

Sprint- 2

Team ID	PNT2022TMID17768
Project Title	Gas Leakage Monitoring And Alerting System
Date	15.11.2022

IBM Watson and Python Integration:

By using Watson IoT Platform, you can collect connected device data and perform analytics on real-time data. The IBM Watson IoT Platform is a fully managed, Cloud-hosted service that provides device management capabilities as well as data collection and management in a time series format.



Your device or gateway

Start with your device and connect it with an IBM Cloud recipe.



MQTT and HTTP

Connect to the IBM Cloud using open, lightweight MQTT or HTTP.



IBM Watson® IoT Platform

Manage connected devices so your apps can access live and historical data.



REST and real-time APIs

Use highly-secure APIs to connect your apps with data from your devices.



Your application and analytics

Create analytic apps in the IBM Cloud, another cloud or your own servers.

Using the Device Created in IBM Watson:

The screenshot shows the IBM Watson IoT Platform interface. The top navigation bar includes 'Browse', 'Action', 'Device Types', and 'Interfaces'. A search bar is present with the text 'Search by Device ID'. The main content area displays a table of devices. The first device, with ID '1234', is shown in a 'Connected' state. Below the table, a detailed view of the device is shown, including its identity, device information, recent events, state, and logs. The 'Device Information' tab is selected, showing details such as Device ID (1234), Device Type (ESP32), Date Added (Nov 12, 2022 1:33 PM), Added By (910619106043@smartinternz.com), and Connection Status (Connected). The connection status details include Connection Time (Nov 15, 2022 11:58 PM) and Client Address (157.49.71.153 SecureToken). A status bar at the bottom indicates '2 Simulations running'.

Device ID	Status	Device Type	Class ID	Date Added	Descriptive Location
1234	Connected	ESP32	Device	Nov 12, 2022 1:33 PM	

Device Information:

- Device ID: 1234
- Device Type: ESP32
- Date Added: Nov 12, 2022 1:33 PM
- Added By: 910619106043@smartinternz.com
- Connection Status: Connected
- Connection Time: Nov 15, 2022 11:58 PM
- Client Address: 157.49.71.153 SecureToken

2 Simulations running

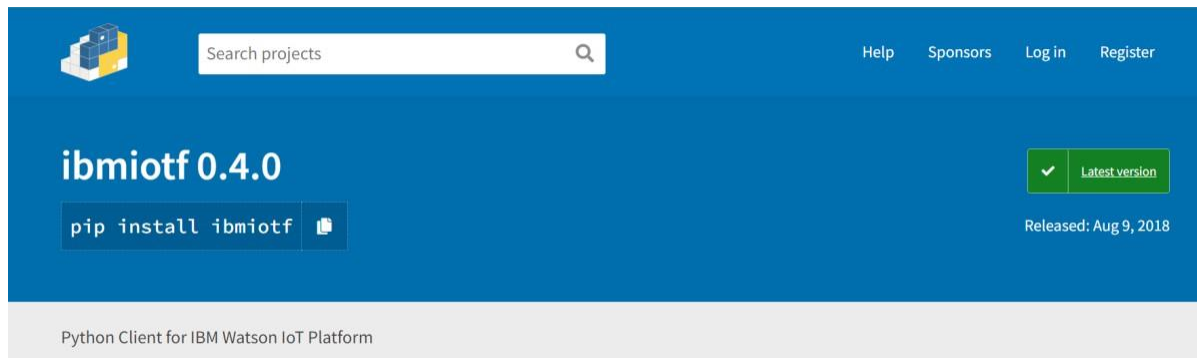
The screenshot shows the IBM Watson IoT Platform interface, specifically the 'Recent Events' tab for the device with ID '1234'. The tab displays a live stream of data coming and going from the device. The data is presented in a table with columns for Event, Value, Format, and Last Received. The events are generated by an IoT Sensor and contain JSON data representing temperature, humidity, and gas concentration. The status bar at the bottom indicates '2 Simulations running'.

Event	Value	Format	Last Received
IoT Sensor	{"temp":17,"Humid":97,"gasconcentration":12}	json	a few seconds ago
IoT Sensor	{"temp":61,"Humid":49,"gasconcentration":48}	json	a few seconds ago
IoT Sensor	{"temp":91,"Humid":49,"gasconcentration":77}	json	a few seconds ago
IoT Sensor	{"temp":51,"Humid":79,"gasconcentration":43}	json	a few seconds ago
IoT Sensor	{"temp":52,"Humid":52,"gasconcentration":57}	json	a few seconds ago

2 Simulations running

Connected sign shows that it is connected and live

Python code execution:



Install this package : Python Client for IBM Watson IoT Platform

python code:

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

#Provide your IBM Watson Device Credentials

```
organization = "z9xrcm"
deviceType = "ESP32"
deviceId = "1234"
authMethod = "token"
authToken = "12345678"
```

Initialize GPIO

```
def myCommandCallback(cmd):
```

```
    print("Command received: %s" % cmd.data['command'])
    status=cmd.data['command']
    if status=="sprinkleron":
        print ("Sprinkler is on")
    else :
        print ("Sprinkler is off")
```

```
#print(cmd)
```

```
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:
    #Get Sensor Data from DHT11

    temp=random.randint(0,100)
    Humid=random.randint(0,100)
    gasconcentration=random.randint(0,100)

    data = { 'temp': temp, 'Humid': Humid, "gasconcentration": gasconcentration}#print

    data
    def myOnPublishCallback():
        print ("Published Temperature = %s C" % temp, "Humidity = %s %" %
Humid, "gasconcentration = %s %" % gasconcentration, "to IBM 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()

```

```
code.py - C:\Users\bala\AppData\Local\Programs\Python\Python36-32\code.py (3.6.0)
File Edit Format Run Options Window Help

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

#Provide your IBM Watson Device Credentials
organization = "z9xrcm"
deviceType = "ESF32"
deviceId = "1234"
authMethod = "token"
authToken = "12345678"

# Initialize GPIO

def myCommandCallback(cmd):
    print("Command received: %s" % cmd.data['command'])
    status=cmd.data['command']
    if status=="sprinkleron":
        print ("Sprinkler is on")
    else:
        print ("Sprinkler is off")
    #print(cmd)

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:
    #Get Sensor Data from DHT11
    #.....

Ln: 57 Col: 0
```

```
code.py - C:\Users\bala\AppData\Local\Programs\Python\Python36-32\code.py (3.6.0)
File Edit Format Run Options Window Help

#print(cmd)

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:
    #Get Sensor Data from DHT11
    temp=random.randint(0,100)
    Humid=random.randint(0,100)
    gasconcentration=random.randint(0,100)

    data = { 'temp' : temp, 'Humid': Humid, "gasconcentration": gasconcentration}

    #print data
    def myOnPublishCallback():
        print ("Published Temperature = %s C" % temp, "Humidity = %s %% " % Humid, "gasconcentration = %s %% " % gasconcentration, "to IBM 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()

Ln: 57 Col: 0
```

```
code.py - C:\Users\bala\AppData\Local\Programs\Python\Python36-32\code.py
import time
import sys
import ibmiotf.application
import ibmiotf.device
import random

#Provide your IBM Watson Device Credentials
organization = "z9xrcm"
deviceType = "ESP32"
deviceId = "1234"
authMethod = "token"
authToken = "12345678"

# Initialize GPIO

def myCommandCallback(cmd):
    print("Command received: %s" % cmd.data['command'])
    status=cmd.data['command']
    if status=="sprinkleron":
        print ("Sprinkler is on")
    else:
        print ("Sprinkler is off")
    #print(cmd)

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
deviceCli.connect()

while True:
    #Get Sensor Data from DHT11
    temperature, humidity, gasconcentration = DHT11.read()
    #Send data to cloud
    deviceCli.publishEvent("hello", "world", {
        "temp": temperature,
        "humid": humidity,
        "gasconcentration": gasconcentration
    })
    time.sleep(10)
```

```
*Python 3.6.0 Shell*
Python 3.6.0 (v3.6.0:41df79263a11, Dec 23 2016, 07:18:10) [MSC v.1900 32 bit (Intel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>>
-- RESTART: C:\Users\bala\AppData\Local\Programs\Python\Python36-32\code.py --
2022-11-16 02:20:17.405 ibmiotf.device.Client INFO Connected successfully: d:z9xrcm:ESP32:1234
Published Temperature = 88 C Humidity = 59 % gasconcentration = 78 % to IBM Watson
Published Temperature = 28 C Humidity = 99 % gasconcentration = 87 % to IBM Watson
Published Temperature = 32 C Humidity = 60 % gasconcentration = 8 % to IBM Watson
Published Temperature = 41 C Humidity = 54 % gasconcentration = 67 % to IBM Watson
Published Temperature = 81 C Humidity = 64 % gasconcentration = 17 % to IBM Watson
Published Temperature = 51 C Humidity = 93 % gasconcentration = 38 % to IBM Watson
Published Temperature = 5 C Humidity = 1 % gasconcentration = 79 % to IBM Watson
Published Temperature = 44 C Humidity = 88 % gasconcentration = 69 % to IBM Watson
Published Temperature = 76 C Humidity = 54 % gasconcentration = 27 % to IBM Watson
Published Temperature = 37 C Humidity = 78 % gasconcentration = 10 % to IBM Watson
```

Recent Events in IBM Watson IoT Platform:

The screenshot shows the IBM Watson IoT Platform interface. The top navigation bar includes 'Browse', 'Action', 'Device Types', and 'Interfaces'. A sidebar on the left contains icons for various functions. The main content area displays the 'Recent Events' tab for a device with ID '1234'. The device is currently 'Disconnected' and is of type 'python'. Below the 'Recent Events' tab, a table lists the live stream of data coming and going from the device. The table has four columns: 'Event', 'Value', 'Format', and 'Last Received'. The events listed are all 'IoT Sensor' events with JSON values for temperature, humidity, and gasconcentration. The last received time for all events is 'a few seconds ago'. At the bottom of the interface, a status bar shows '2 Simulations running'.

Event	Value	Format	Last Received
IoT Sensor	{"temp":17,"Humid":97,"gasconcentration":12}	json	a few seconds ago
IoT Sensor	{"temp":61,"Humid":49,"gasconcentration":48}	json	a few seconds ago
IoT Sensor	{"temp":91,"Humid":49,"gasconcentration":77}	json	a few seconds ago
IoT Sensor	{"temp":51,"Humid":79,"gasconcentration":43}	json	a few seconds ago
IoT Sensor	{"temp":52,"Humid":52,"gasconcentration":57}	json	a few seconds ago

Boards in IBM Platform:

