

Project Development Phase

Sprint Delivery – II

PROJECT TITLE	Gas Leakage Monitoring and Alerting System
TEAM ID	PNT2022TMID06977

Introduction:

In this sprint delivery - II, we have completed a python code for the random data generation and connecting with IBM Watson IoT platform for publishing sensor data and the current Valve State in accordance with MQTT protocol. We visualized the published sensor data by the python code using the Card options in the IBM Watson IoT Platform Boards.

USER STORY/TASK 6: Random Data Generation

We used the random package in python to generate random data for the sake of data from the actual sensor in the industry or the location where the IOT kit is placed. We also used 5 Global Variables namely Valve1, Valve2, Valve3, Valve4, Valve5 to store the current state of the valves. We are generating five pieces of random data as we are using five gas sensors at five different locations.

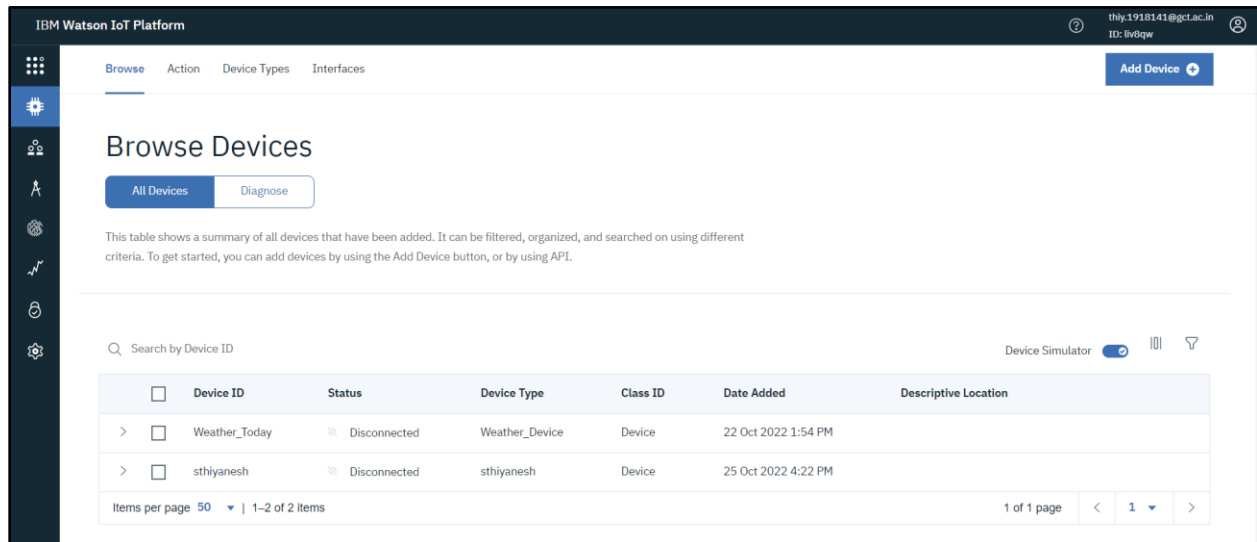
Random Data generation in Python Code:

```
Sensor1= random.randint(0,500)
```

USER STORY/TASK 7: Create Device in IBM Watson IoT Platform

Open IBM Watson IoT platform and login with your IBM Cloud login credentials. Click your profile and start your free Bluemix ID session. Create a device in IBM Watson IoT platform and create credentials for the device and store the credentials for making connection with this device by the python code in the later stage of the project.

Device Created:



Device ID	Status	Device Type	Class ID	Date Added	Descriptive Location
Weather_Today	Disconnected	Weather_Device	Device	22 Oct 2022 1:54 PM	
sthiyanesh	Disconnected	sthiyanesh	Device	25 Oct 2022 4:22 PM	

Devices created in IBM Watson IoT Platform

USER STORY/TASK 8: Create Connection with IBM Watson IoT platform

Store the IBM Watson Credentials as variables such as organization, deviceType, device ID, authentication method and authentication token. Store these device credentials as a json format and call `ibmiotf.device.Client(deviceOptions)` function and `deviceCli.connect()` function to create connection between python code and IBM watson IoT Platform.

Python Code for Connection Establishment:

#Provide your IBM Watson Device Credentials

```
organization = "liv8qw"
```

```
deviceType = "sthiyanesh"
```

```
deviceId = "sthiyanesh"
```

```
authMethod = "token"
```

```
authToken = "sthiyanesh"
```

```
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()
```

USER STORY/TASK 9: Publish and Subscribe Data to IBM Watson IoT platform

Convert the Randomly generated Sensor Data and Current State of the Valve into a publishable Json Format and publish the Data using deviceCli.publishEvent() Function and Call the deviceCli.commandCallback = myCommandCallback to continuously listen for the subscribed data which gives a command to change the current Valve state. Once a Subscribed Topic data is published check for the valve Number and State in the Command Message and use if-condition and change the current Valve state of the valve.

USER STORY/TASK 10: Create Cards in IBM Watson IoT Platform Boards:

Use the create board option in IBM Watson IoT Platform and name your board as Gas Sensor data visualization and give some description about it. Open your board and create a card and select your device then give the details of the event and data in the message of the event published and select the type of visualization and save your card. Now you can Visualize the data from the python code in your IBM Watson IoT platform card.

Final Python Code of Sprint 2:

```
import time
import sys
import ibmiotf.application
import ibmiotf.device
import random

#Provide your IBM Watson Device Credentials
organization = "liv8qw"
deviceType = "sthiyanesh"
deviceId = "sthiyanesh"
authMethod = "token"
authToken = "sthiyanesh"
valve1="Opened";
valve2="Opened";
valve3="Opened";
valve4="Closed";
valve5="Opened";
# Initialize GPIO

def myCommandCallback(cmd):
    global valve1, valve2, valve3, valve4, valve5;
    #print("Command received: %s" % cmd.data['valve1'])
    ValveNumber=cmd.data['Valve']
    ValveState=cmd.data['State']
    if(((ValveNumber==1)|(ValveNumber=="1"))):
        valve1=ValveState
        print(valve1)
    if((ValveNumber==2)|(ValveNumber=="2")):
        valve2=ValveState
```

```

if((ValveNumber==3)|(ValveNumber=="3")):
    valve3=ValveState
if((ValveNumber==4)|(ValveNumber=="4")):
    valve4=ValveState
if((ValveNumber==5)|(ValveNumber=="5")):
    valve5=ValveState
print("Valve "+str(ValveNumber)+" is "+ValveState)

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:
    #Randomly generated Sensor Data
    Sensor1= random.randint(0,500)
    Sensor2= random.randint(0,500)
    Sensor3= random.randint(0,500)
    Sensor4= random.randint(0,500)
    Sensor5= random.randint(0,500)
    data = {"SensorData":{"Sensor1": Sensor1, 'Sensor2': Sensor2, 'Sensor3':
Sensor3, "Sensor4":Sensor4, "Sensor5":Sensor5}, "ValveData":{ "Valve1" : { "Valve": 1, "State": valve1
}, "Valve2" : { "Valve": 2, "State": valve2 }, "Valve3" : { "Valve": 3, "State": valve3 }, 'Valve4' : { "Valve": 4,
"State": valve4 }, 'Valve5' : { "Valve": 5, "State": valve5 } } }
    #print data
    def myOnPublishCallback():
        print ("Published ",data, "to pyIBM Watson")
    success = deviceCli.publishEvent("IoTSensor", "json", data, qos=0, on_publish=myOnPublishCallback)
    if not success:
        print("Not connected to IoT")
    time.sleep(1)

    deviceCli.commandCallback = myCommandCallback

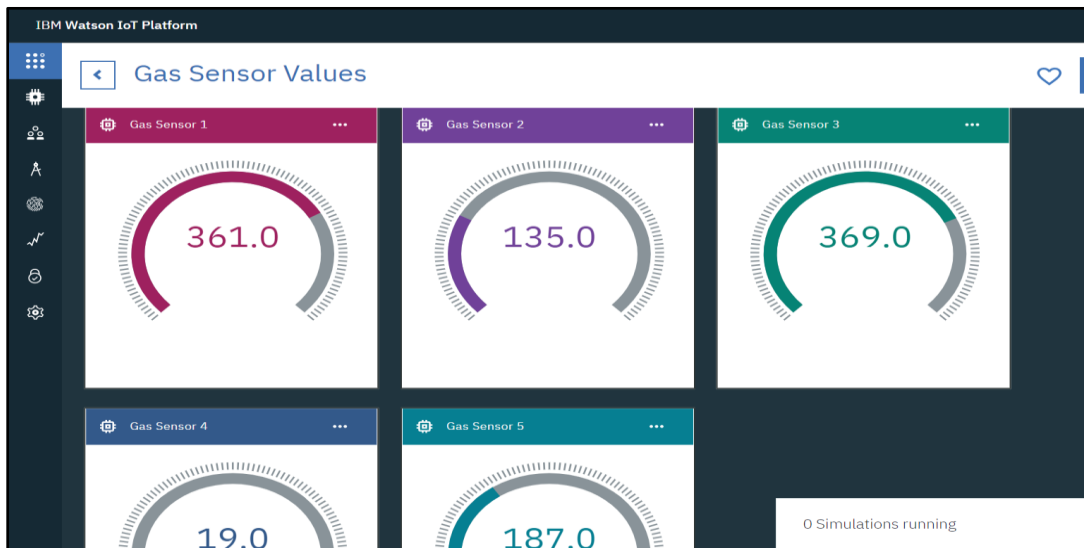
# Disconnect the device and application from the cloud
deviceCli.disconnect()

```

Output:

```
PS D:\Nalaiya Thiran\Assignments\Thiyanesh\Assignment 3> py .\GasSensorData.py
2022-11-19 17:28:10,782 ibmiotf.device.Client INFO Connected successfully: d:liv8qw:sthiyanesh:sthiyanesh
Published {'SensorData': {'Sensor1': 216, 'Sensor2': 41, 'Sensor3': 310, 'Sensor4': 403, 'Sensor5': 394}, 'ValveData': {'Valve1': True, 'Valve2': False, 'Valve3': False, 'Valve4': False, 'Valve5': False}} to pyIBM Watson
Published {'SensorData': {'Sensor1': 294, 'Sensor2': 8, 'Sensor3': 464, 'Sensor4': 268, 'Sensor5': 56}, 'ValveData': {'Valve1': True, 'Valve2': False, 'Valve3': False, 'Valve4': False, 'Valve5': False}} to pyIBM Watson
Valve 1 is Opened
Valve 2 is Opened
Valve 3 is Closed
Valve 4 is Opened
Valve 5 is Opened
Published {'SensorData': {'Sensor1': 131, 'Sensor2': 148, 'Sensor3': 366, 'Sensor4': 264, 'Sensor5': 134}, 'ValveData': {'Valve1': True, 'Valve2': True, 'Valve3': False, 'Valve4': True, 'Valve5': True}} to pyIBM Watson
Valve 1 is Opened
Valve 2 is Opened
Valve 3 is Closed
Valve 4 is Opened
Valve 5 is Opened
Published {'SensorData': {'Sensor1': 275, 'Sensor2': 319, 'Sensor3': 145, 'Sensor4': 132, 'Sensor5': 131}, 'ValveData': {'Valve1': True, 'Valve2': True, 'Valve3': False, 'Valve4': True, 'Valve5': True}} to pyIBM Watson
Valve 1 is Opened
```

Output of Python Code connecting to IBM Watson IoT Platform
Published and Subscribed Data to the Device in IBM Watson IoT Platform



Visualization of Data published to the Device in IBM Watson IoT Platform

Python Code Link:

Link of the Python Code File:

<https://github.com/IBM-EPBL/IBM-Project-2995-1658493718/blob/main/Project%20Development%20phase/Sprint%20-%202/GasSensorData.py>

Demo Link:

Link of the Demo Video of Output:

<https://drive.google.com/drive/folders/1ZesOrCqdZJQKc8DVe00X15nj0hnsbTc?usp=sharing>