

PROJECT DEVELOPMENT PHASE

Delivery Of Sprint-3

Date	12 November 2022
Team ID	PNT2022TMID53925
Project Name	Gas Leakage Monitoring and Alerting Systems

SPRINT 3:

Functional Requirement

Customization of Coding and Code Testing.

User Story

--> As a designer, I Develop a code with related Libraries.

--> As a designer, I can create an overall programming with testing of code.

Procedure

The code is generated and the output of parameters such as temperature,pressure and Gas level is displayed in ibm Watson platform.

Required components :

1.Wokwi Simulator.

2.IBM Watson Platform.

Code:

```
#include <WiFi.h>//library for wifi
#include <PubSubClient.h>//library for MQTT
#include <LiquidCrystal.h>
#include <ESP32Servo.h>
#include "DHT.h"// Library for dht11
#define DHTPIN 15 // what pin we're connected to
```

```

#define DHTTYPE DHT22    // define type of sensor DHT 11

LiquidCrystal
lcd(2,4,19,21,12,14); int GreenLED
= 18; int RedLED = 5; int
BUZZER_PIN = 13; const int
servoPin = 22; String data3; int
g;

Servo door;

int pos;

DHT dht (DHTPIN, DHTTYPE); // creating the instance by passing pin and typtr of dht connected

void callback(char* subscribetopic, byte* payload, unsigned int payloadLength);

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

#define ORG "u9pz01"//IBM ORGANITION ID

#define DEVICE_TYPE "gassense"//Device type mentioned in ibm watson IOT Platform

#define DEVICE_ID "sensor"//Device ID mentioned in ibm watson IOT Platform

```

```

#define TOKEN "12345678"      //Token

float h, t;

//----- Customise the above values -----char server[] = ORG ".messaging.internetofthings.ibmcloud.com";//
Server Name char publishTopic[] = "iot-2/evt/Data/fmt/json";// topic name and type of event perform and format
in which data

to be send char subscribetopic[] = "iot-2/cmd/command/fmt/String";// cmd REPRESENT command type AND COMMAND IS
TEST OF
FORMAT STRING 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 predefined client id by passing parameter
like server id,portand wificredential

void setup() {
  Serial.begin(115200);
  dht.begin();

```

```
pinMode (GreenLED, OUTPUT);

pinMode (RedLED, OUTPUT);

pinMode (BUZZER_PIN, OUTPUT);

lcd.begin(16,2);

lcd.setCursor(1,0);

lcd.print(("GAS

DETECTION"));

door.attach(servoPin, 500,

2400); Serial.println();

wificonnect();

mqttconnect(); } void loop()

{ g =random(0,100);

Serial.print("Gas Level in

Percentage :");

Serial.println(g); h =

dht.readHumidity(); t =

dht.readTemperature();

Serial.print("temp:");

Serial.println(t);

Serial.print("Humid:");

Serial.println(h);

condition(g); PublishData(t,

h ,g); delay(1000); if
```

```

        (!client.loop()) {
            mqttconnect();
        }

        delay(5000);
    }

    //                Condition for buzzer

void myTone( int pin)
{
    ledcAttachPin(pin, 0);           // pin, channel
    ledcWriteNote(0, NOTE_F, 4);     // channel, frequency, octave
}

void myNoTone( int pin)
{ ledcDetachPin(pin);
}

//                Condition for Gaslevel

void condition(int g)
{ if(g > 50)
    { myTone(BUZZER_PIN);
      digitalWrite(RedLED, HIGH);
      digitalWrite(GreenLED, LOW);
      delay(500);
    }
}

```

```

    lcd.setCursor(0,1);

    lcd.print("ALERT!!");

    delay(300);

    lcd.setCursor(0,1);

    lcd.print("HAZARDOUS

    LEVEL!");

} else { myNoTone(BUZZER_PIN);

digitalWrite(RedLED, LOW);

digitalWrite(GreenLED, HIGH);

delay(500); lcd.setCursor(0,1);

lcd.print("NORMAL GAS LEVEL");

}

} /*.....retrieving to Cloud.....*/

void PublishData(float temp, float Humid, int Gas) {

    mqttconnect();//function call for connecting to ibm

    /* creating the String in in form JSON to update the data to ibm

        cloud

    */

    String payload =

    "{\"temp\":\""; payload +=

    temp; payload += ","

```

```
"\"Humid\":"; payload +=
Humid; payload += ","
"\"Gas\":"; payload += Gas;
payload += "}";

Serial.print("Sending payload: "); Serial.println(payload);

if (client.publish(publishTopic, (char*) payload.c_str()))

{

    Serial.println("Publish ok");// if it sucessfully upload data on the cloud then it will print publish ok in Serial
monitor or else it will print publish failed
}
else {

    Serial.println("Publish failed");
} } void mqttconnect() {

if (!client.connected()) {

    Serial.print("Reconnecting client to ");
    Serial.println(server); while
```

```

    (!!!client.connect(clientId, authMethod, token)) {
        Serial.print("."); delay(500);
    }

    initManagedDevice();

    Serial.println();
} } void wificonnect() //function defination for
wificonnect
{
    Serial.println();

    Serial.print("Connecting to ");

    WiFi.begin("Wokwi-GUEST", "", 6); //passing the wifi credentials to establish the
    connection while (WiFi.status() != WL_CONNECTED) { delay(500); Serial.print(".");
}

    Serial.println("");

    Serial.println("WiFi connected");

    Serial.println("IP address: ");

    Serial.println(WiFi.localIP());
}

void initManagedDevice() { if
    (client.subscribe(subscribetopic)) {

        Serial.println((subscribetopic));
    }
}

```



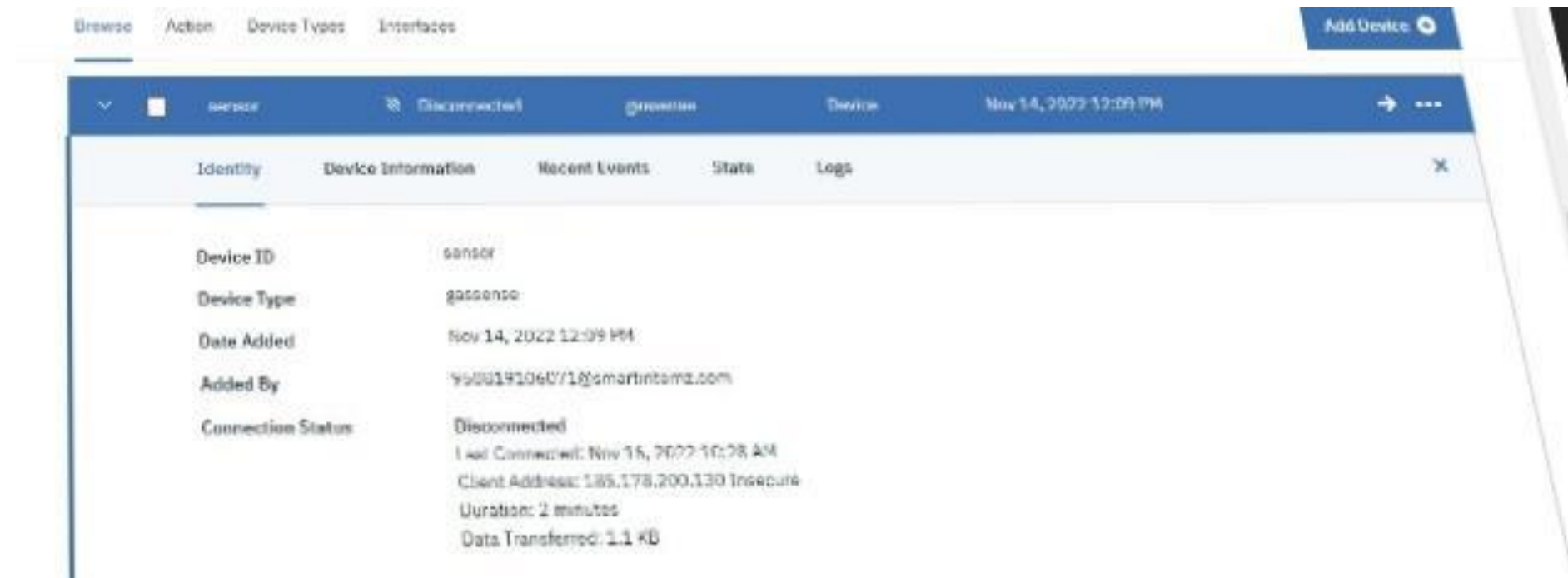
```
    Serial.println("subscribe to cmd OK");
} else {
    Serial.println("subscribe to cmd FAILED");
}
}
```

```
void callback(char* subscribetopic, byte* payload, unsigned int payloadLength)
{
```

```
    Serial.print("callback invoked for topic: ");
    Serial.println(subscribetopic); for (int i =
0; i < payloadLength; i++) {
    //Serial.print((char)payload[i]); data3 +=
(char)payload[i];
} Serial.println("data: "+
data3); if(data3=="dooropen") {
    Serial.println(data3); pos =
180; //open the door
    door.write(pos);
}
else
{
```

```
Serial.println(data3); pos =  
0; // closing the door  
door.write(pos);  
}  
data3="";  
}
```

IBM Watson Platform:



The screenshot displays the IBM Watson Platform interface. At the top, there are tabs for 'Browse', 'Action', 'Device Types', and 'Interfaces'. A blue 'Add Device' button is located in the top right corner. Below these tabs is a header bar with a dropdown menu, a status indicator 'Disconnected', a 'Properties' link, a 'Device' label, and a timestamp 'Nov 14, 2022 12:09 PM'. A right arrow and a three-dot menu are also present. Below the header bar is a table with columns: 'Identity', 'Device Information', 'Recent Events', 'State', and 'Logs'. The 'Identity' column is selected. The table contains one row with the following data:

Identity	Device Information	Recent Events	State	Logs
Device ID	sensor			
Device Type	gasense			
Date Added	Nov 14, 2022 12:09 PM			
Added By	9500191060/1@smartintama.com			
Connection Status	Disconnected			

Below the 'Connection Status' row, there is additional information: 'Last Connected: Nov 15, 2022 10:28 AM', 'Client Address: 185.178.200.130 Insecure', 'Duration: 2 minutes', and 'Data Transferred: 1.1 KB'.



GAS SENSOR



Line chart



5 minutes ▾

temp Humid

now



Gauge

