HINDUSTHAN COLLEGE OF ENGINEERING AND TECHNOLOGY

(AUTONOMOUS)

Team ID	PNT2022TMID10014
Project Name	Project - IOT Gas Leakage Monitoring and Alerting System.

PHASE 1:

In this phase we have developed a python code to generate random sensor data and publish that data to the IBM internet of things platform using a python package called ibmiotf. These data will be published to the respected device in that platform.

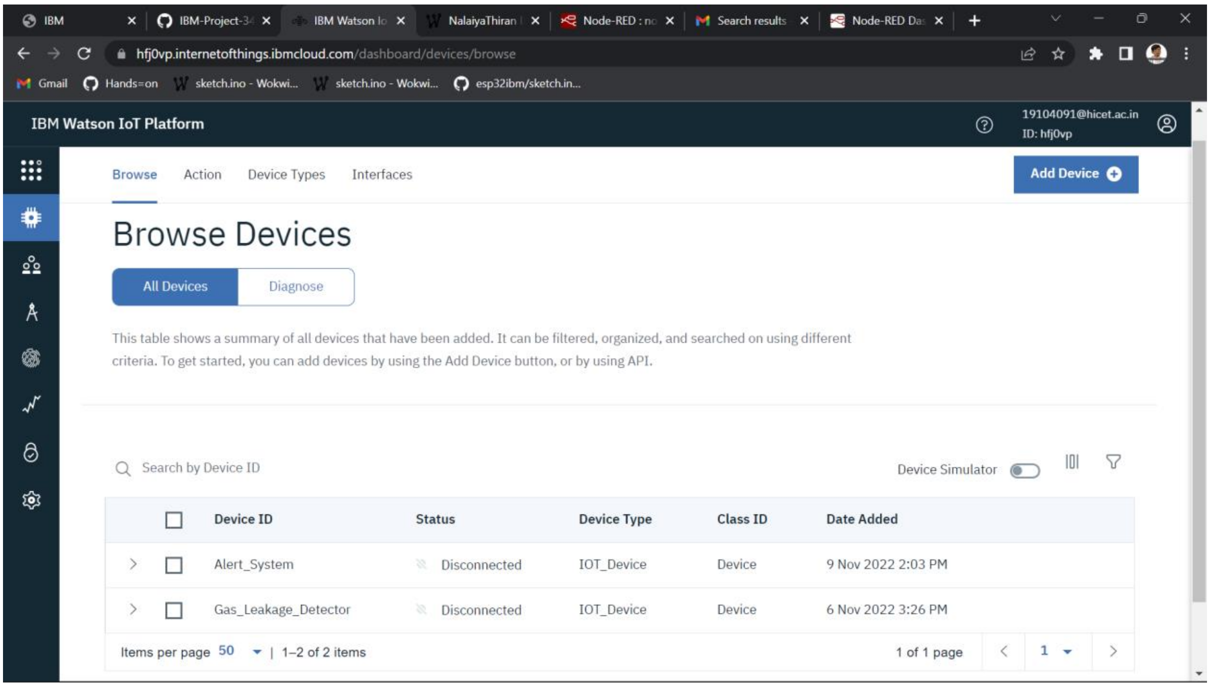
PYTHON CODE:

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

# IBM Watson Device Credentials
organization = "hfj0vp"  # Organization ID
deviceType = "IOT_Device"  # Device type
deviceId = "Gas_Leakage_Detector"  # Device id
authMethod = "token"
authToken = " "  # Authentication token should be given here. It is not provided
here since it is a demo and for security reasons.

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:
    # Ransom sensor data generation
    T = random.randint(-40, 80)
    H = random.randint(0, 100)
    G = random.randint(100, 10000)
    A = "OFF"  # Alert flag

    if G >= 1000:  # We can add as many conditions here to check other sensor
data
        A = "ON"

    else:
        A = "OFF"

    # Send sensor data to IBM Watson
    data = {'temperature': T, 'humidity': H, 'gas': G, 'alert': A}

    # print data
    def myOnPublishCallback():
        print("Published Temperature = %s C" % T, "Humidity = %s %%" % H,
"Gas level = %s ppm" % G, "to IBM Watson")


    success = deviceCli.publishEvent("event", "json", data, qos=0,
on_publish=myOnPublishCallback)
    if not success:
        print("Not connected to IoTF")
    time.sleep(5)


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

PHASE 2:

In this phase we have created IBM Watson internet of things platform cloud services and 2 devices, one for publishing sensor data another one for subscribing to alert system.

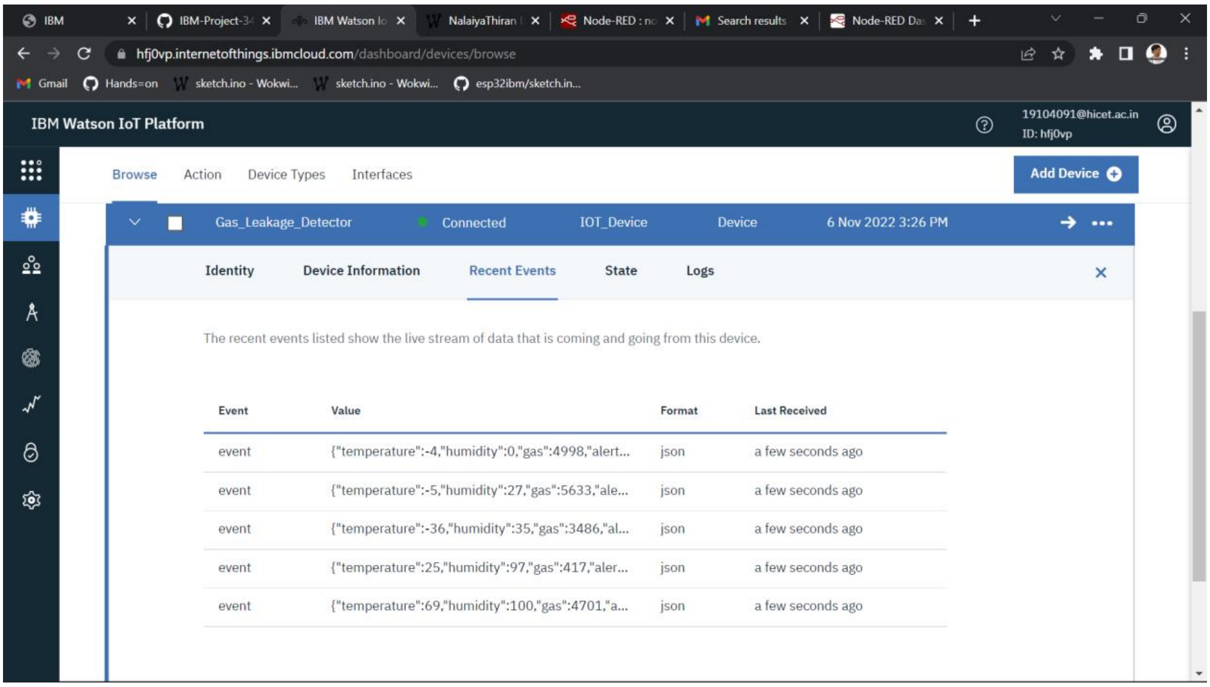


Browse Devices

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.

Device ID	Status	Device Type	Class ID	Date Added
Alert_System	Disconnected	IOT_Device	Device	9 Nov 2022 2:03 PM
Gas_Leakage_Detector	Disconnected	IOT_Device	Device	6 Nov 2022 3:26 PM

Here the random sensor data are successfully published in the json format from the python code that we have developed during the previous phase.



The recent events listed show the live stream of data that is coming and going from this device.

Event	Value	Format	Last Received
event	{"temperature":-4,"humidity":0,"gas":4998,"alert...	json	a few seconds ago
event	{"temperature":-5,"humidity":27,"gas":5633,"ale...	json	a few seconds ago
event	{"temperature":-36,"humidity":35,"gas":3486,"al...	json	a few seconds ago
event	{"temperature":25,"humidity":97,"gas":417,"aler...	json	a few seconds ago
event	{"temperature":69,"humidity":100,"gas":4701,"a...	json	a few seconds ago

PHASE 3:

In this phase we have created and configured the node red services and developed a Web UI dashboard for the users to monitor the sensor and to toggle the state of the alarm. The data from the IBM Watson IOT platform are sent to this node red application and an email is sent to the admins every 5 minutes with the node red UI dashboard link if the gas leakage is detected and the alarm is automatically triggered. Using that link the admin can monitor the gas levels and can toggle the alarm switch from any device using the internet.

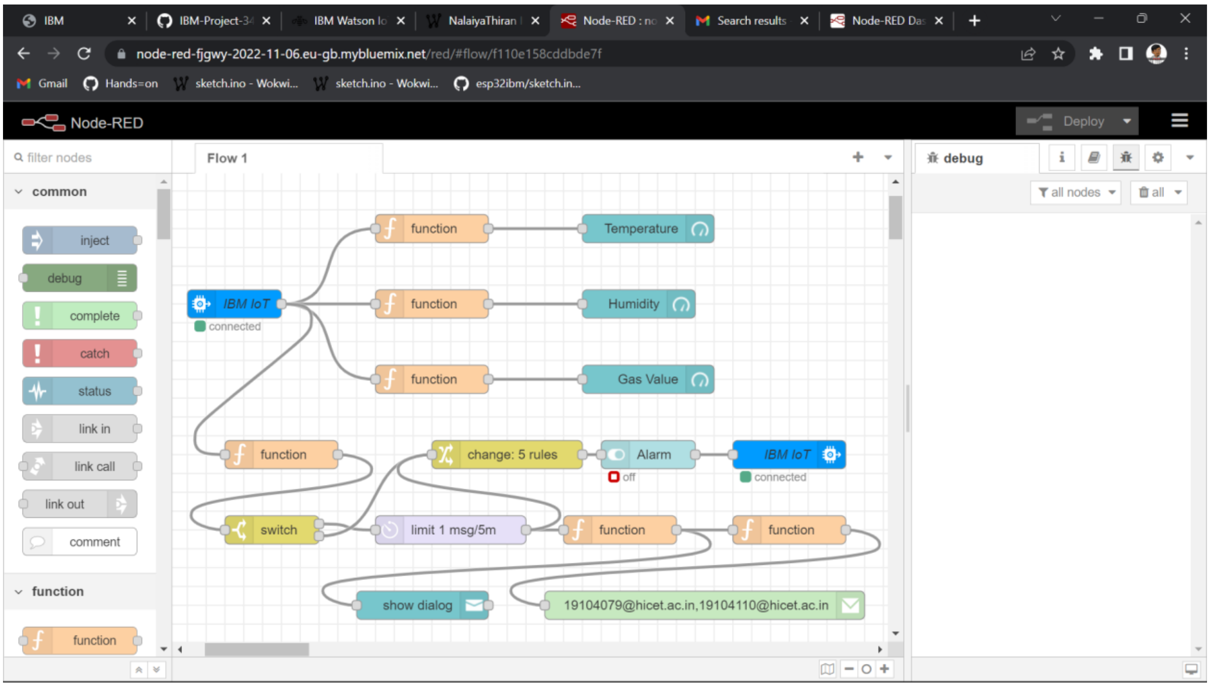
NODE RED FLOW LINK:

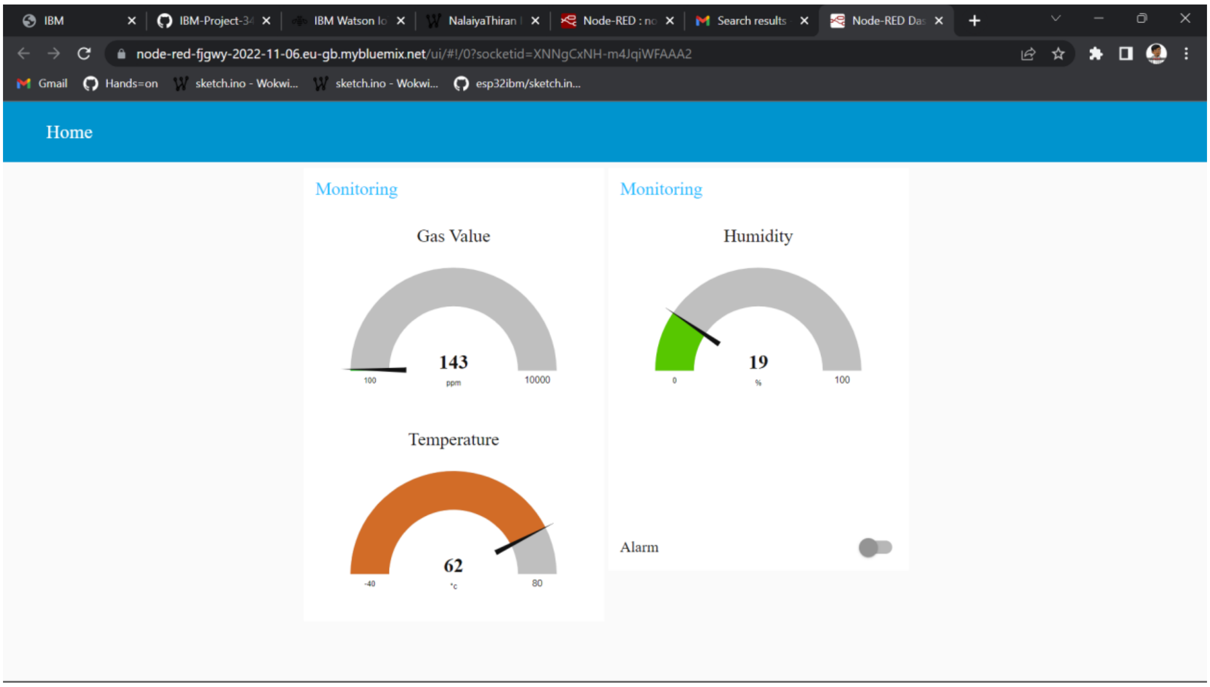
https://node-red-fjgwy-2022-11-06.eu-gb.mybluemix.net/red/#flow/f110e158cddbde7f

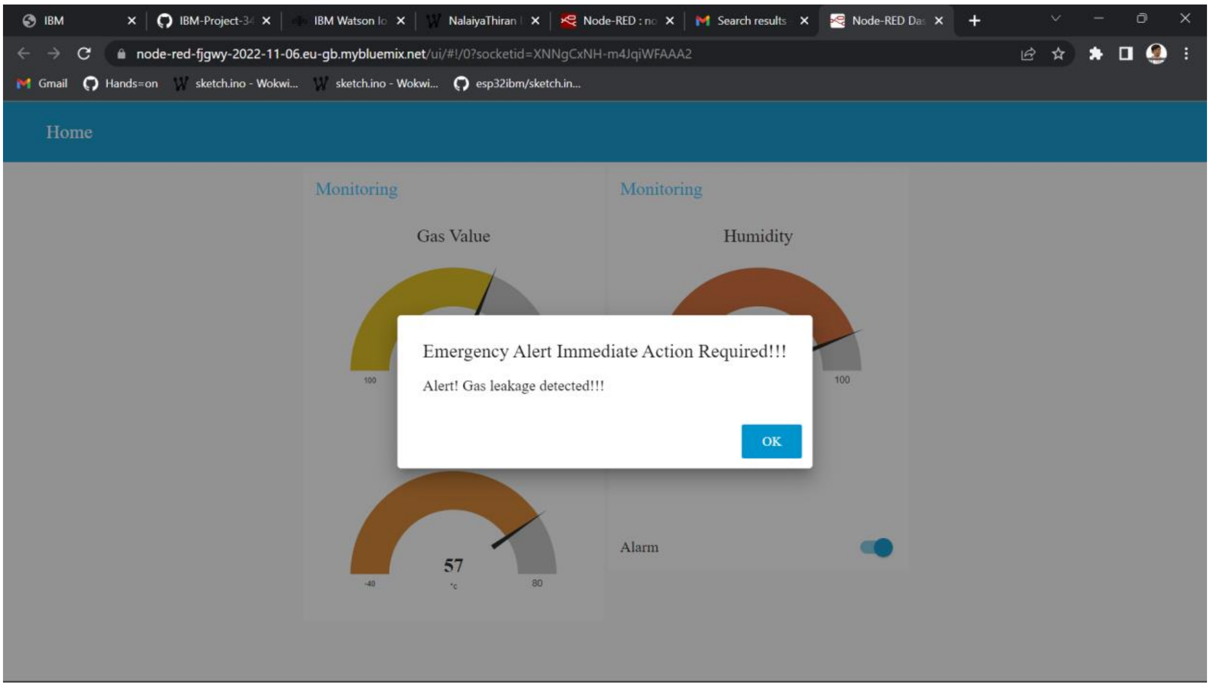
NODE RED UI DASHBOARD LINK:

https://node-red-fjgwy-2022-11-06.eu-gb.mybluemix.net/ui/#!/0?socketid=XNNgCxNH-m4JqiWFAAA2

SCREENSHOTS:

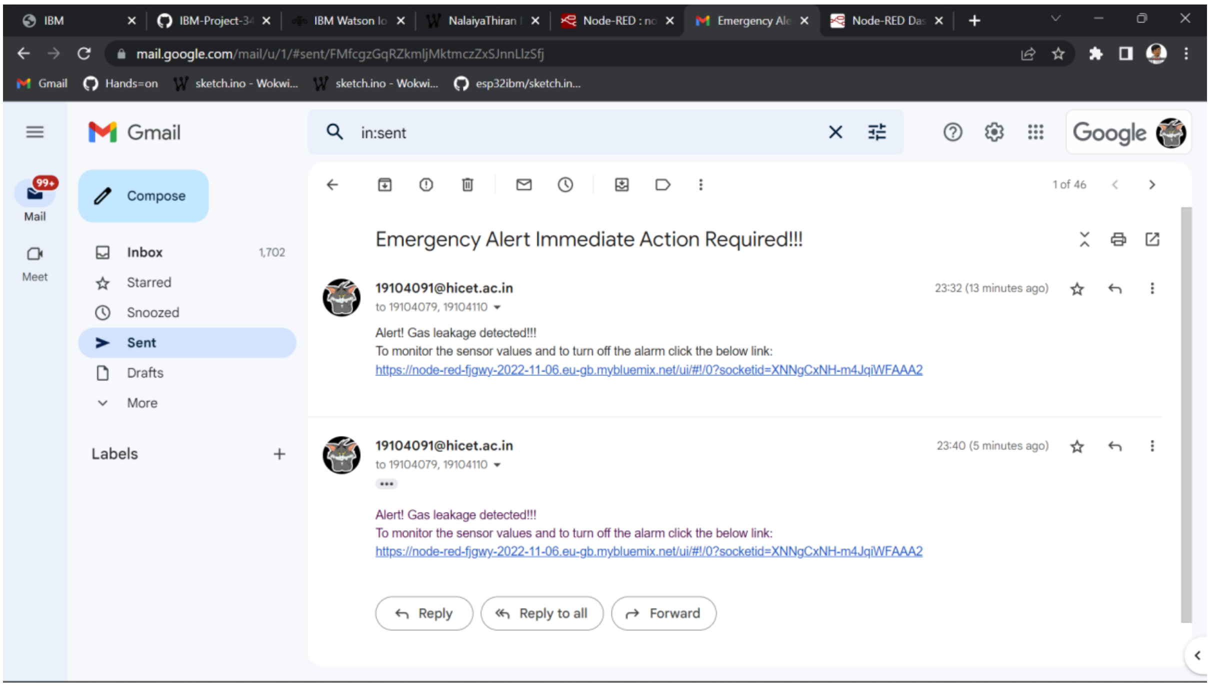





node-red-fjgwy-2022-11-06.eu-gb.mybluemix.net/ui/#!/0?socketid=XNNgCxNH-m4JqiWFAAA2
Home
Monitoring
Gas Value
Monitoring
Humidity
100
100
57
°c
-40
80
Alarm
OK

Emergency Alert Immediate Action Required!!!

Alert! Gas leakage detected!!!



Emergency Alert Immediate Action Required!!!

19104091@hicet.ac.in
to 19104079, 19104110
23:32 (13 minutes ago)

Alert! Gas leakage detected!!!
To monitor the sensor values and to turn off the alarm click the below link:
https://node-red-fjgwy-2022-11-06.eu-gb.mybluemix.net/ui/#!/0?socketid=XNNgCxNH-m4JqiWFAAA2

19104091@hicet.ac.in
to 19104079, 19104110
23:40 (5 minutes ago)

Alert! Gas leakage detected!!!
To monitor the sensor values and to turn off the alarm click the below link:
https://node-red-fjgwy-2022-11-06.eu-gb.mybluemix.net/ui/#!/0?socketid=XNNgCxNH-m4JqiWFAAA2

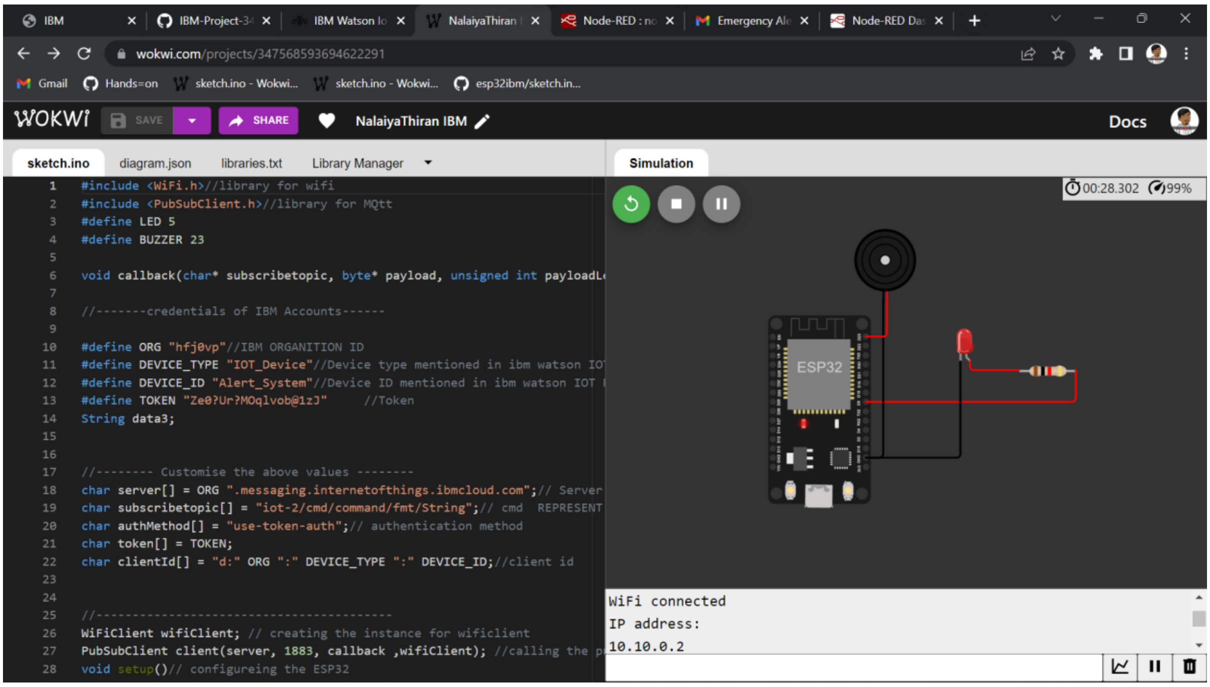
PHASE 4:

In this phase we have developed an alarm system simulation using a led, buzzer and ESP32 microcontroller. The subscribe model device named Alert_System in IBM Watson IOT platform is connected to this simulation using device credentials. Thus the alarm gets toggles ON automatically when a gas leakage is detected. However, this alarm can be toggled ON and OFF manually from the Node Red Web Application dashboard by the admins.

WOKWI WEBSITE LINK:

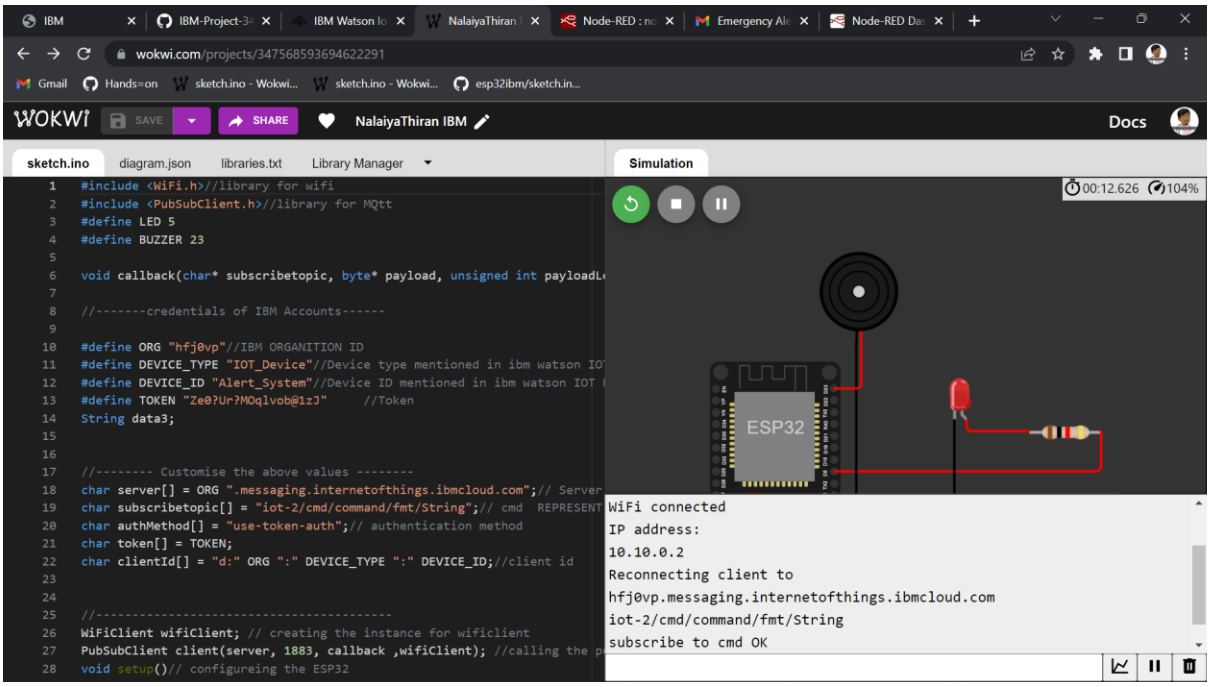
https://wokwi.com/projects/347568593694622291

SCREENSHOTS:



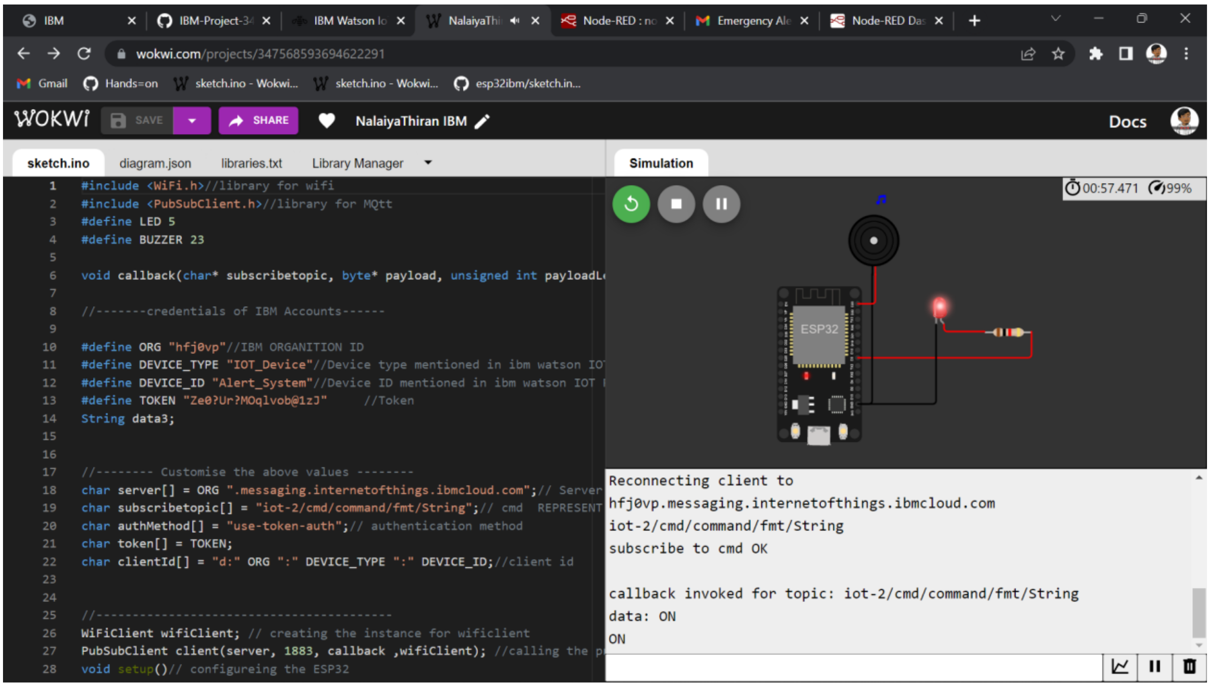
```
#include <WiFi.h>//library for wifi
#include <PubSubClient.h>//library for MQtt
#define LED 5
#define BUZZER 23

void callback(char* subscribetopic, byte* payload, unsigned int payloadL

//-------credentials of IBM Accounts------

#define ORG "hfj0vp"//IBM ORGANITION ID
#define DEVICE_TYPE "IOT_Device"//Device type mentioned in ibm watson IO
#define DEVICE_ID "Alert_System"//Device ID mentioned in ibm watson IOT
#define TOKEN "Ze0?Ur?MOqlvob@1zJ"     //Token
String data3;


//-------- Customise the above values --------
char server[] = ORG ".messaging.internetofthings.ibmcloud.com";// Server
char subscribetopic[] = "iot-2/cmd/command/fmt/String";// cmd  REPRESENT
char authMethod[] = "use-token-auth";// authentication method
char token[] = TOKEN;
char clientId[] = "d:" ORG ":" DEVICE_TYPE ":" DEVICE_ID;//client id


//-----------------------------------------
WiFiClient wifiClient; // creating the instance for wificlient
PubSubClient client(server, 1883, callback ,wifiClient); //calling the p
void setup()// configureing the ESP32
```

```
WiFi connected
IP address:
10.10.0.2
```



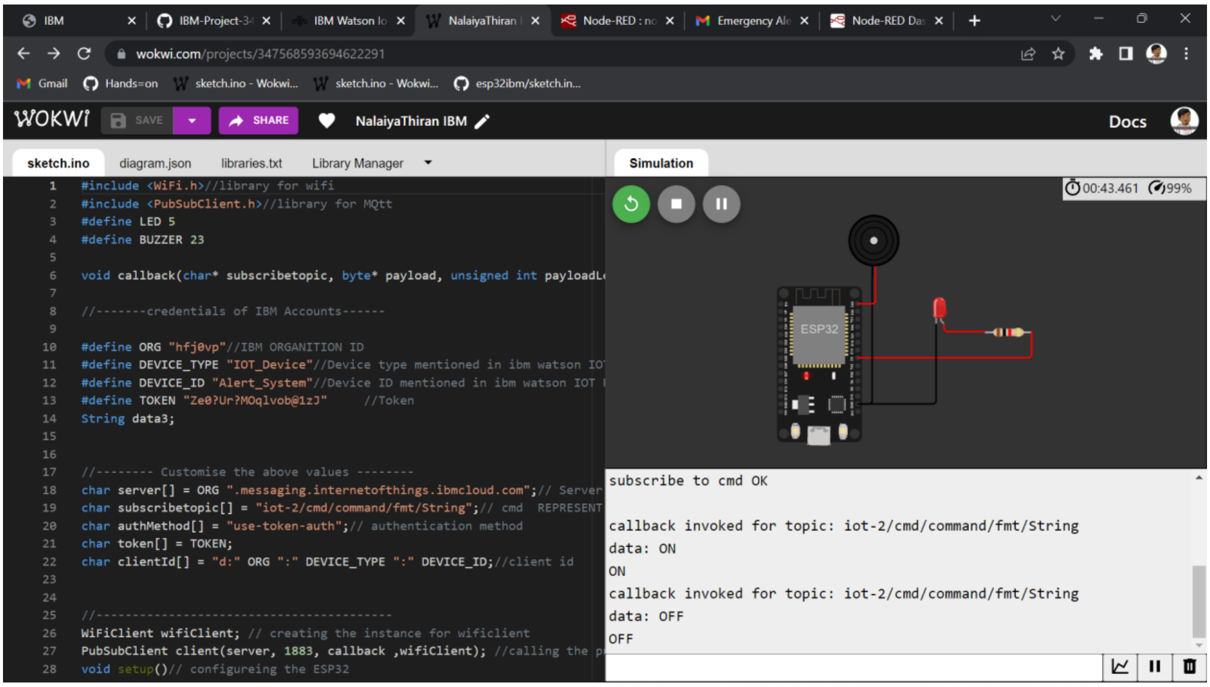
```
#include <WiFi.h>//library for wifi
#include <PubSubClient.h>//library for MQtt
#define LED 5
#define BUZZER 23

void callback(char* subscribetopic, byte* payload, unsigned int payloadL

//-------credentials of IBM Accounts------

#define ORG "hfj0vp"//IBM ORGANITION ID
#define DEVICE_TYPE "IOT_Device"//Device type mentioned in ibm watson IO
#define DEVICE_ID "Alert_System"//Device ID mentioned in ibm watson IOT
#define TOKEN "Ze0?Ur?MOqlvob@1zJ"     //Token
String data3;


//-------- Customise the above values --------
char server[] = ORG ".messaging.internetofthings.ibmcloud.com";// Server
char subscribetopic[] = "iot-2/cmd/command/fmt/String";// cmd  REPRESENT
char authMethod[] = "use-token-auth";// authentication method
char token[] = TOKEN;
char clientId[] = "d:" ORG ":" DEVICE_TYPE ":" DEVICE_ID;//client id


//-----------------------------------------
WiFiClient wifiClient; // creating the instance for wificlient
PubSubClient client(server, 1883, callback ,wifiClient); //calling the p
void setup()// configureing the ESP32
```

```
WiFi connected
IP address:
10.10.0.2
Reconnecting client to
hfj0vp.messaging.internetofthings.ibmcloud.com
iot-2/cmd/command/fmt/String
subscribe to cmd OK
```



```
#include <WiFi.h>//library for wifi
#include <PubSubClient.h>//library for MQtt
#define LED 5
#define BUZZER 23

void callback(char* subscribetopic, byte* payload, unsigned int payloadL

//-------credentials of IBM Accounts------

#define ORG "hfj0vp"//IBM ORGANITION ID
#define DEVICE_TYPE "IOT_Device"//Device type mentioned in ibm watson IO
#define DEVICE_ID "Alert_System"//Device ID mentioned in ibm watson IOT 
#define TOKEN "Ze0?Ur?MOqlvob@1zJ"     //Token
String data3;


//-------- Customise the above values --------
char server[] = ORG ".messaging.internetofthings.ibmcloud.com";// Server
char subscribetopic[] = "iot-2/cmd/command/fmt/String";// cmd  REPRESENT
char authMethod[] = "use-token-auth";// authentication method
char token[] = TOKEN;
char clientId[] = "d:" ORG ":" DEVICE_TYPE ":" DEVICE_ID;//client id


//-----------------------------------------
WiFiClient wifiClient; // creating the instance for wificlient
PubSubClient client(server, 1883, callback ,wifiClient); //calling the p
void setup()// configureing the ESP32
```

```
Reconnecting client to
hfj0vp.messaging.internetofthings.ibmcloud.com
iot-2/cmd/command/fmt/String
subscribe to cmd OK

callback invoked for topic: iot-2/cmd/command/fmt/String
data: ON
ON
```



```
#include <WiFi.h>//library for wifi
#include <PubSubClient.h>//library for MQtt
#define LED 5
#define BUZZER 23

void callback(char* subscribetopic, byte* payload, unsigned int payloadL

//-------credentials of IBM Accounts------

#define ORG "hfj0vp"//IBM ORGANITION ID
#define DEVICE_TYPE "IOT_Device"//Device type mentioned in ibm watson IO
#define DEVICE_ID "Alert_System"//Device ID mentioned in ibm watson IOT 
#define TOKEN "Ze0?Ur?MOqlvob@1zJ"     //Token
String data3;


//-------- Customise the above values --------
char server[] = ORG ".messaging.internetofthings.ibmcloud.com";// Server
char subscribetopic[] = "iot-2/cmd/command/fmt/String";// cmd  REPRESENT
char authMethod[] = "use-token-auth";// authentication method
char token[] = TOKEN;
char clientId[] = "d:" ORG ":" DEVICE_TYPE ":" DEVICE_ID;//client id


//-----------------------------------------
WiFiClient wifiClient; // creating the instance for wificlient
PubSubClient client(server, 1883, callback ,wifiClient); //calling the p
void setup()// configureing the ESP32
```

```
subscribe to cmd OK

callback invoked for topic: iot-2/cmd/command/fmt/String
data: ON
ON
callback invoked for topic: iot-2/cmd/command/fmt/String
data: OFF
OFF
```