**Code:**

import ibmiotf.application

import ibmiotf.device

import random

import json

import time

#Provide your IBM Watson Device Credentials

organization = "frtx4v"

deviceType = "iotdevice"

deviceId = "1001"

authMethod = "token"

authToken = "1234567890"

# Initialize the device client.

T=0

H=0

def myCommandCallback(cmd):

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

if cmd.data['command']=='lighton':

print("LIGHT ON IS RECEIVED")

elif cmd.data['command']=='lightoff':

print("LIGHT OFF IS RECEIVED")

if cmd.command == "setInterval":

if 'interval' not in cmd.data:

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

else:

interval = cmd.data['interval']

elif cmd.command == "print":

if 'message' not in cmd.data:

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

else:

print(cmd.data['message'])

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:

T=23

H=45

#Send Temperature & Humidity to IBM Watson

data = {"d":{ 'temperature' : T, 'humidity': H }}

#print (data)

def myOnPublishCallback():

print ("Published Temperature = %s C" % T, "Humidity = %s %%" % H, "to IBM Watson")

success = deviceCli.publishEvent("Data", "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()

**Node Red:**



**MIT APP INVENTOR:**

****

****

**Python Output:**

****