AJDWIAOHo"
COS_AUTH_ENDPOINT = "https://iam.cloud.ibm.com/identity/token"
COS_RESOURCE_CRN="crn:v1:bluemix:public:cloud-object-storage:global:a/c2fa2836eaf3434bbc8b5b58fefff3f0:62e450fd-4c82-4153-ba41-ccb53adb8111::"
clientdb=cloudant("apikey-W2njldnwtjO16V53LAVUCqPwc2aHTLmIj1xXvtdGKJBn","88cc5f47c1a28afbfb8ad16161583f5a",url="https://d6c89f97-cf91-48b7-b14b-c99b2fe27c2f-bluemix.cloudantnosqldb.appdomain.cloud")
clientdb.connect()
```

```
#Create resource
cos = ibm_boto3.resource("s3",
                ibm_api_key_id=COS_API_KEY_ID,
                ibm_service_instance_id=COS_RESOURCE_CRN,
                ibm_auth_endpoint=COS_AUTH_ENDPOINT,
                config=Config(signature_version="oauth"),
                endpoint_url=COS_ENDPOINT
                )
def = multi_part_upload(bucket_name, item_name, file_path):
    try:
        print("Starting file transfer for {0} to bucket:
{1}\n".format(item_name, bucket_name))
        #set 5 MB chunks
        part_size = 1024 * 1024 * 5
        #set threadhold to 15 MB
        file_threshold = 1024 * 1024 * 15
        #set the transfer threshold and chunk size
        transfer_config = ibm_boto3.s3.transfer.TransferConfig(
            multipart_threshold=file_threshold,
            multipart_chunksize=part_size
            )
        #the upload_fileobj method will automatically execute a multi-part
upload
        #in 5 MB chunks size
        with open(file_path, "rb") as file_data:
            cos.Object(bucket_name, item_name).upload_fileobj(
                Fileobj=file_data,
                Config=transfer_config
                )
        print("Transfer for {0} Complete!\n".format(item_name))
    except ClientError as be:
        print("CLIENT ERROR: {0}\n".format(be))
    except Exception as e:
        print("Unable to complete multi-part upload: {0}".format(e))

def myCommandCallback(cmd):
    print("Command received: %s" % cmd.data)
    command=cmd.data['command']
    print(command)
    if(commamd=="lighton"):
        print('lighton')
    elif(command=="lightoff"):
        print('lightoff')
    elif(command=="motoron"):
        print('motoron')
    elif(command=="motoroff"):
        print('motoroff')
myConfig = {
    "identity": {
        "orgId": "fzb72x",
        "typeId": "ESP-",
        "deviceId": "1234567890"
        },
    "auth": {
        "token": "pByAf4p(2nTbtBIMQM"
        }
    }
client = wiot.sdk.device.DeviceClient(config=myConfig,
logHandlers=None)
```

```
client.connect()


database_name = "sample"

my_database = clientdb.create_database(database_name)

if my_dtabase.exists():

    print(f"'(database_name)' successfully created.")

cap=cv2.VideoCapture("garden.mp4")

if(cap.isOpened()==True):

    print('File opened')

else:

    print('File not found')


while(cap.isOpened()):

    ret, frame = cap.read()

    gray = cv3.cvtColor(frame, cv2.COLOR_BGR@GRAY)

    imS= cv2.resize(frame, (960,540))

    cv2.inwrite('ex.jpg',imS)

    with open("ex.jpg", "rb") as f:

        file_bytes = f.read()
    #This is the model ID of a publicly available General model. You
may use any other public or custom model ID.
    request = service_pb2.PostModeloutputsRequest(
        model_id='e9359dbe6ee44dbc8842ebe97247b201',

inputs=[resources_pb2.Input(data=resources_pb2.Data(image=resour
ces_pb2.Image(base64=file_bytes))
                                    )])
    response = stub.PostModelOutputs(request, metadata=metadata)
    if response.status.code != status_code_pb2.SUCCESS:
        raise Exception("Request failed, status code: " +
str(response.status.code))
    detect=False
    for concept in response.outputs[0].data.concepts:
        #print('%12s: %.f' % (concept.name, concept.value))
        if(concept.value>0.98):
            #print(concept.name)
            if(concept.name=="animal"):
                print("Alert! Alert! animal detected")
                playsound.playsound('alert.mp3')
```

```
picname=datetime.datetime.now().strftime("%y-%m-%d-%H-%M")
                cv2.inwrite(picname+'.jpg',frame)
                multi_part_upload('Dhakshesh', picname+'.jpg',
picname+'.jpg')

json_document={"link":COS_ENDPOINT+'/'+'Dhakshesh'+'/'+picname
+'.jpg'}
                new_document =
my_database.create_document(json_document)
                if new_document.exists():
                    print(f"Document successfully created.")
                time.sleep(5)
                detect=True
    moist=random.randint(0,100)
    humidity=random.randint(0,100)
    myData={'Animal':detect,'moisture':moist,'humidity':humidity}
    print(myData)
    if(humidity!=None):
        client.publishEvent(eventId="status",msgFormat="json",
daya=myData, qos=0, onPublish=None)
```

```
        print("Publish Ok..")
    client.commandCallback = myCommandCallback
    cv2.imshow('frame',imS)
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break
client.disconnect()
cap.release()
cv2.destroyAllWindows()
```

```
import random

import ibmiotf.application

import ibmiotf.device

from time import sleep

import sys

#IBM Watson Device Credentials.

organization = "fzb72x"

deviceType = "ESP-"

deviceId = "1234567890"

authMethod = "token"

authToken = "pByAf4p(2nTbtBlMQM"

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