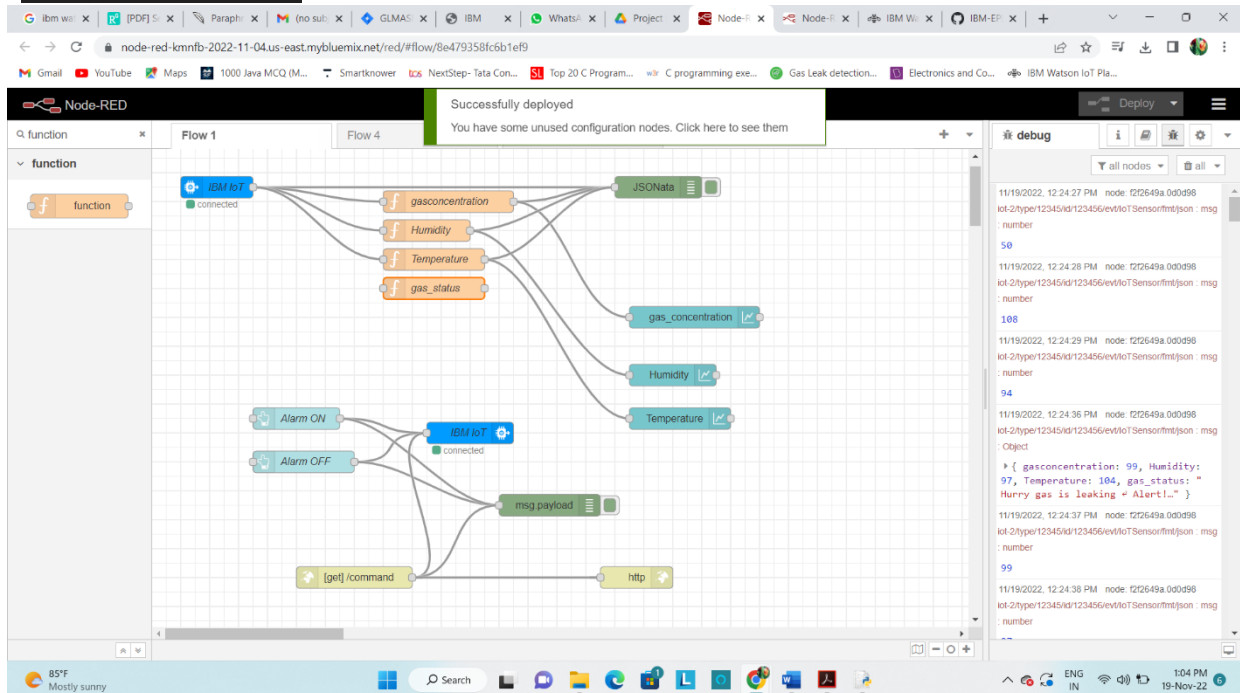


PROJECT DEVELOPMENT PHASE

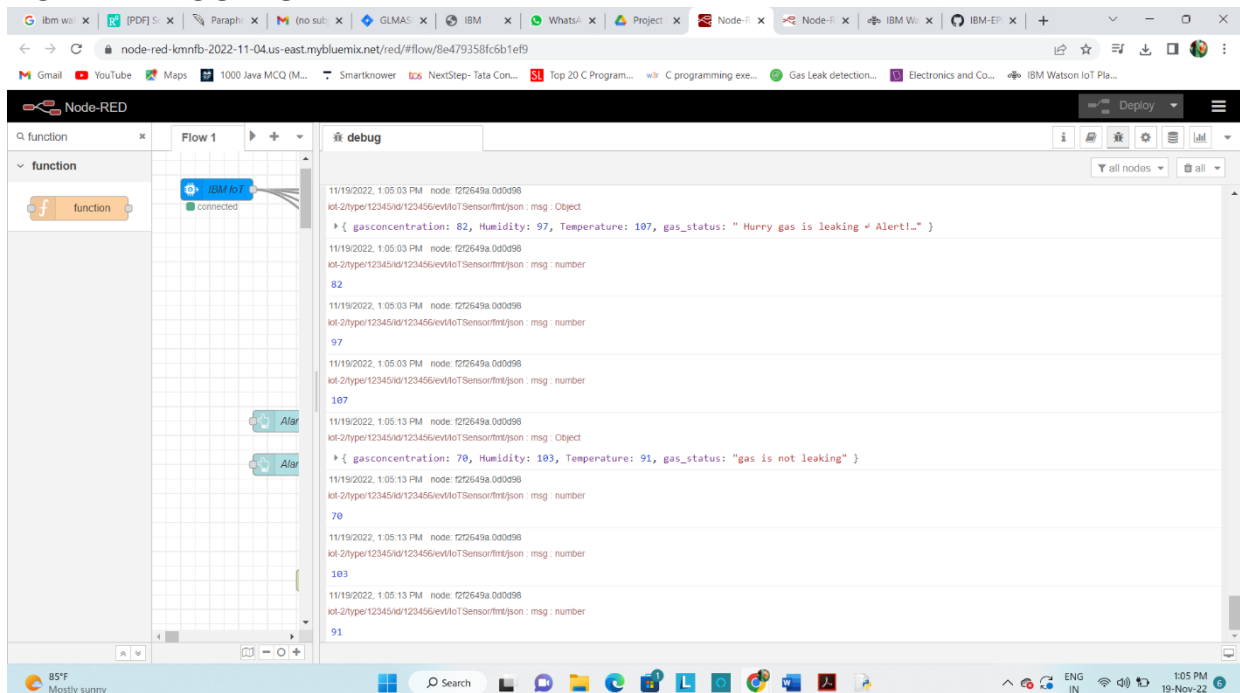
SPRINT 3

Date	19 November 2022
Team ID	PNT2022TMID13514
Project name	Gas Leakage Monitoring & Alerting System for Industries

NODE RED FLOW



NODE RED OUTPUT:



CLOUDANT CONNECTION IN NODE-RED

The screenshot shows the Node-RED web interface in a browser. The main workspace displays a flow diagram with the following components:

- Flow 1:** A `[get] /gas` node connected to a `gasdetection` node, which is then connected to an `http` node.
- Flow 4:** An `IBM IoT` node (labeled "connected") connected to a `json` node, which is then connected to a `[ws] /ws/gas` node.
- Flow 5:** A `mydb` node connected to the `json` node in Flow 4.

The right-hand sidebar shows the **debug** console, displaying a series of log messages. The messages are JSON objects containing sensor data and status updates, such as:

```
{ gasconcentration: 87, Humidity: 99, Temperature: 107, gas_status: "Hurry gas is leaking - Alert!" }
```

The bottom of the interface shows a Windows taskbar with various application icons and a system tray indicating the time as 1:09 PM on 19-Nov-22.

NODE RED DASHBOARD:

The screenshot shows the Node-RED Dashboard interface. The top section is titled **Smart Industry**. Below this, there are three line graphs under the heading **Gas Detection**:

- gas_concentration:** A line graph showing fluctuating values between approximately 90 and 110 over a time period from 20:06:00 to 20:12:00.
- Humidity:** A line graph showing fluctuating values between approximately 90 and 110 over the same time period.
- Temperature:** A line graph showing fluctuating values between approximately 90 and 110 over the same time period.

Below the graphs is a section titled **Smart Switch Board**. It contains two large blue buttons:

- ALARM OFF**
- ALARM ON**

The bottom of the interface shows a Windows taskbar with various application icons and a system tray indicating the time as 8:11 PM on 18-Nov-22.

10:57



st.mybluemix.net



21



Smart Industry

Gas Detection



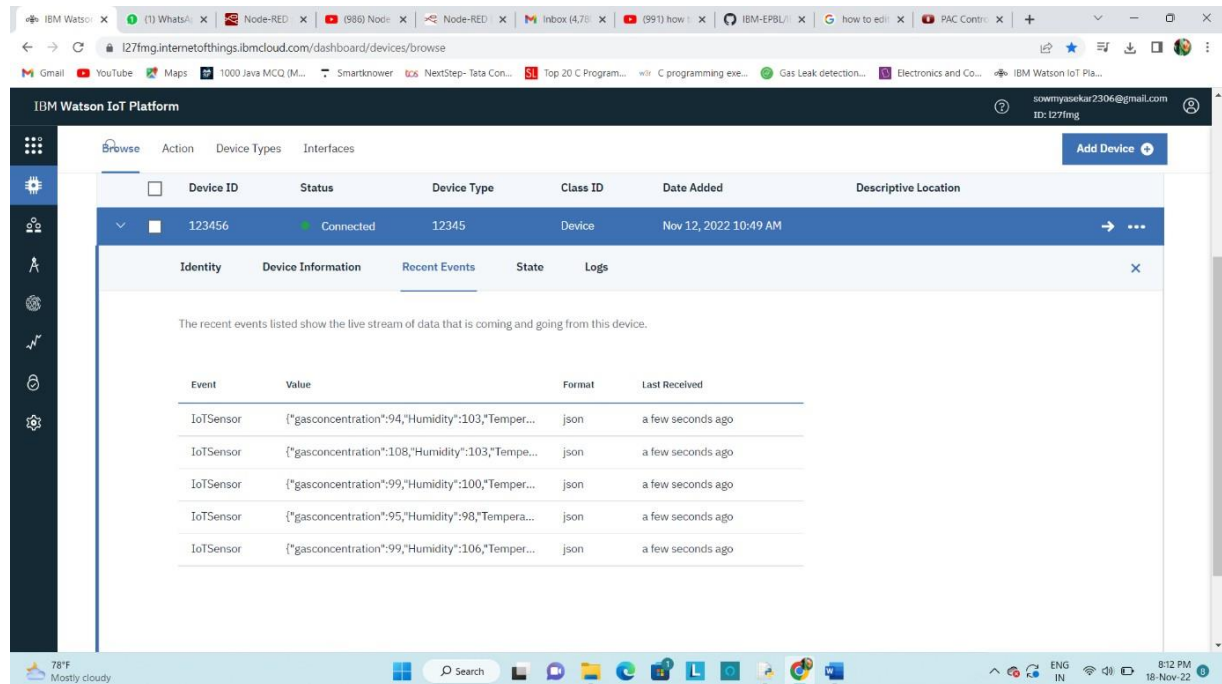
Smart Switch Board

ALARM OFF

ALARM ON



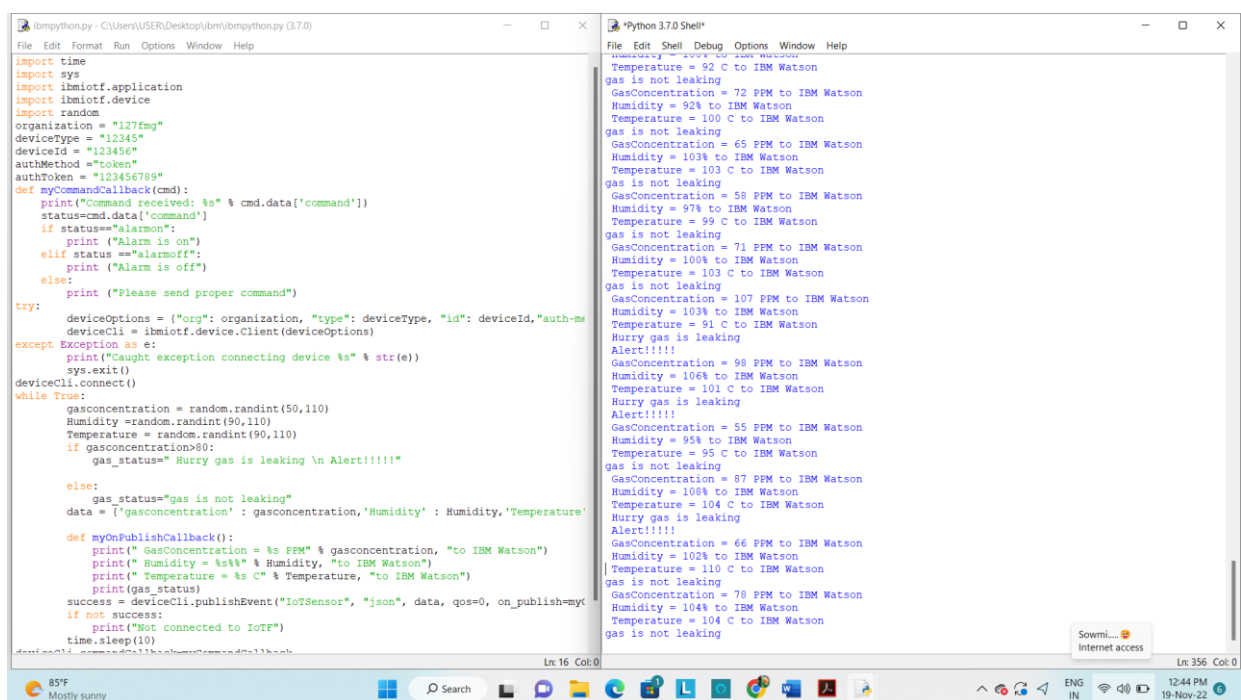
RECEIVE OF MESSAGE FROM WATSON:



The screenshot displays the IBM Watson IoT Platform interface. The top navigation bar includes 'Browse', 'Action', 'Device Types', and 'Interfaces'. The main content area shows a table of devices, with one device (ID: 123456) selected. Below the table, the 'Recent Events' tab is active, displaying a list of events received from the device. The events are summarized in the following table:

Event	Value	Format	Last Received
IoTSensor	{"gasconcentration":94,"Humidity":103,"Temper...	json	a few seconds ago
IoTSensor	{"gasconcentration":108,"Humidity":103,"Tempe...	json	a few seconds ago
IoTSensor	{"gasconcentration":99,"Humidity":100,"Temper...	json	a few seconds ago
IoTSensor	{"gasconcentration":95,"Humidity":98,"Tempera...	json	a few seconds ago
IoTSensor	{"gasconcentration":99,"Humidity":106,"Temper...	json	a few seconds ago

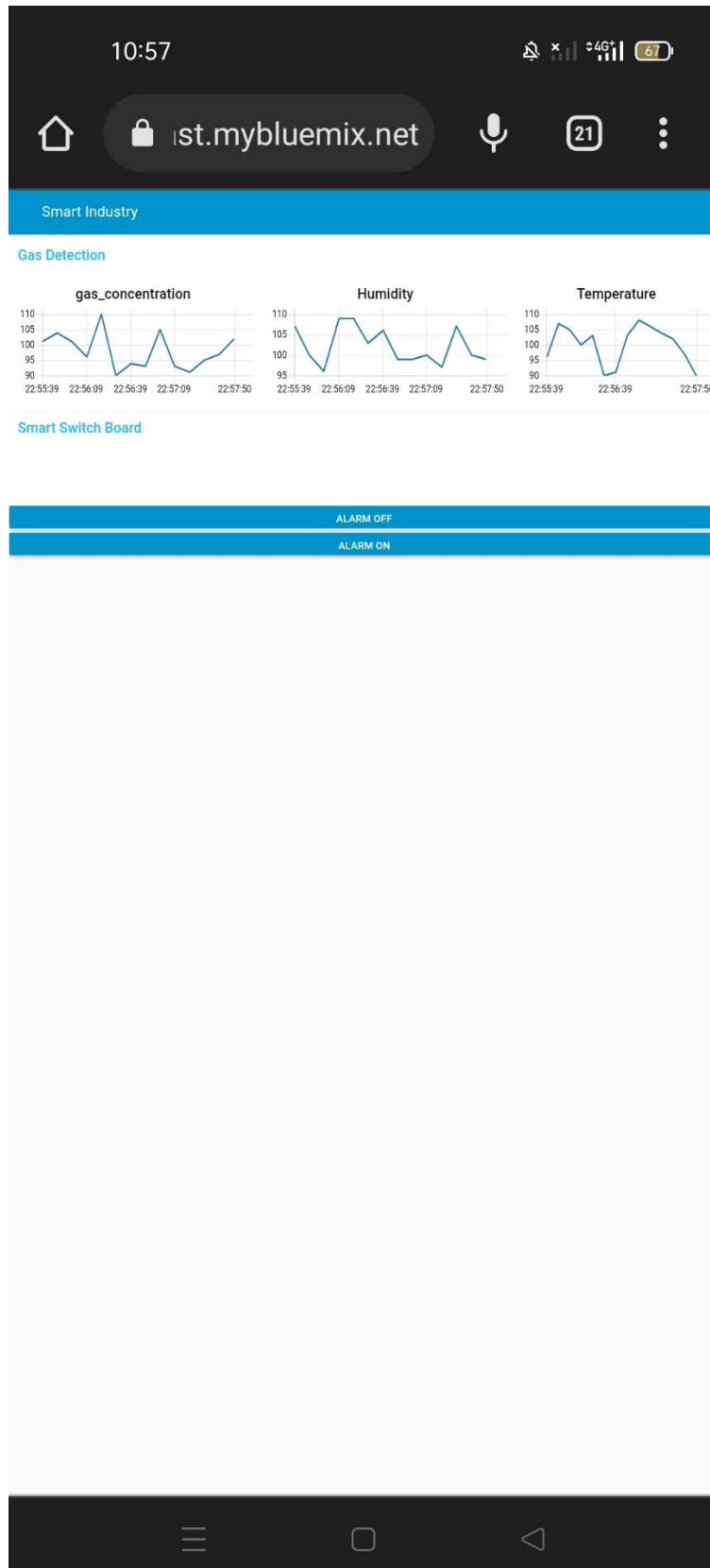
PYTHON CODE OUTPUT:

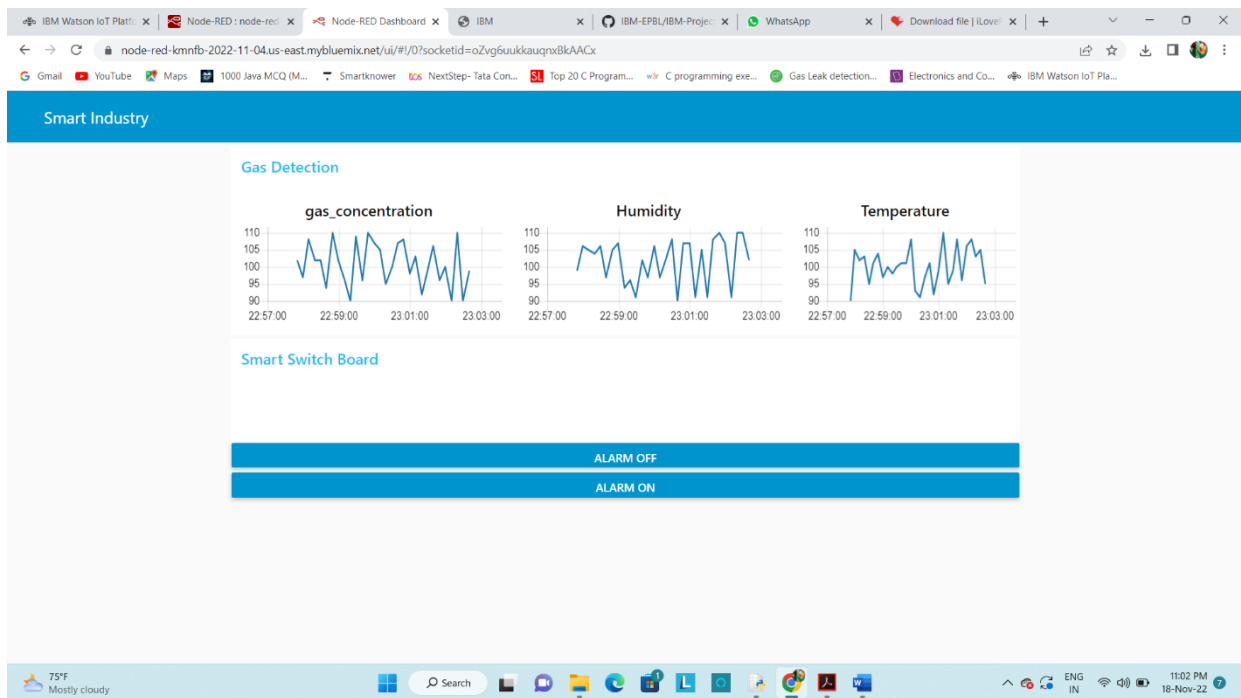


The screenshot shows a Python script running in a terminal window. The script is designed to simulate a gas leak detection system. It generates random values for gas concentration, humidity, and temperature, and then publishes these values to the IBM Watson IoT Platform. The output of the script is displayed in the terminal window, showing the generated data and the status of the gas leak detection system.

```
File Edit Shell Debug Options Window Help
Python 3.7.0 Shell*
temp = 92 C to IBM Watson
Temperature = 92 C to IBM Watson
gas is not leaking
GasConcentration = 72 PPM to IBM Watson
Humidity = 92% to IBM Watson
Temperature = 100 C to IBM Watson
gas is not leaking
GasConcentration = 65 PPM to IBM Watson
Humidity = 103% to IBM Watson
Temperature = 103 C to IBM Watson
gas is not leaking
GasConcentration = 58 PPM to IBM Watson
Humidity = 97% to IBM Watson
Temperature = 99 C to IBM Watson
gas is not leaking
GasConcentration = 71 PPM to IBM Watson
Humidity = 100% to IBM Watson
Temperature = 103 C to IBM Watson
gas is not leaking
GasConcentration = 107 PPM to IBM Watson
Humidity = 103% to IBM Watson
Temperature = 91 C to IBM Watson
Hurry gas is leaking
Alert!!!!!!
GasConcentration = 98 PPM to IBM Watson
Humidity = 106% to IBM Watson
Temperature = 101 C to IBM Watson
Hurry gas is leaking
Alert!!!!!!
GasConcentration = 55 PPM to IBM Watson
Humidity = 95% to IBM Watson
Temperature = 95 C to IBM Watson
gas is not leaking
GasConcentration = 87 PPM to IBM Watson
Humidity = 108% to IBM Watson
Temperature = 104 C to IBM Watson
Hurry gas is leaking
Alert!!!!!!
GasConcentration = 66 PPM to IBM Watson
Humidity = 102% to IBM Watson
Temperature = 110 C to IBM Watson
gas is not leaking
GasConcentration = 78 PPM to IBM Watson
Humidity = 104% to IBM Watson
Temperature = 104 C to IBM Watson
gas is not leaking
```

WEB UI:





CLOUDANT:

The screenshot shows the Cloudant dashboard interface. On the left is a sidebar with navigation options: "All Documents", "Query", "Permissions", "Changes", and "Design Documents". The main area displays a table of documents. The table has three columns: "id", "key", and "value". The "value" column contains JSON objects with a "rev" field and a "features" field. The table is sorted by "id" in ascending order. At the bottom, it indicates "Showing document 1 - 20" and "page: 20".

id	key	value
0d910213621f26c419c01ddc60aefd4	0d910213621f26c419c01ddc60aefd4	{ "rev": "2-1e500b59f0093cd9799dba330c7f1..." }
0d910213621f26c419c01ddc60bd4fed	0d910213621f26c419c01ddc60bd4fed	{ "rev": "1-c5abb185a5ef51cd426f7309dea..." }
0d910213621f26c419c01ddc60d002fa	0d910213621f26c419c01ddc60d002fa	{ "rev": "1-1b4b7d9d3b16909e9e2c9a681022..." }
163b140cdd6c0710c4532159969105db	163b140cdd6c0710c4532159969105db	{ "rev": "1-61cd9af17f30a11c50952d533935..." }
1a96abc4136dc3acd540191c43641a35	1a96abc4136dc3acd540191c43641a35	{ "rev": "1-1c6154788545f245fbaad43c60f4..." }
1c93c65d488f96833e85493972d1409	1c93c65d488f96833e85493972d1409	{ "rev": "1-ec0c82c5476cece47bf44588cc1e..." }
1c93c65d488f96833e854939729f50	1c93c65d488f96833e854939729f50	{ "rev": "1-dffe2b5215adde84ea60a77e9bda..." }
1c93c65d488f96833e85493974b8caf	1c93c65d488f96833e85493974b8caf	{ "rev": "1-c7d348f9f125a085a16614012191..." }
1c93c65d488f96833e854939754e763	1c93c65d488f96833e854939754e763	{ "rev": "1-503c12a35ecac3822a553f32dcc8..." }
1c93c65d488f96833e8549397cfb571	1c93c65d488f96833e8549397cfb571	{ "rev": "1-cab8ae88b2031c4615d5dd191db..." }
1c93c65d488f96833e8549397d46ab5	1c93c65d488f96833e8549397d46ab5	{ "rev": "1-57a0f0c0000000000000000000000000..." }
1ea8bb627676e5fe989c080165ac46cb	1ea8bb627676e5fe989c080165ac46cb	{ "rev": "1-c102..." }

PYTHON CODE:

```
import time
```

```

import sys
import ibmiotf.application
import ibmiotf.device
import random
organization = "l27fmg"
deviceType = "12345"
deviceId = "123456"
authMethod = "token"
authToken = "123456789"
def myCommandCallback(cmd):
    print("Command received: %s" % cmd.data['command'])
    status=cmd.data['command']
    if status=="alarmon":
        print ("Alarm is on")
    elif status=="alarmoff":
        print ("Alarm 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 Exception as e:
    print("Caught exception connecting device %s" % str(e))
    sys.exit()
deviceCli.connect()
while True:
    gasconcentration = random.randint(50,110)
    Humidity =random.randint(90,110)
    Temperature = random.randint(90,110)
    if gasconcentration>80:
        gas_status=" Hurry gas is leaking \n Alert!!!!!"

    else:
        gas_status="gas is not leaking"
    data = {'gasconcentration' : gasconcentration,'Humidity' : Humidity,'Temperature'
:Temperature,'gas_status':gas_status}

    def myOnPublishCallback():
        print(" GasConcentration = %s PPM" % gasconcentration, "to IBM Watson")
        print(" Humidity = %s%%" % Humidity, "to IBM Watson")
        print(" Temperature = %s C" % Temperature, "to IBM Watson")
        print(gas_status)
        success = deviceCli.publishEvent("IoTSensor", "json", data, qos=0,
on_publish=myOnPublishCallback)
        if not success:
            print("Not connected to IoT")
            time.sleep(10)
deviceCli.commandCallback=myCommandCallback
deviceCli.disconnect()

```