

# WOKWI SIMULATION

TEAMID:PNT2022TMID49056

## 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
typr of dht connected
void callback(char* subscribetopic, byte* payload, unsigned int
payloadLength);
//-----credentials of IBM Accounts-----
#define ORG "domlyv"//IBM ORGANITION ID
#define DEVICE_TYPE "abcd"//Device type mentioned in ibm watson IOT
Platform
#define DEVICE_ID "12"//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 datato 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="";
}

```

## WOKWI OUTPUT:

## WOKWI TO IBM WATSON IOT PLATFORM:

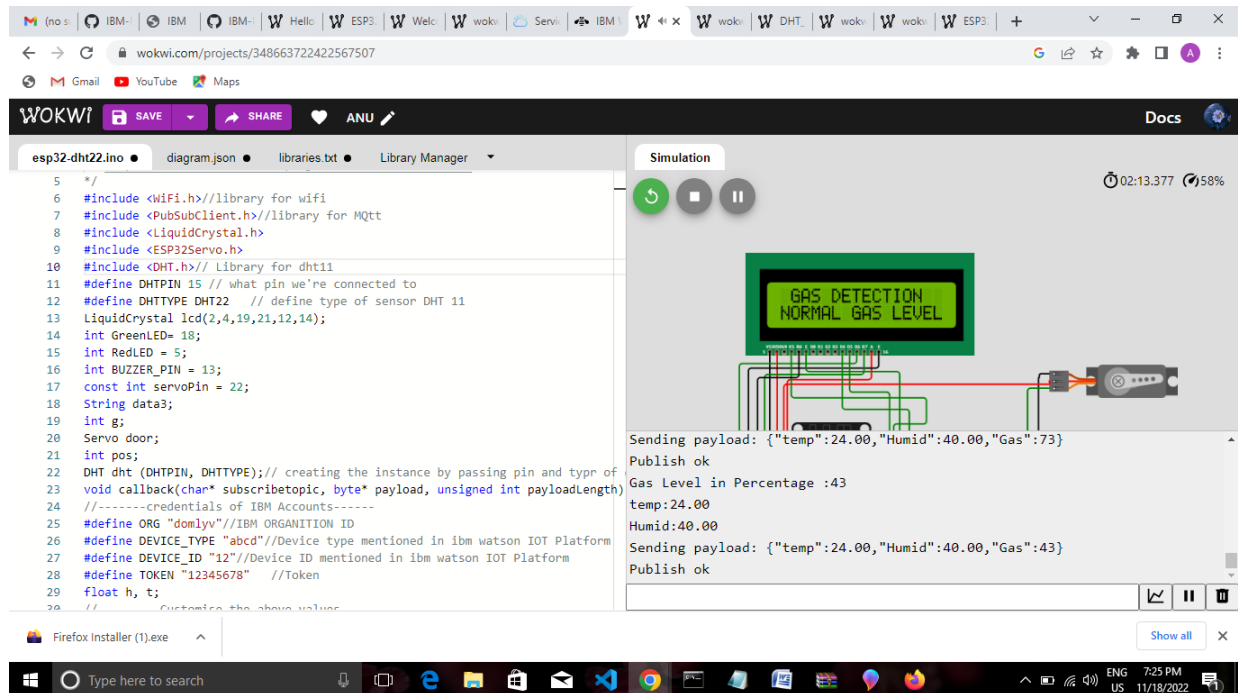
The screenshot displays the Wokwi IDE interface. On the left, the code editor shows a C++ program for an ESP32. The code includes libraries for WiFi, PubSubClient, LiquidCrystal, and ESP32Servo. It defines pins for a DHT22 sensor and a servo motor. The main function sets up the sensor and servo, and implements a callback function for the 'cmd' topic. The simulation window on the right shows a green LCD screen displaying 'GAS DETECTION' and 'NORMAL GAS LEVEL'. Below the screen, there's a servo motor connected to a buzzer. The status bar at the bottom shows 'Connecting to ...', 'WiFi connected', and 'IP address: 10.10.0.2'.

## WOKWI LINK:

<https://wokwi.com/projects/348663722422567507>

## GAS DETECTION:

## NORMAL GAS LEVEL:



The screenshot displays the Wokwi web IDE interface. On the left, the code editor shows the following code:

```
1  */
2  #include <WiFi.h> //library for wifi
3  #include <PubSubClient.h> //library for MQTT
4  #include <LiquidCrystal.h>
5  #include <ESP32Servo.h>
6  #include <DHT.h> // Library for dht11
7  #define DHTPIN 15 // what pin we're connected to
8  #define DHTTYPE DHT22 // define type of sensor DHT 11
9  LiquidCrystal lcd(2,4,19,21,12,14);
10 int GreenLED= 18;
11 int RedLED = 5;
12 int BUZZER_PIN = 13;
13 const int servoPin = 22;
14 String data3;
15 int g;
16 Servo door;
17 int pos;
18 DHT dht (DHTPIN, DHTTYPE); // creating the instance by passing pin and type of
19 void callback(char* topic, byte* payload, unsigned int payloadLength)
20 //-----credentials of IBM Accounts-----
21 #define ORG "domlyv" //IBM ORGANITION ID
22 #define DEVICE_TYPE "abcd" //Device type mentioned in ibm watson IOT Platform
23 #define DEVICE_ID "12" //Device ID mentioned in ibm watson IOT Platform
24 #define TOKEN "12345678" //Token
25 float h, t;
26 // Customize the above values
```

On the right, the simulation area shows a green LCD displaying "GAS DETECTION NORMAL GAS LEVEL" and a servo motor. The console output shows the following:

```
Sending payload: {"temp":24.00,"Humid":40.00,"Gas":73}
Publish ok
Gas Level in Percentage :43
temp:24.00
Humid:40.00
Sending payload: {"temp":24.00,"Humid":40.00,"Gas":43}
Publish ok
```

# GAS DETECTION:

## HAZARDOUS LEVEL:

WOKWI

esp32-dht22.ino

```
5 */
6 #include <WiFi.h> //library for wifi
7 #include <PubSubClient.h> //library for MQTT
8 #include <LiquidCrystal.h>
9 #include <ESP32Servo.h>
10 #include <DHT.h> // Library for dht11
11 #define DHTPIN 15 // what pin we're connected to
12 #define DHTTYPE DHT22 // define type of sensor DHT 11
13 LiquidCrystal lcd(2,4,19,21,12,14);
14 int GreenLED = 18;
15 int RedLED = 5;
16 int BUZZER_PIN = 13;
17 const int servoPin = 22;
18 String data3;
19 int g;
20 Servo door;
21 int pos;
22 DHT dht (DHTPIN, DHTTYPE); // creating the instance by passing pin and type of
23 void callback(char* topic, byte* payload, unsigned int payloadLength)
24 //-----credentials of IBM Accounts-----
25 #define ORG "domlyv" // IBM ORGANITION ID
26 #define DEVICE_TYPE "abcd" // Device type mentioned in ibm watson IOT Platform
27 #define DEVICE_ID "12" // Device ID mentioned in ibm watson IOT Platform
28 #define TOKEN "12345678" // Token
29 float h, t;
30 // Customize the above values
```

Simulation

01:12:502 99%

GAS DETECTION  
HAZARDOUS LEVEL!

Sending payload: {"temp":24.00,"Humid":40.00,"Gas":46}  
Publish ok  
Gas Level in Percentage :96  
temp:24.00  
Humid:40.00  
Sending payload: {"temp":24.00,"Humid":40.00,"Gas":96}  
Publish ok

Firefox Installer (1).exe

WOKWI

esp32-dht22.ino

```
5 */
6 #include <WiFi.h> //library for wifi
7 #include <PubSubClient.h> //library for MQTT
8 #include <LiquidCrystal.h>
9 #include <ESP32Servo.h>
10 #include <DHT.h> // Library for dht11
11 #define DHTPIN 15 // what pin we're connected to
12 #define DHTTYPE DHT22 // define type of sensor DHT 11
13 LiquidCrystal lcd(2,4,19,21,12,14);
14 int GreenLED = 18;
15 int RedLED = 5;
16 int BUZZER_PIN = 13;
17 const int servoPin = 22;
18 String data3;
19 int g;
20 Servo door;
21 int pos;
22 DHT dht (DHTPIN, DHTTYPE); // creating the instance by passing pin and type of
23 void callback(char* topic, byte* payload, unsigned int payloadLength)
24 //-----credentials of IBM Accounts-----
25 #define ORG "domlyv" // IBM ORGANITION ID
26 #define DEVICE_TYPE "abcd" // Device type mentioned in ibm watson IOT Platform
27 #define DEVICE_ID "12" // Device ID mentioned in ibm watson IOT Platform
28 #define TOKEN "12345678" // Token
29 float h, t;
30 // Customize the above values
```

Simulation

03:12:910 97%

GAS DETECTION  
HAZARDOUS LEVEL!

Sending payload: {"temp":24.00,"Humid":40.00,"Gas":83}  
Publish ok  
Gas Level in Percentage :76  
temp:24.00  
Humid:40.00  
Reconnecting client to domlyv.messaging.internetofthings.ibmcloud.com

image (18).png image (17).png image (16).png WOK (1).doc WOK.doc