

Assignment 3

(Keerti Srivastava)

Develop a code to upload the water tank level and light intensity values to the IBM IoT platform and visualize them in the web application.

Code:

```
import wiotp.sdk.device

import time

import random

myConfig = {
    "identity": {
        "orgId": "j1cf0v",
        "typeId": "VITdevice",
        "deviceId": "12345"
    },
    "auth": {
        "token": "1234567890"
    }
}

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:
    water_level=random.randint(0,125)
```

```
light_intensity=random.randint(0,100)

myData={'water_tank_level':water_level, 'intensity':light_intensity}

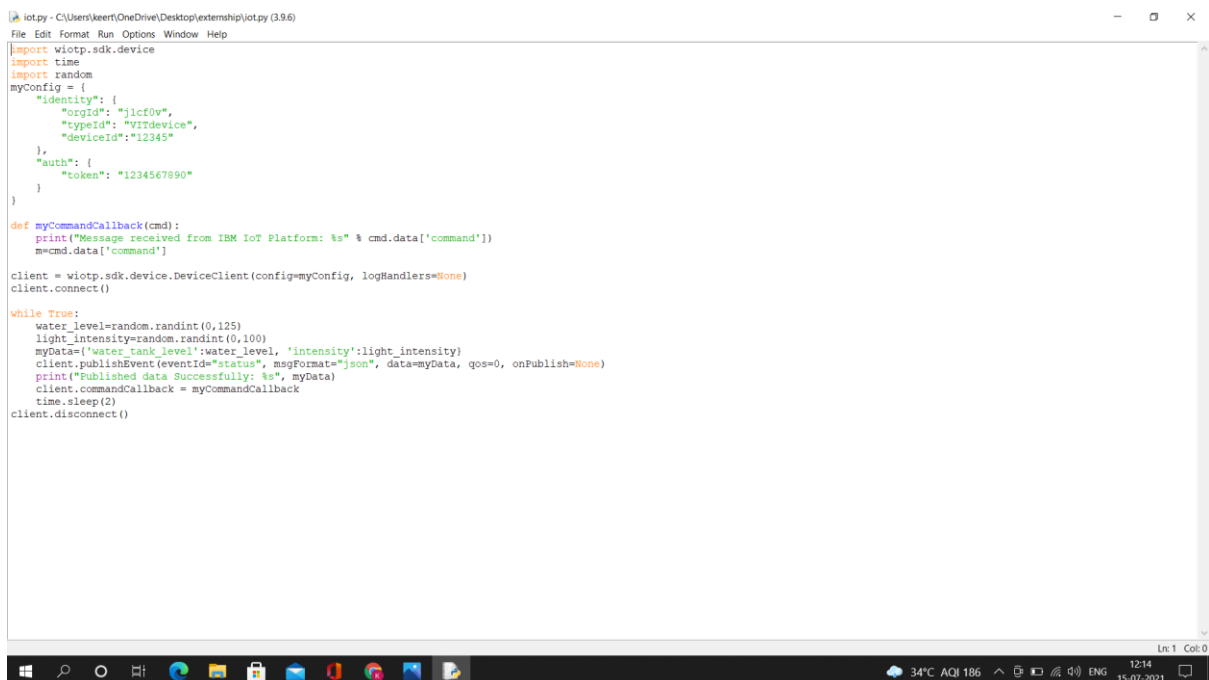
client.publishEvent(eventId="status", msgFormat="json", data=myData, qos=0,
onPublish=None)

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

client.commandCallback = myCommandCallback

time.sleep(2)

client.disconnect()
```

A screenshot of a Windows desktop showing a code editor window titled 'iot.py - C:\Users\keert\OneDrive\Desktop\extenship\iot.py (3.9.6)'. The code in the editor is a Python script for an IoT device. It imports 'wiotp.sdk.device', 'time', and 'random'. It defines a 'myConfig' dictionary with 'identity' (orgId, typeId, deviceId) and 'auth' (token). It defines a 'myCommandCallback' function that prints incoming commands. It creates a 'DeviceClient' and connects it. A 'while True' loop generates random 'water_level' and 'light_intensity' values, creates a 'myData' dictionary, publishes a 'status' event in JSON format, prints a success message, sets a command callback, sleeps for 2 seconds, and then disconnects. The Windows taskbar at the bottom shows the Start button, search, and several application icons. The system tray on the right shows the date and time as '12:14 15-07-2021'.

Node-RED Use Case, Integrating x Student Dashboard x Service Details - IBM Cloud x IBM Watson IoT Platform x Node-RED : node-red-gdk x Node-RED Dashboard x

node-red-gdkm-2021-07-08.eu-gb.mybluemix.net/#flow/4ce8af2.84e255

IBM Watson IoT Platform

keerti.srivastava2019@vitstudent.ac.in
ID: j1cd0v

Browse Action Device Types Interfaces

This table shows a summary of all devices that have been added. It can be filtered, organized, and searched on using different criteria. To get started, you can add devices by using the Add Device button, or by using API.

Search by Device ID Device Simulator

Device ID	Status	Device Type	Class ID	Date Added	Descriptive Location
12345	Connected	VITdevice	Device	8 Jul 2021 18:52	

Identity Device Information Recent Events State Logs

Device ID: 12345
Device Type: VITdevice
Date Added: 8 Jul 2021 18:52
Added By: keerti.srivastava2019@vitstudent.ac.in
Connection Status: Connected
Connection Time: 15 Jul 2021 12:15
Client Address: 103.226.226.224 SecureToken

Cookie Preferences

34°C AQI 186 12:18 15-07-2021

Node-RED Use Case, Integrating x Student Dashboard x Service Details - IBM Cloud x IBM Watson IoT Platform x Node-RED : node-red-gdkm-20 x Node-RED Dashboard x

node-red-gdkm-2021-07-08.eu-gb.mybluemix.net/#flow/4ce8af2.84e255

Node-RED

Flow 1

filter nodes

dashboard

- button
- dropdown
- switch
- slider
- numeric
- text input
- date picker
- colour picker
- form
- text
- gauge
- chart
- audio out
- notification

msg.payload

Tank level

Light intensity

Water Tank Level

Light Intensity

debug

all nodes

Deploy

34°C AQI 186 12:14 15-07-2021

Node-RED interface showing a Python script for IoT device communication. The script uses the `wiotsdk` library to connect to an IBM IoT device and publish data to a message payload.

```
import wiotsdk.device
import time
import random

myConfig = {
    "identity": {
        "orgId": "j1cfo9",
        "typeId": "VITdevice",
        "deviceId": "12345"
    },
    "auth": {
        "token": "1234567890"
    }
}

def myCommandCallback(cmd):
    print("Message received from IBM IoT P")
    m=cmd.data['command']

client = wiotsdk.device.DeviceClient(con
client.connect()

while True:
    water_level=random.randint(0,125)
    light_intensity=random.randint(0,100)
    myData={'water_tank_level':water_level
    client.publishEvent(eventId="status",
    print("Published data Successfully: %s"
    client.commandCallback = myCommandCall
    time.sleep(2)
    client.disconnect()
```

The terminal output shows the device connecting successfully and publishing data:

```
Python 3.9.6 (tags/v3.9.6:db3ff76, Jun 28 2021, 15:26:21) [MSC v.1929 64 bit (AM
64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: C:\Users\keert\OneDrive\Desktop\externship\iot.py =====
2021-07-15 12:15:10,674 wiotsdk.device.client.DeviceClient INFO Connecte
d successfully: d1j1cfo9:VITdevice:12345
Published data Successfully: %s ('water_tank_level': 93, 'intensity': 99)
Published data Successfully: %s ('water_tank_level': 80, 'intensity': 93)
Published data Successfully: %s ('water_tank_level': 65, 'intensity': 54)
Published data Successfully: %s ('water_tank_level': 73, 'intensity': 93)
Published data Successfully: %s ('water_tank_level': 86, 'intensity': 91)
```

The debug console shows the received messages:

```
7/15/2021, 12:15:17 PM node: ftsa5abfe 698648
iot-2/type/VITdeviceId/12345/ev/status/fmt/json :
msg.payload : Object
> { water_tank_level: 73, intensity:
93 }
7/15/2021, 12:15:17 PM node: ftsa5abfe 698648
iot-2/type/VITdeviceId/12345/ev/status/fmt/json :
msg.payload : number
73
7/15/2021, 12:15:17 PM node: ftsa5abfe 698648
iot-2/type/VITdeviceId/12345/ev/status/fmt/json :
msg.payload : number
93
7/15/2021, 12:15:19 PM node: ftsa5abfe 698648
iot-2/type/VITdeviceId/12345/ev/status/fmt/json :
msg.payload : Object
> { water_tank_level: 86, intensity:
91 }
7/15/2021, 12:15:19 PM node: ftsa5abfe 698648
iot-2/type/VITdeviceId/12345/ev/status/fmt/json :
msg.payload : number
86
7/15/2021, 12:15:19 PM node: ftsa5abfe 698648
iot-2/type/VITdeviceId/12345/ev/status/fmt/json :
msg.payload : number
91
```

Node-RED interface showing a flow diagram for IoT data processing. The flow starts with an `IBM IoT` node, which connects to a `msg.payload` node. The data is then processed by `Tank level` and `Light intensity` nodes, which output to `Water Tank Level` and `Light Intensity` nodes respectively.

The debug console shows the received messages:

```
89
7/15/2021, 12:15:27 PM node: ftsa5abfe 698648
iot-2/type/VITdeviceId/12345/ev/status/fmt/json :
msg.payload : Object
> { water_tank_level: 36, intensity:
60 }
7/15/2021, 12:15:27 PM node: ftsa5abfe 698648
iot-2/type/VITdeviceId/12345/ev/status/fmt/json :
msg.payload : number
36
7/15/2021, 12:15:27 PM node: ftsa5abfe 698648
iot-2/type/VITdeviceId/12345/ev/status/fmt/json :
msg.payload : number
60
7/15/2021, 12:15:29 PM node: ftsa5abfe 698648
iot-2/type/VITdeviceId/12345/ev/status/fmt/json :
msg.payload : Object
> { water_tank_level: 105, intensity:
33 }
7/15/2021, 12:15:29 PM node: ftsa5abfe 698648
iot-2/type/VITdeviceId/12345/ev/status/fmt/json :
msg.payload : number
105
7/15/2021, 12:15:29 PM node: ftsa5abfe 698648
iot-2/type/VITdeviceId/12345/ev/status/fmt/json :
msg.payload : number
33
```

