import time

import sys

import ibmiotf.application

import ibmiotf.device

import random

# Provide your IBM Watson Device Credentials

organization = "c1n0yk"

deviceType = "Hazard"

deviceId = "2"

authMethod = "token"

authToken = "123456789"

# Initialize GPIO

def myCommandCallback(cmd):

print(cmd)

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

status = cmd.data['command']

if status == "lighton":

print("led is on")

elif status == "lightoff":

print("led is off")

else:

print("please send proper command")

try:

deviceOptions = {"org": organization, "type": deviceType, "id": deviceId, "auth-method": authMethod,

"auth-token": authToken}

deviceCli = ibmiotf.device.Client(deviceOptions)

# ..............................................

except ibmiotf.ConnectionException as e:

print("Caught exception connecting device: %s" % str(e))

sys.exit()

deviceCli.connect()

while True:

# Get Sensor Data from DHT11

temp = random.randint(50, 100)

mydata = {'temp': temp}

def on\_publish():

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

success = deviceCli.publishEvent("Temp sensor", "json", mydata, qos=0, on\_publish=on\_publish)

if not success:

print("Not connected to IoTF")

deviceCli.commandCallback = myCommandCallback

time.sleep(5)

# Disconnect the device and application from the cloud

deviceCli.disconnect()import time

import sys

import ibmiotf.application

import ibmiotf.device

import random

# Provide your IBM Watson Device Credentials

organization = "c1n0yk"

deviceType = "Hazard"

deviceId = "2"

authMethod = "token"

authToken = "123456789"

# Initialize GPIO

def myCommandCallback(cmd):

print(cmd)

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

status = cmd.data['command']

if status == "lighton":

print("led is on")

elif status == "lightoff":

print("led is off")

else:

print("please send proper command")

try:

deviceOptions = {"org": organization, "type": deviceType, "id": deviceId, "auth-method": authMethod,

"auth-token": authToken}

deviceCli = ibmiotf.device.Client(deviceOptions)

# ..............................................

except ibmiotf.ConnectionException as e:

print("Caught exception connecting device: %s" % str(e))

sys.exit()

deviceCli.connect()

while True:

# Get Sensor Data from DHT11

temp = random.randint(50, 100)

mydata = {'temp': temp}

def on\_publish():

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

success = deviceCli.publishEvent("Temp sensor", "json", mydata, qos=0, on\_publish=on\_publish)

if not success:

print("Not connected to IoTF")

deviceCli.commandCallback = myCommandCallback

time.sleep(5)

# Disconnect the device and application from the cloud

deviceCli.disconnect()import time

import sys

import ibmiotf.application

import ibmiotf.device

import random

# Provide your IBM Watson Device Credentials

organization = "c1n0yk"

deviceType = "Hazard"

deviceId = "2"

authMethod = "token"

authToken = "123456789"

# Initialize GPIO

def myCommandCallback(cmd):

print(cmd)

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

status = cmd.data['command']

if status == "lighton":

print("led is on")

elif status == "lightoff":

print("led is off")

else:

print("please send proper command")

try:

deviceOptions = {"org": organization, "type": deviceType, "id": deviceId, "auth-method": authMethod,

"auth-token": authToken}

deviceCli = ibmiotf.device.Client(deviceOptions)

# ..............................................

except ibmiotf.ConnectionException as e:

print("Caught exception connecting device: %s" % str(e))

sys.exit()

deviceCli.connect()

while True:

# Get Sensor Data from DHT11

temp = random.randint(50, 100)

mydata = {'temp': temp}

def on\_publish():

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

success = deviceCli.publishEvent("Temp sensor", "json", mydata, qos=0, on\_publish=on\_publish)

if not success:

print("Not connected to IoTF")

deviceCli.commandCallback = myCommandCallback

time.sleep(5)

# Disconnect the device and application from the cloud

deviceCli.disconnect()