

# Gas Leakage Monitoring And Alerting System

Develop the Web Application using Node-RED

**Team Members:**

**Team ID:** PNT2022TMID15951

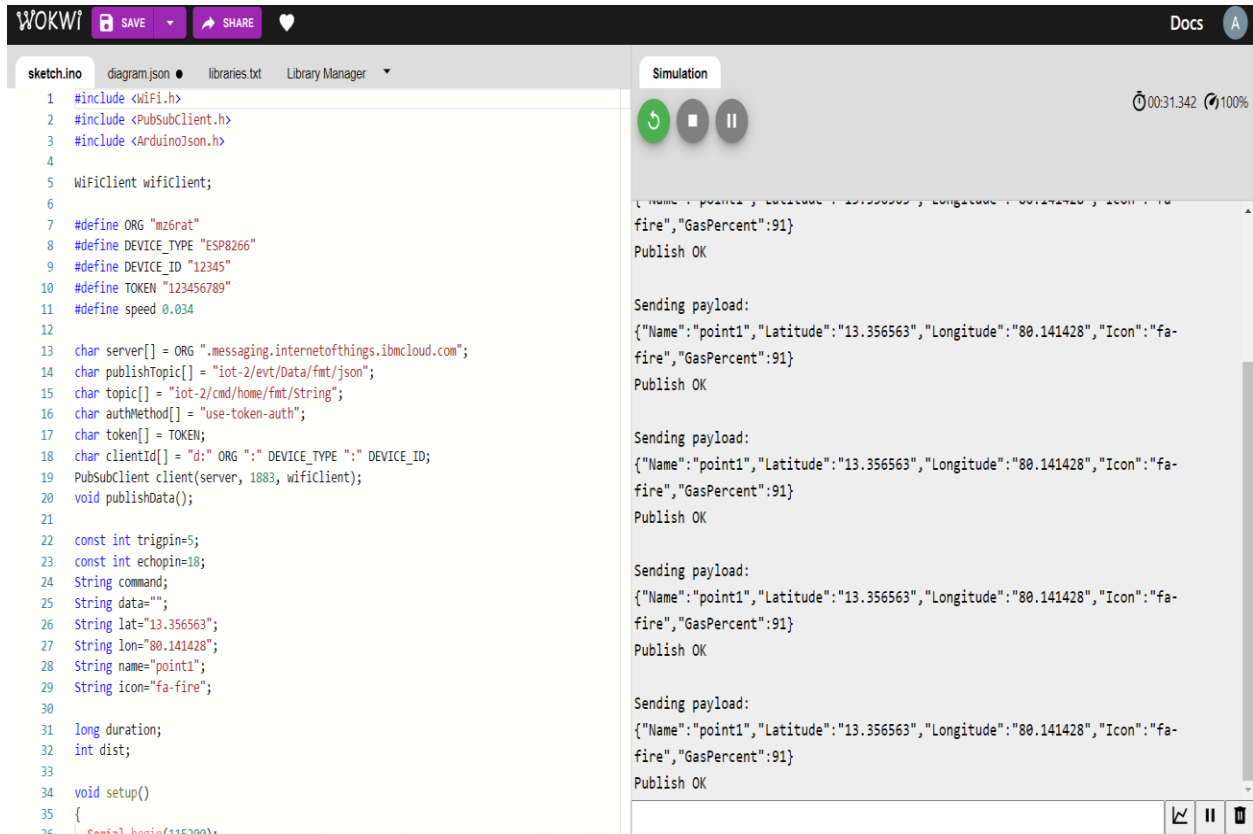
1. Akshaya KS
2. Barani G
3. Ashwini R
4. Abitha J

## **Features of Web UI:**

1. Firstly, connect to the IBM IOT platform to get the location data of the gas leakage.
2. Display the location on the map in the Node-RED UI
3. Send the e-mail to the user with the alert message.

## Step 1:

Find the location of the gas leakage along with gas percentage.



The screenshot displays the Wokwi IoT simulator interface. On the left, the 'sketch.ino' file is open, showing an Arduino sketch that configures an MQTT client to publish data to an IBM Cloud IoT Platform. The sketch includes headers for WiFi, PubSubClient, and ArduinoJson, and defines constants for the organization, device type, device ID, token, and a publish speed. The main loop publishes a JSON payload containing location data (latitude, longitude) and a gas percentage of 91% at a rate of 0.034 seconds. On the right, the 'Simulation' window shows the execution of the sketch. It displays the MQTT client connecting to the broker, sending the payload, and receiving a 'Publish OK' response. The payload is shown as a JSON object: {"Name": "point1", "Latitude": "13.356563", "Longitude": "80.141428", "Icon": "fa-fire", "GasPercent": 91}. The simulation is running at 100% speed.

```
1 #include <WiFi.h>
2 #include <PubSubClient.h>
3 #include <ArduinoJson.h>
4
5 WiFiClient wificlient;
6
7 #define ORG "mz6rat"
8 #define DEVICE_TYPE "ESP8266"
9 #define DEVICE_ID "12345"
10 #define TOKEN "123456789"
11 #define speed 0.034
12
13 char server[] = ORG ".messaging.internetofthings.ibmcloud.com";
14 char publishTopic[] = "iot-2/evt/Data/fmt/json";
15 char topic[] = "iot-2/cmd/home/fmt/String";
16 char authMethod[] = "use-token-auth";
17 char token[] = TOKEN;
18 char clientId[] = "d:" ORG ":" DEVICE_TYPE ":" DEVICE_ID;
19 PubSubClient client(server, 1883, wificlient);
20 void publishData();
21
22 const int trigpin=5;
23 const int echopin=18;
24 String command;
25 String data="";
26 String lat="13.356563";
27 String lon="80.141428";
28 String name="point1";
29 String icon="fa-fire";
30
31 long duration;
32 int dist;
33
34 void setup()
35 {
36   Serial.begin(115200);
```

Simulation

00:31.342 100%

{ "Name": "point1", "Latitude": "13.356563", "Longitude": "80.141428", "Icon": "fa-fire", "GasPercent": 91 }

Publish OK

Sending payload:

{ "Name": "point1", "Latitude": "13.356563", "Longitude": "80.141428", "Icon": "fa-fire", "GasPercent": 91 }

Publish OK

Sending payload:

{ "Name": "point1", "Latitude": "13.356563", "Longitude": "80.141428", "Icon": "fa-fire", "GasPercent": 91 }

Publish OK

Sending payload:

{ "Name": "point1", "Latitude": "13.356563", "Longitude": "80.141428", "Icon": "fa-fire", "GasPercent": 91 }

Publish OK

Sending payload:

{ "Name": "point1", "Latitude": "13.356563", "Longitude": "80.141428", "Icon": "fa-fire", "GasPercent": 91 }

Publish OK

## Step 2:

Output shown on the IBM IOT platform.

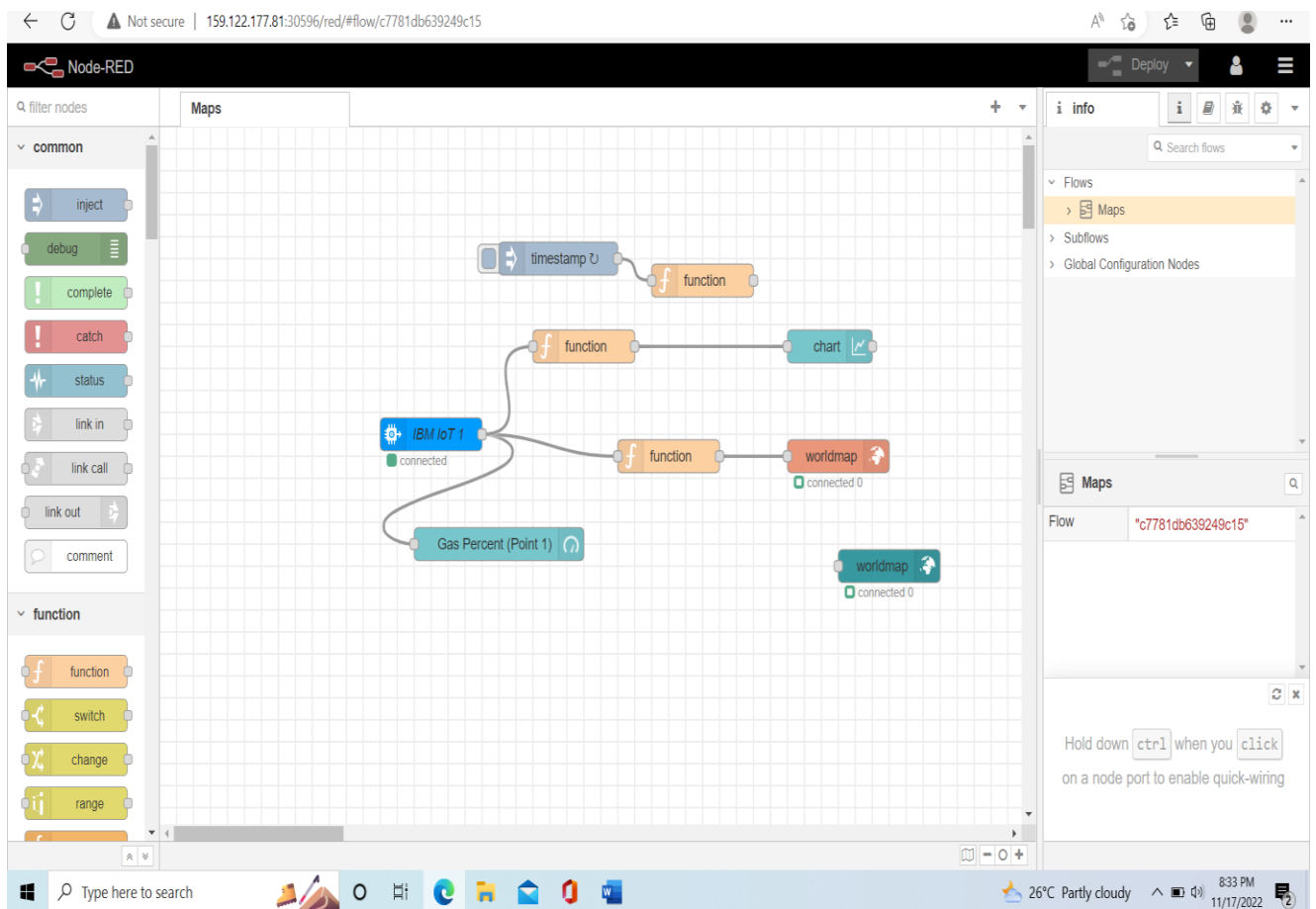
The screenshot displays the IBM Watson IoT Platform interface. The top navigation bar includes 'Browse', 'Action', 'Device Types', and 'Interfaces'. A sidebar on the left contains various icons for navigation. The main content area shows a table of devices. The first device, ID 12345, is 'Connected' and of type 'ESP8266'. It has a 'Recent Events' tab selected, showing a live stream of data. The data events are as follows:

Event	Value	Format	Last Received
Data	{"Name":"point1","Latitude":"13.356563","Longi..."}	json	a few seconds ago
Data	{"Name":"point1","Latitude":"0.000000","Longitu..."}	json	a few seconds ago

The second device, ID 54321, is 'Disconnected' and of type 'arduino'. The interface also includes a search bar, a 'Device Simulator' toggle, and pagination controls at the bottom.

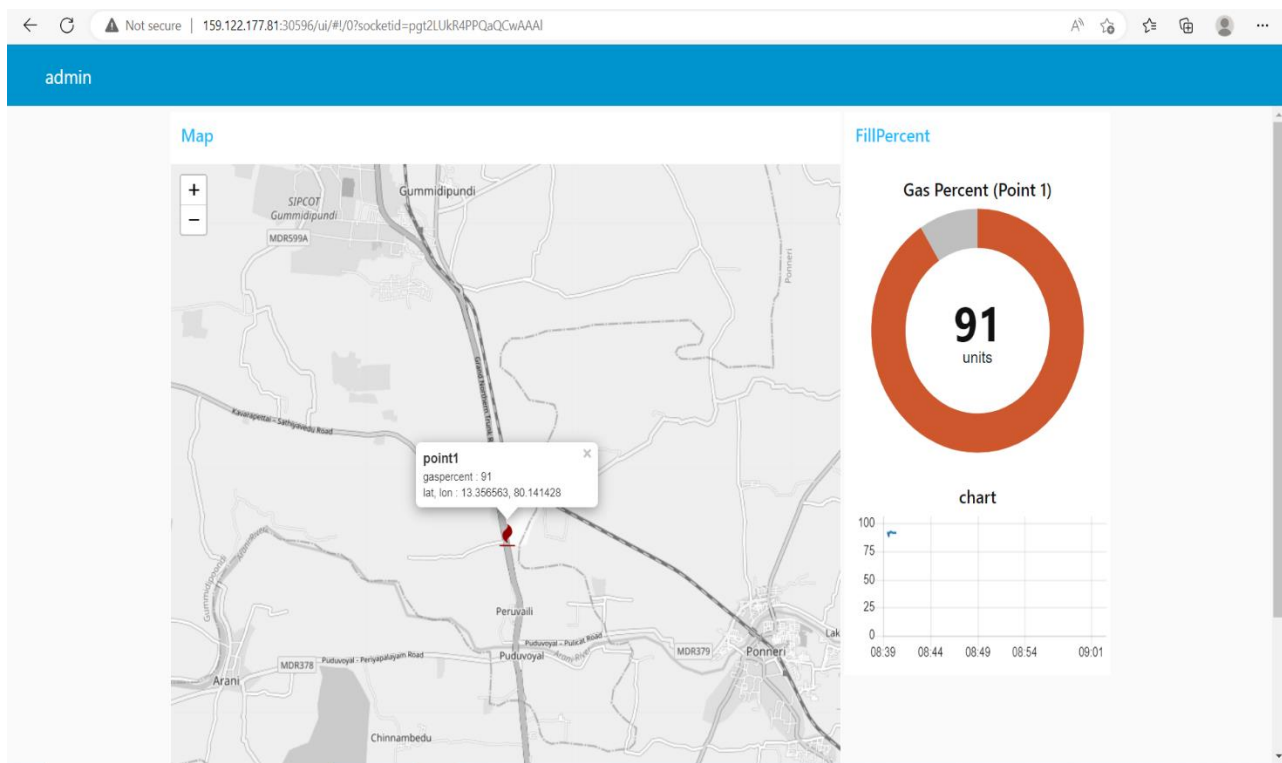
## Step 3:

In this flow we use the IBM IoT node for getting data from IBM Watson IOT platform and changing them into the required format with the help of the function node and passing the values to the Gauge node (UI node) and to the World Map node.



## Step 4:

The below figure shows the location along with the gas percentage and it denotes that the leakage is more.



## Step 5:

Mail come to the user along with the Alert Message. The mail also contains the location and gas percentage.

