

## SPRINT-4

TEAM ID	PNT2022TMID29910
PROJECT TITLE	INDUDTRY-SPECIFIC INTELLIGENT FIRE MANAGEMENT SYSTEM

### PYTHON PROGRAM:

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

[#Provide](#) your IBM Watson Device Credentials

```
organization = "a6n32x"
deviceType = "Mainproject"
deviceId = "ibmproject"
authMethod = "token"
authToken = "1234567890"
```

[# Initialize GPIO](#)

```
def myCommandCallback(cmd):
    print("Command received: %s" % cmd.data['command'])
    status=cmd.data['command']
    if status=="lighton":
        print ("led is on")
    else :
        print ("led 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 DHT22,DHT11,

    Temp=random.randint(-20,120)
    Humidity=random.randint(0,120)
    Flame=random.randint(0,100)
    Gas=random.randint(0,80)

    data = {'Temp' :Temp , 'Humidity' : Humidity, 'Flame' : Flame, 'Gas' : Gas}

    def myOnPublishCallback():
        if Flame > 100:
            data = {'Flame' : Flame}

    print ("Temperature =%s c" % Temp , "Humidity =%s u" % Humidity, "Flame =%s ir" % Flame , "Gas
    =%s ppm" % Gas )
    success = deviceCli.publishEvent("IoTSensor", "json", data, qos=0, on_publish=myOnPublishCallback)

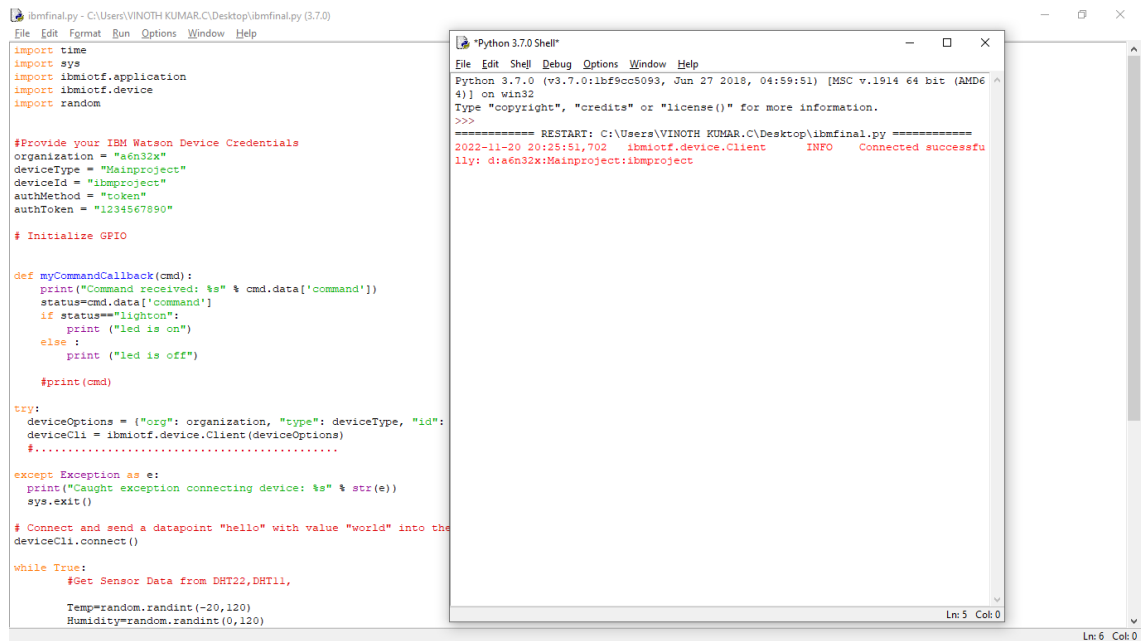
    if not success:
        print("Not connected to IoT")
        time.sleep(1)

    deviceCli.commandCallback = myCommandCallback

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

```

## PYTHON CODE OUTPUT:



The image shows a Python script in a text editor and its execution output in a terminal window.

```
ibmfinal.py - C:\Users\VINOTH KUMAR.C\Desktop\ibmfinal.py (3.7.0)
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 = "a6n32x"
deviceType = "Mainproject"
deviceId = "ibmproject"
authMethod = "token"
authToken = "1234567890"

# Initialize GPIO

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

try:
    deviceOptions = {"org": organization, "type": deviceType, "id":
    deviceCli = ibmiotf.device.Client(deviceOptions)
    #.....

except Exception as e:
    print("Caught exception connecting device: %s" % str(e))
    sys.exit()

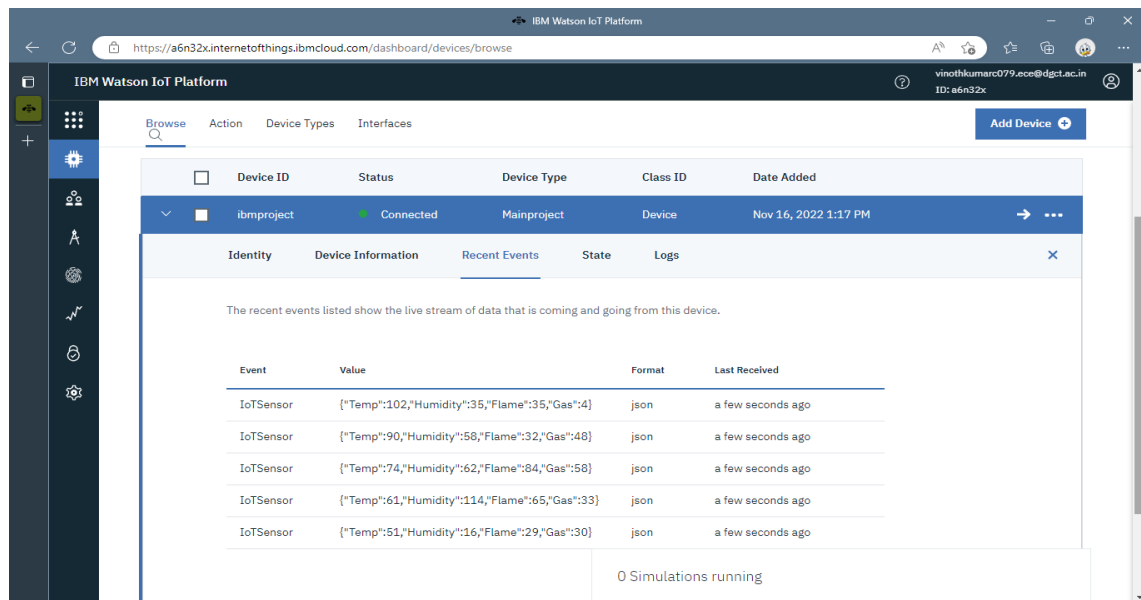
# Connect and send a datapoint "hello" with value "world" into the
deviceCli.connect()

while True:
    #Get Sensor Data from DHT22,DHT11,
    Temp=random.randint(-20,120)
    Humidity=random.randint(0,120)
```

The terminal window shows the output of the script:

```
Python 3.7.0 Shell
Python 3.7.0 (v3.7.0:1bf9cc5093, Jun 27 2018, 04:59:51) [MSC v.1914 64 bit (AMD6
4)] on win32
Type "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: C:\Users\VINOTH KUMAR.C\Desktop\ibmfinal.py =====
2022-11-20 20:25:51,702 ibmiotf.device.Client INFO Connected successfu
lly: d:a6n32x:Mainproject:ibmproject
```

## IBM WATSON OUTPUT:



The image shows the IBM Watson IoT Platform dashboard. The top navigation bar includes "Browse", "Action", "Device Types", and "Interfaces". The main content area displays a table of devices.

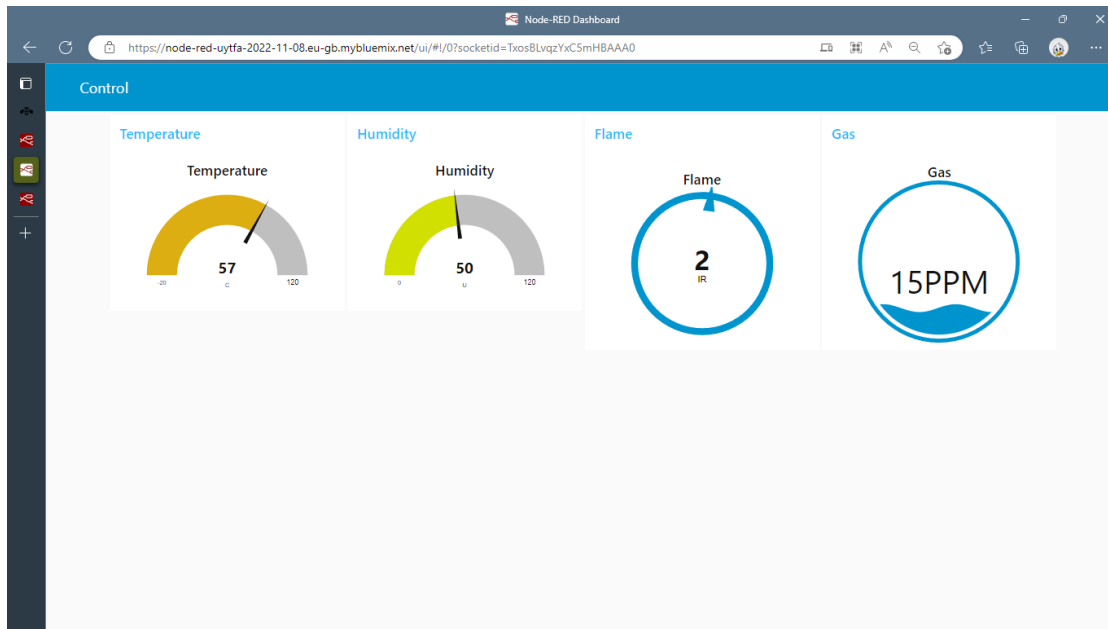
Device ID	Status	Device Type	Class ID	Date Added
ibmproject	Connected	Mainproject	Device	Nov 16, 2022 1:17 PM

Below the table, there is a section for "Recent Events" showing a live stream of data. The events are listed in a table:

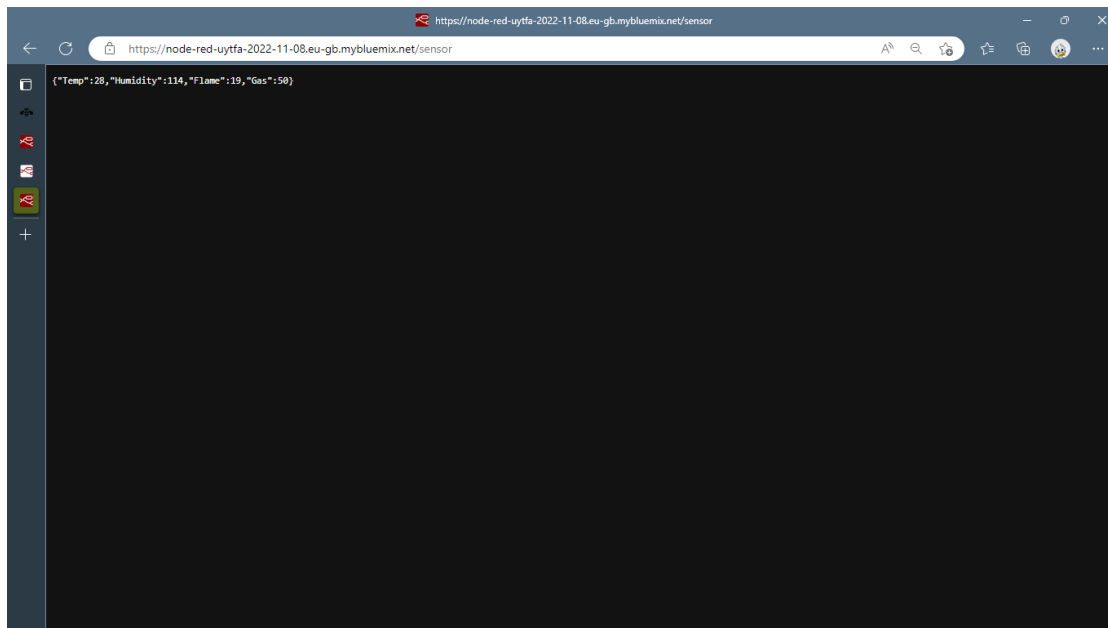
Event	Value	Format	Last Received
IoTSensor	{"Temp":102,"Humidity":35,"Flame":35,"Gas":4}	json	a few seconds ago
IoTSensor	{"Temp":90,"Humidity":58,"Flame":32,"Gas":48}	json	a few seconds ago
IoTSensor	{"Temp":74,"Humidity":62,"Flame":84,"Gas":58}	json	a few seconds ago
IoTSensor	{"Temp":61,"Humidity":114,"Flame":65,"Gas":33}	json	a few seconds ago
IoTSensor	{"Temp":51,"Humidity":16,"Flame":29,"Gas":30}	json	a few seconds ago

At the bottom, it indicates "0 Simulations running".

## NODERED UI OUTPUT:



## NODE RED SENSOR READING:



The screenshot shows the Node-RED interface for sensor readings. The browser's address bar displays the URL: <https://node-red-uytfa-2022-11-08.eu-gb.mybluemix.net/sensor>. The main content area is a dark console displaying a JSON object representing the sensor data:

```
{"Temp":28,"Humidity":114,"Flame":19,"Gas":50}
```