

Gas Leakage Monitoring And Alerting System

Develop the Web Application using Node-RED

Team Members:

Team ID: PNT2022TMID15951

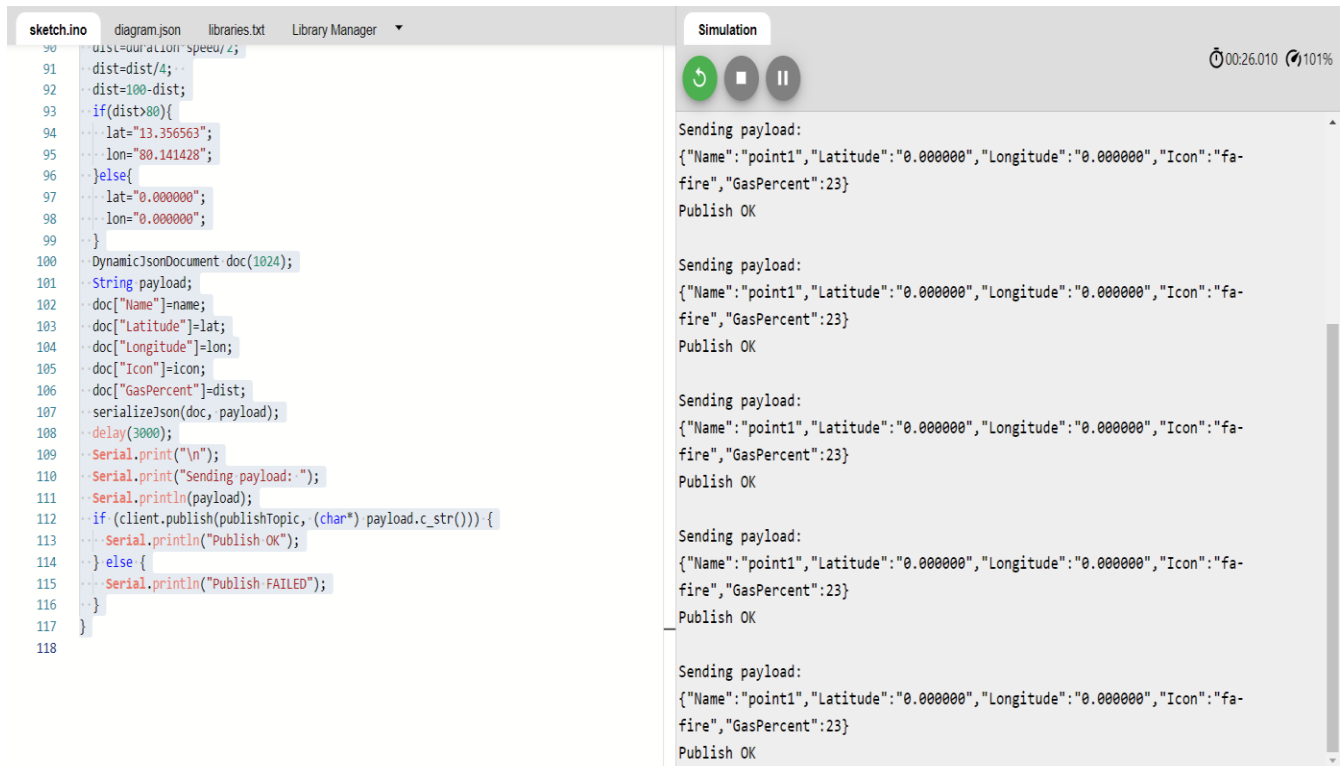
1. Akshaya KS
2. Barani G
3. Ashwini R
4. Abitha J

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.



The screenshot displays the Arduino IDE interface. On the left, the 'sketch.ino' file is open, showing a C++ program that simulates a gas leak detection system. The code includes comments for each line, ranging from 90 to 118. The program calculates a distance, determines a latitude and longitude based on the distance, and then publishes a JSON payload to a topic. The payload contains the name 'point1', latitude '0.000000', longitude '0.000000', icon 'fa-fire', and gas percentage '23'. The code also includes a delay of 3000ms and prints the payload and publish status to the serial monitor.

```
90 // dist=duration*speed/z;  
91 dist=dist/4;  
92 dist=100-dist;  
93 if(dist>80){  
94   lat="13.356563";  
95   lon="80.141428";  
96 }else{  
97   lat="0.000000";  
98   lon="0.000000";  
99 }  
100 DynamicJsonDocument doc(1024);  
101 String payload;  
102 doc["Name"]=name;  
103 doc["Latitude"]=lat;  
104 doc["Longitude"]=lon;  
105 doc["Icon"]=icon;  
106 doc["GasPercent"]=dist;  
107 serializeJson(doc, payload);  
108 delay(3000);  
109 Serial.print("\n");  
110 Serial.print("Sending payload: ");  
111 Serial.println(payload);  
112 if (client.publish(publishTopic, (char*) payload.c_str())) {  
113   Serial.println("Publish OK");  
114 } else {  
115   Serial.println("Publish FAILED");  
116 }  
117 }  
118
```

On the right, the 'Simulation' window shows the output of the code. It displays the JSON payload being sent and the 'Publish OK' status. The simulation is running at 00:26.010 with 101% battery level.

Step 2:

Output shown on the IBM IOT platform.

The screenshot displays the IBM Watson IoT Platform interface. At the top, the header shows the platform name and user information. Below the header, there's a navigation bar with tabs for '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 selected, and its details are shown in a modal window. This modal has tabs for 'Identity', 'Device Information', 'Recent Events', 'State', and 'Logs'. The 'Recent Events' tab is active, showing a list of events with columns for Event, Value, Format, and Last Received. The second device, ID 54321, is shown as disconnected below the modal. At the bottom, there's a pagination bar indicating '1 of 1 page' and '1-2 of 2 items'.

IBM Watson IoT Platform

ksakshya2623@gmail.com
ID: mufarat

Browse Action Device Types Interfaces

Add Device

This table shows a summary of all devices that have been added. It can be filtered, organized, and searched on using different criteria. To get started, you can add devices by using the Add Device button, or by using API.

Search by Device ID

Device Simulator

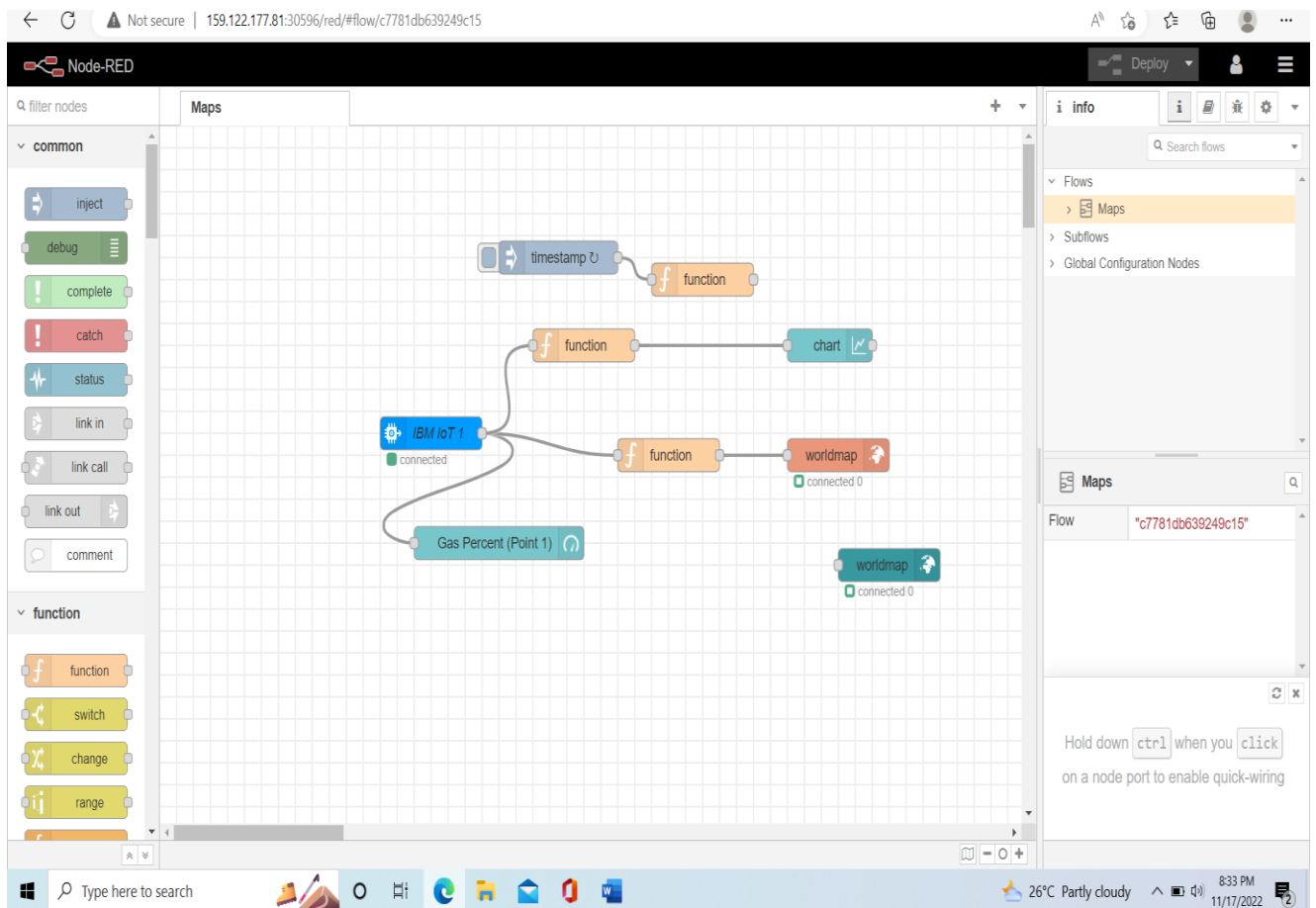
Device ID	Status	Device Type	Class ID	Date Added
12345	Connected	ESP8266	Device	Nov 15, 2022 9:24 PM
54321	Disconnected	arduino	Device	Oct 26, 2022 2:05 PM

Items per page 50 | 1-2 of 2 items

1 of 1 page

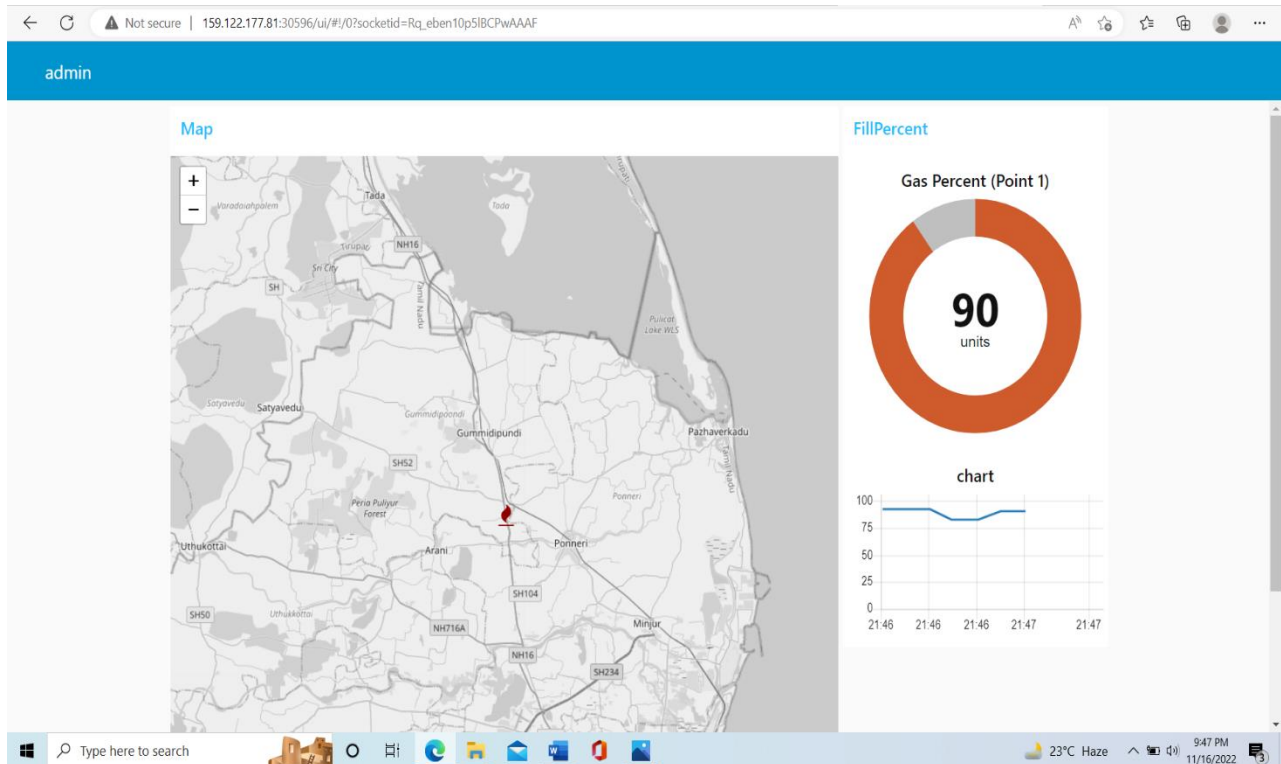
Step 3:

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.

