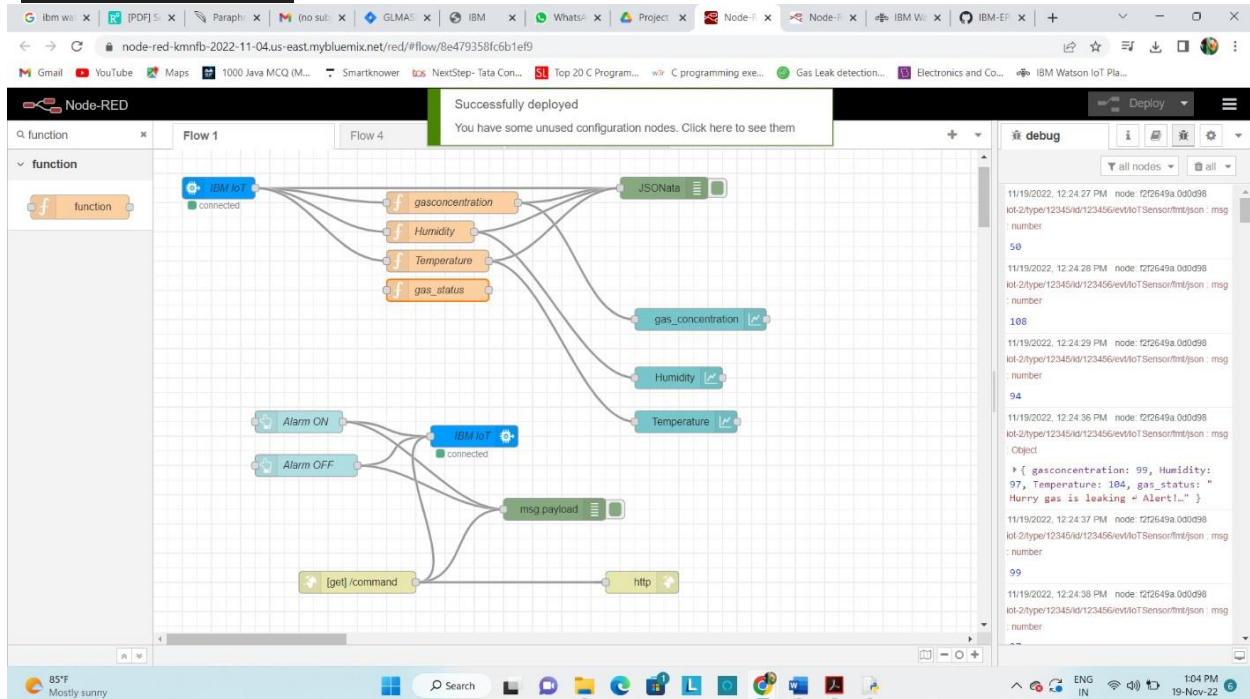


PROJECT DEVELOPMENT PHASE

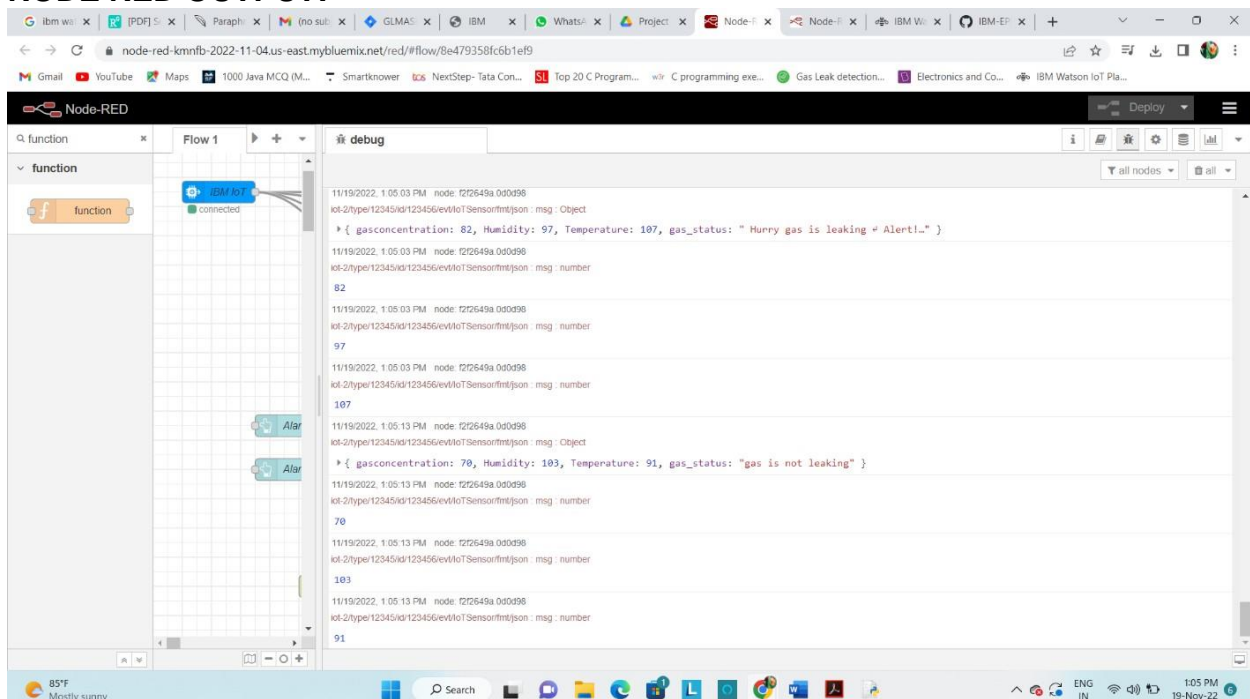
SPRINT 3

Date	18 November 2022
Team ID	PNT2022TMID53681
Project name	Gas Leakage Monitoring & Alerting System for Industries

NODE RED FLOW



NODE RED OUTPUT:



CLOUDANT CONNECTION IN NODE-RED

The screenshot displays the Node-RED web interface in a browser. The main workspace shows a flow with the following components: a 'get' node for /gas, a 'gasdetection' node, an 'http' node, an 'IBM IoT' node (labeled 'connected'), a 'json' node, a 'mydb' node, and a 'ws' node for /ws/gas. The debug console on the right shows a series of incoming JSON messages, each containing gas concentration, humidity, and temperature data. The messages are as follows:

```

{
  "gasconcentration": 87,
  "Humidity": 99,
  "Temperature": 107,
  "gas_status": "Hurry gas is leaking + Alert!.."
}
{
  "gasconcentration": 85,
  "Humidity": 93,
  "Temperature": 96,
  "gas_status": "Hurry gas is leaking + Alert!.."
}

```

NODE RED DASHBOARD:

The screenshot shows the Node-RED dashboard titled 'Smart Industry'. It contains three line graphs for 'gas_concentration', 'Humidity', and 'Temperature'. The 'gas_concentration' graph shows values fluctuating between 90 and 110. The 'Humidity' graph shows values fluctuating between 90 and 110. The 'Temperature' graph shows values fluctuating between 90 and 110. Below the graphs is a 'Smart Switch Board' section with two buttons: 'ALARM OFF' and 'ALARM ON'.

10:57



ist.mybluemix.net



21



Smart Industry

Gas Detection



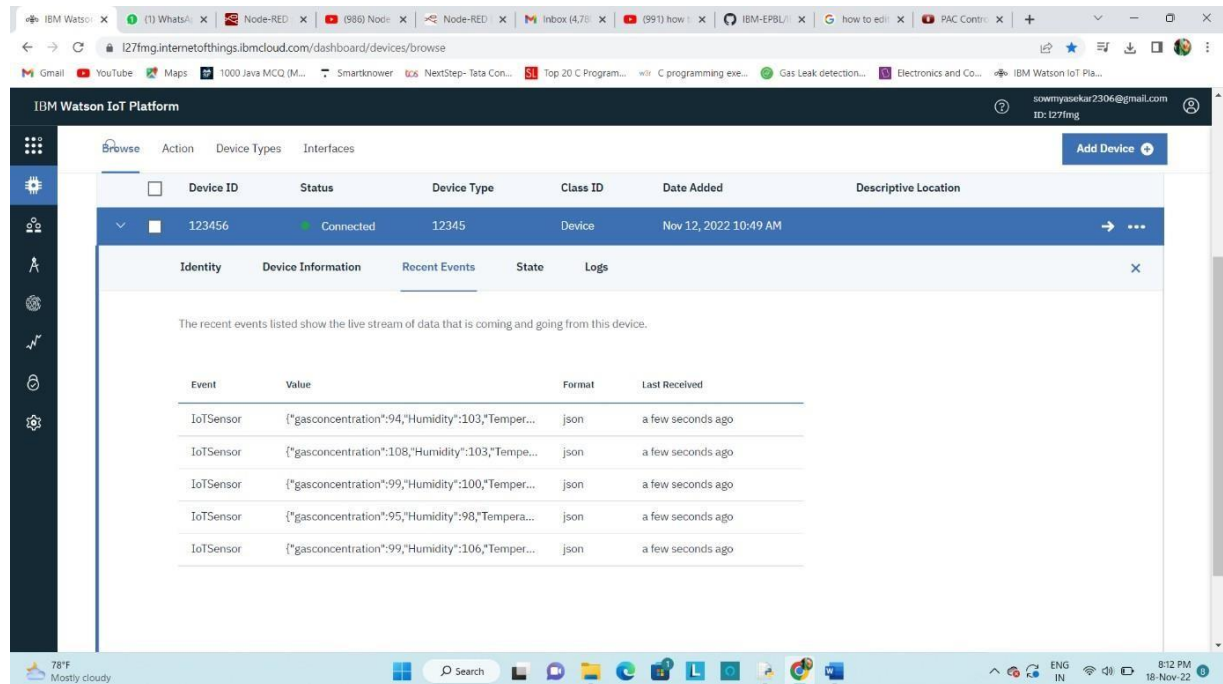
Smart Switch Board

ALARM OFF

ALARM ON



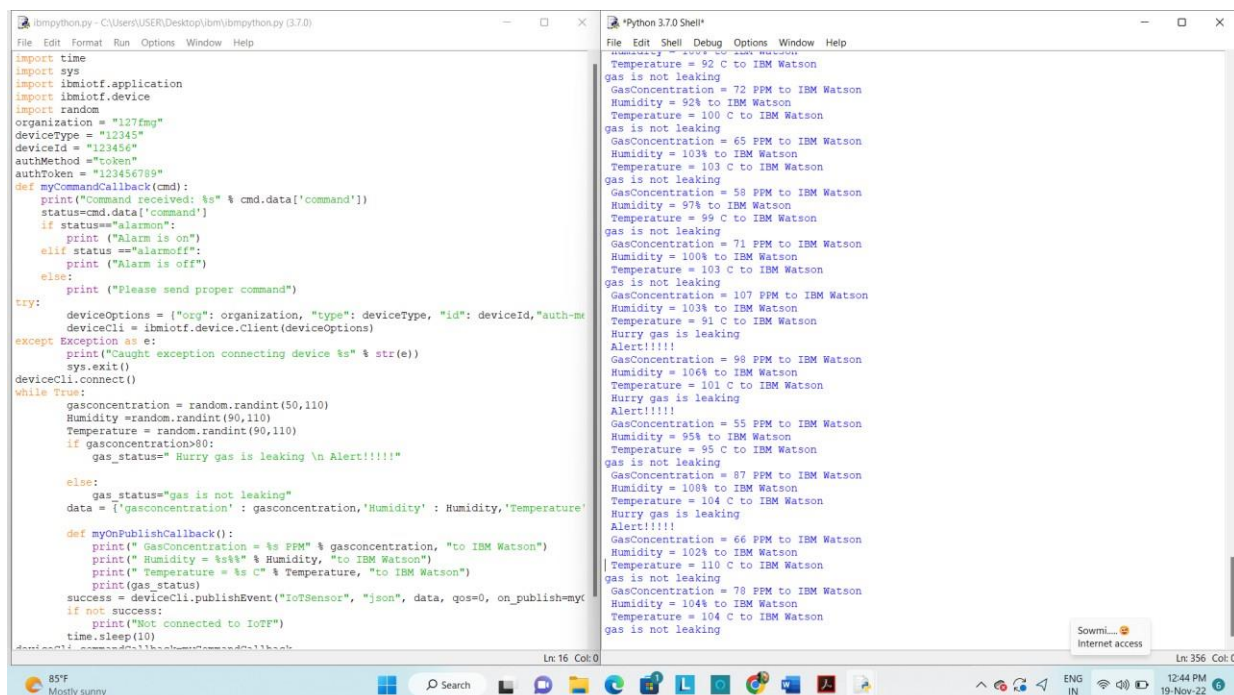
RECEIVE OF MESSAGE FROM WATSON:



The screenshot shows the IBM Watson IoT Platform interface. The top navigation bar includes 'Browse', 'Action', 'Device Types', and 'Interfaces'. A table lists devices, with one device (ID: 123456) highlighted. Below the table, the 'Recent Events' tab is selected, displaying a list of events. The events are JSON messages containing gas concentration, humidity, and temperature data, received from an IoT sensor.

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 in a text editor and its output in a terminal window. The script simulates an IoT device sending data to the IBM Watson IoT Platform. The output shows the device sending a series of JSON messages containing gas concentration, humidity, and temperature data, which are received by the Watson IoT Platform.

```
import time
import sys
import ibmiotf.application
import ibmiotf.device
import random
organization = "127fmg"
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-m": authMethod, "auth-t": authToken}
    deviceCli = ibmiotf.device.Client(deviceOptions)
except Exception as e:
    print("Caught exception connecting device %s" % str(e))
    sys.exit()

deviceCli.connect()

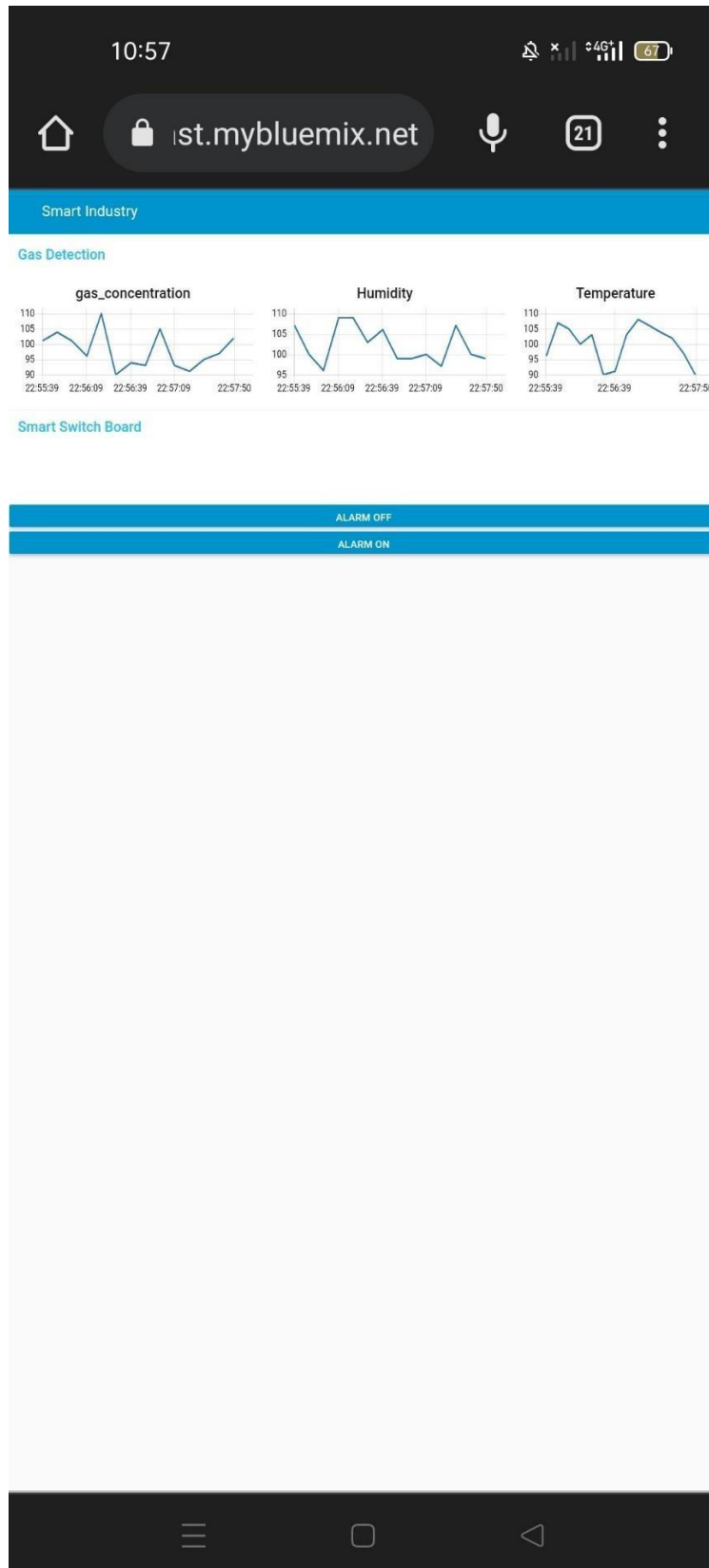
while True:
    gasconcentration = random.randint(50,110)
    Humidity = random.randint(90,110)
    Temperature = random.randint(90,110)
    if gasconcentration>80:
        gas_status="Hurry gas is leaking \n Alert!!!!!"
    else:
        gas_status="gas is not leaking"
    data = {"gasconcentration": gasconcentration, "Humidity": Humidity, "Temperature": Temperature}

    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 IoTF")
    time.sleep(10)
```

Output:

```
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 = 99 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 = 100% 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:




```

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()

```