**PYTHON CODE**

|  |  |
| --- | --- |
| TEAM ID | PNT2022TMID50747 |
| PROJECT | Smart Waste Management System For Metropolitan Cities |

**1)PYTHON CODE**

#IBM Watson IOT Platform

#pip install wiotp-sdk

import wiotp.sdk.device

import time

import random

myConfig = {

"identity": {

"orgId": "yy5t0v",

"typeId": "lmnopq",

"deviceId":"234567"

},

"auth": {

"token": "87654321"

}

}

def myCommandCallback(cmd):

print("Message received from IBM IoT Platform: %s" % cmd.data['command'])

m=cmd.data['command']

client = wiotp.sdk.device.DeviceClient(config=myConfig, logHandlers=None)

client.connect()

while True:

latitude=random.uniform(27.2046,125.25)

longitude=random.uniform(77.4977,100.1526)

binlevel=random.randint(10,100)

if binlevel >= 90:

myData={'latitude':latitude, 'longitude':longitude,'binlevel':binlevel}

client.publishEvent(eventId="status", msgFormat="json", data=myData, qos=0,onPublish=None)

##print("Published data Successfully: %s", myData)

print("!!!!!!!!!BIN IS FULL!!!!!!!!!! ",myData)

client.commandCallback = myCommandCallback

time.sleep(2)

else :

print("BIN IS IN NORMAL LEVEL")

time.sleep(2)

client.disconnect()

**2)PYTHON CODE**

import requests

import json

import ibmiotf.application

import ibmiotf.device

import time

import random

import sys

# watson device details

organization = "yy5t0v"

devicType = "lmnopq"

deviceId = "234567"

authMethod= "token"

authToken= "87654321"

#generate random values for randomo variables for distance and loadcell

def myCommandCallback(cmd):

global a

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

control=cmd.data['command']

print(control)

try:

deviceOptions={"org": organization, "type": devicType,"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 "distance and loadcell" with value integer value into the cloud as a type of event for every 10 seconds

deviceCli.connect()

while True:

distance= random.randint(10,70)

loadcell= random.randint(5,15)

data= {'dist':distance,'load':loadcell}

if loadcell < 13 and loadcell > 15:

load = "90 %"

elif loadcell < 8 and loadcell > 12:

load = "60 %"

elif loadcell < 4 and loadcell > 7:

load = "40 %"

else:

load = "0 %"

if distance < 15:

dist = 'Risk warning:' 'Dumpster poundage getting high, Time to collect :) 90 %'

elif distance < 40 and distance >16:

dist = 'Risk warning:' 'dumpster is above 60%'

elif distance < 60 and distance > 41:

dist = 'Risk warning:' '40 %'

else:

dist = 'Risk warning:' '17 %'

if load == "90 %" or distance == "90 %":

warn = 'alert :' 'Risk Warning: Dumpster poundage getting high, Time to collect :)'

elif load == "60 %" or distance == "60 %":

warn = 'alert :' 'dumpster is above 60%'

else :

warn = 'alert :' 'No need to collect right now '

def myOnPublishCallback(lat=8.7139,long= 77.7567):

print("Tirunelveli")

print("published distance = %s " %distance,"loadcell:%s " %loadcell,"lon = %s " %long,"lat = %s" %lat)

print(load)

print(dist)

print(warn)

time.sleep(10)

success=deviceCli.publishEvent ("IoTSensor","json",warn,qos=0,on\_publish= myOnPublishCallback)

success=deviceCli.publishEvent ("IoTSensor","json",data,qos=0,on\_publish= myOnPublishCallback)

if not success:

print("not connected to ibmiot")

time.sleep(10)

deviceCli.commandCallback=myCommandCallback

#disconnect the device

deviceCli.disconnect()