

SPRINT -1

GAS LEAKAGE MONITORING AND ALERTING SYSTEM

Team ID	PNT2022TMID22877
Project Name	Gas Leakage Monitoring and Alerting System for Industries

SIMULATION CREATION USING WOKWI:

CODE:

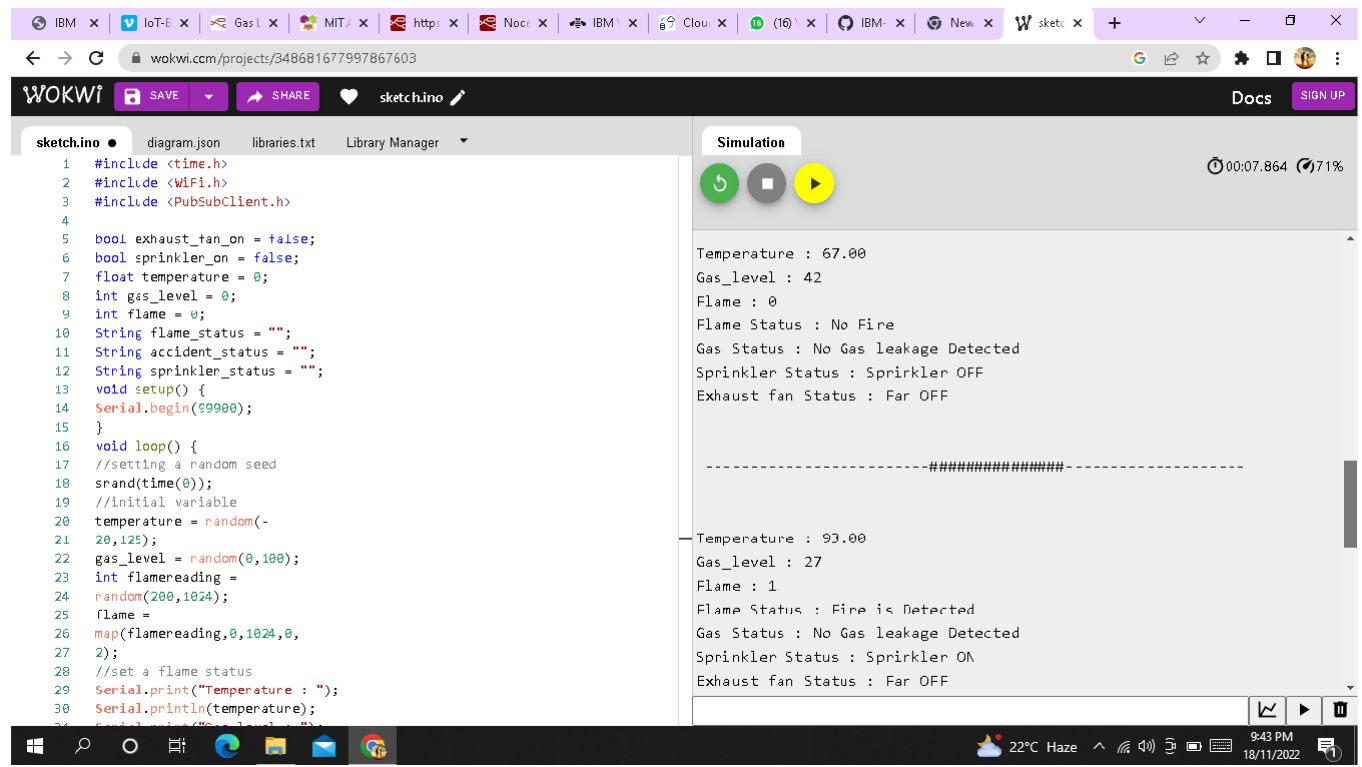
```
#include <time.h>  
#include <WiFi.h>  
#include <PubSubClient.h>  
bool exhaust_fan_on = false;  
bool sprinkler_on = false;  
float temperature = 0;  
int gas_level = 0;  
int flame = 0;  
String flame_status = "";  
String accident_status = "";  
String sprinkler_status = "";  
void setup() {  
Serial.begin(99900);  
}  
void loop() {  
//setting a random seed  
srand(time(0));  
//initial variable  
temperature = random(-  
20,125);  
gas_level = random(0,1000);  
int flamereading =  
random(200,1024);  
flame =  
map(flamereading,0,1024,0,  
2);  
//set a flame status  
Serial.print("Temperature : ");  
Serial.println(temperature);  
Serial.print("Gas_level : ");  
Serial.println(gas_level);  
Serial.print("Flame : ");  
Serial.println(flame);  
switch (flame) {  
case 0:
```

```

flame_status = "No Fire";
Serial.println("Flame Status : "+flame_status);
break;
case 1:
flame_status = "Fire is Detected";
Serial.println("Flame Status : "+flame_status);
break;
}
//Gas Detection
if(gas_level > 100){
Serial.println("Gas Status : Gas leakage Detected");
}
else{
exhaust_fan_on = false;
Serial.println("Gas Status : No Gas leakage Detected");
}
//send the sprinkler status
if(flame){
sprinkler_status =
"Sprinkler ON";
Serial.println("Sprinkler Status : "+sprinkler_status);
}
else{
sprinkler_status = "Sprinkler OFF";
Serial.println("Sprinkler Status : "+sprinkler_status);
}
//toggle the fan according to gas
if(gas_level > 100){
exhaust_fan_on = true;
Serial.println("Exhaust fan Status : Fan ON");
}
else{
exhaust_fan_on = false;
Serial.println("Exhaust fan Status : Fan OFF");
}
Serial.println("");
Serial.println("");
Serial.println(" -----##### ----- ");
Serial.println("");
Serial.println("");
delay(1000);
}

```

SIMULATION OUTPUT:



The screenshot shows the Wokwi IoT simulator interface. On the left, a sketch named 'sketch.ino' is displayed with the following code:

```
1 #include <time.h>
2 #include <WiFi.h>
3 #include <PubSubClient.h>
4
5 bool exhaust_fan_on = false;
6 bool sprinkler_on = false;
7 float temperature = 0;
8 int gas_level = 0;
9 int flame = 0;
10 String flame_status = "";
11 String accident_status = "";
12 String sprinkler_status = "";
13 void setup() {
14   Serial.begin(99900);
15 }
16 void loop() {
17   //setting a random seed
18   srand(time(0));
19   //initial variable
20   temperature = random(-
21     20,125);
22   gas_level = random(0,100);
23   int flamereading =
24     random(200,1024);
25   flame =
26     map(flamereading,0,1024,0,
27     2);
28   //set a flame status
29   Serial.print("Temperature : ");
30   Serial.println(temperature);
31   Serial.print("Gas Level : ");
32   Serial.println(gas_level);
33   Serial.print("Flame : ");
34   Serial.println(flame);
35   Serial.print("Flame Status : ");
36   Serial.println(flame_status);
37   Serial.print("Gas Status : ");
38   Serial.println(gas_status);
39   Serial.print("Sprinkler Status : ");
40   Serial.println(sprinkler_status);
41   Serial.print("Exhaust fan Status : ");
42   Serial.println(exhaust_fan_status);
43 }
```

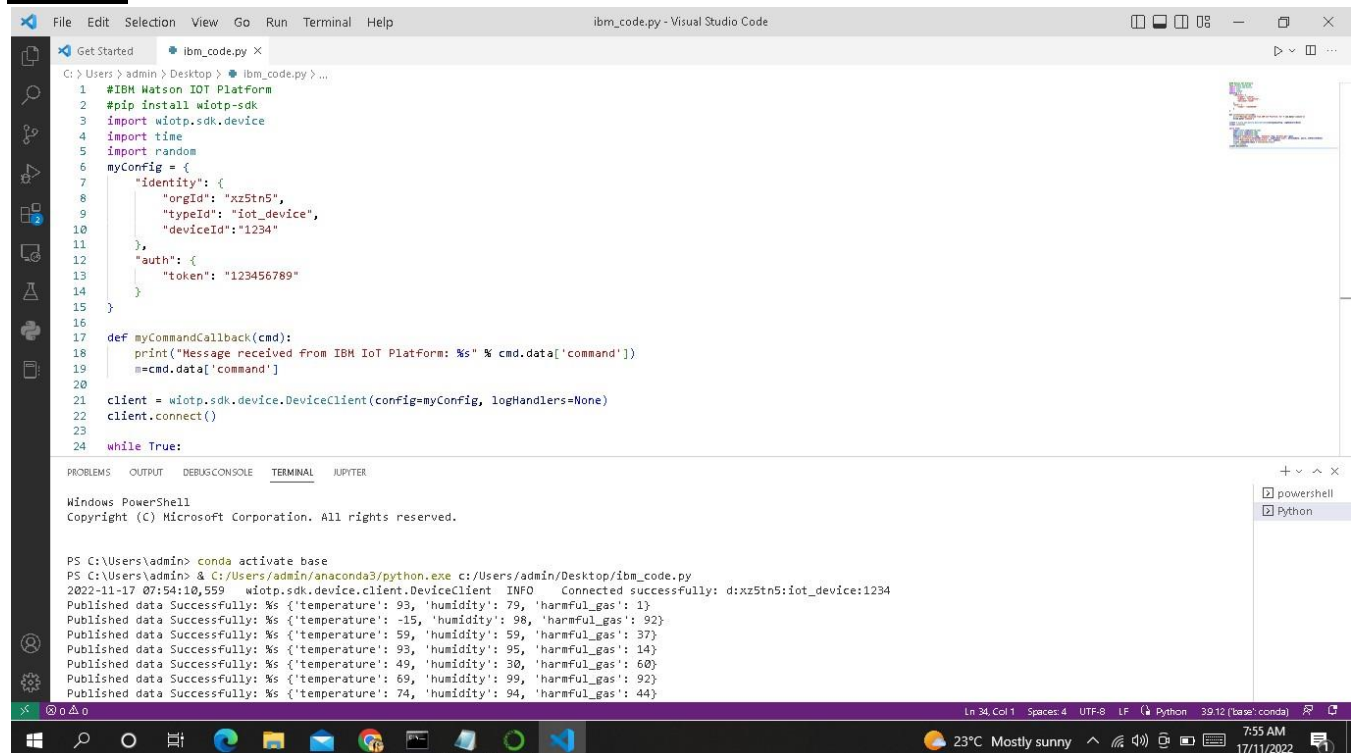
On the right, the 'Simulation' panel shows the output of the sketch. The simulation is running at 71% speed. The output is as follows:

```
Temperature : 67.00
Gas_level : 42
Flame : 0
Flame Status : No Fire
Gas Status : No Gas Leakage Detected
Sprinkler Status : Sprinkler OFF
Exhaust fan Status : Far OFF

-----#####-----
Temperature : 93.00
Gas_level : 27
Flame : 1
Flame Status : Fire is Detected
Gas Status : No Gas leakage Detected
Sprinkler Status : Sprinkler ON
Exhaust fan Status : Far OFF
```

CONNECTING IBM CLOUD USING PYTHON CODE:

CODE:



The screenshot shows a Visual Studio Code editor with a Python script named 'ibm_code.py'. The script is designed to connect to the IBM IoT Platform and publish data. The code is as follows:

```
1 #IBM Watson IOT Platform
2 #pip install wiotp-sdk
3 import wiotp.sdk.device
4 import time
5 import random
6 myConfig = {
7   "identity": {
8     "orgId": "xz5tn5",
9     "typeId": "iot_device",
10    "deviceId": "1234"
11  },
12  "auth": {
13    "token": "123456789"
14  }
15 }
16
17 def myCommandCallback(cmd):
18   print("Message received from IBM IoT Platform: %s" % cmd.data['command'])
19   #cmd.data['command']
20
21 client = wiotp.sdk.device.DeviceClient(config=myConfig, logHandlers=None)
22 client.connect()
23
24 while True:
```

The terminal output shows the successful execution of the script, including the installation of the 'wiotp-sdk' package and the successful connection to the IBM IoT Platform. The output also shows the successful publishing of data to the IoT Platform.

```
PS C:\Users\admin> conda activate base
PS C:\Users\admin> & C:/Users/admin/anaconda3/python.exe c:/Users/admin/Desktop/ibm_code.py
2022-11-17 07:54:10,559 wiotp.sdk.device.client.DeviceClient INFO Connected successfully: d:xz5tn5:iot_device:1234
Published data Successfully: %s {'temperature': 93, 'humidity': 79, 'harmful_gas': 1}
Published data Successfully: %s {'temperature': -15, 'humidity': 98, 'harmful_gas': 92}
Published data Successfully: %s {'temperature': 59, 'humidity': 59, 'harmful_gas': 37}
Published data Successfully: %s {'temperature': 93, 'humidity': 95, 'harmful_gas': 14}
Published data Successfully: %s {'temperature': 49, 'humidity': 30, 'harmful_gas': 60}
Published data Successfully: %s {'temperature': 69, 'humidity': 99, 'harmful_gas': 92}
Published data Successfully: %s {'temperature': 74, 'humidity': 94, 'harmful_gas': 44}
```

OUTPUT IN IBM CLOUD:

The screenshot displays the IBM Watson IoT Platform interface. The top navigation bar includes tabs for 'Browse', 'Action', 'Device Types', and 'Interfaces'. A user profile for 'vijip1107@gmail.com' is visible in the top right. The main content area shows a device with ID '1234' in a 'Connected' state. Below this, the 'Recent Events' tab is active, displaying a table of events. The table has columns for 'Event', 'Value', 'Format', and 'Last Received'. Five events are listed, each with a 'status' event type and a JSON value containing temperature, humidity, and harmful_gas data. The 'Last Received' column indicates that all events were received 'a few seconds ago'. At the bottom of the dashboard, it shows '0 Simulations running'.

Event	Value	Format	Last Received
status	{"temperature":49,"humidity":30,"harmful_gas"...	json	a few seconds ago
status	{"temperature":93,"humidity":95,"harmful_gas"...	json	a few seconds ago
status	{"temperature":59,"humidity":59,"harmful_gas"...	json	a few seconds ago
status	{"temperature":-15,"humidity":98,"harmful_gas"...	json	a few seconds ago
status	{"temperature":93,"humidity":79,"harmful_gas"...	json	a few seconds ago