

Developing Wokwi Simulation

Date	09 November 2022
Team Id	PNT2022TMID35923
Title	Hazardous Area Monitoring for Industrial Plant using IoT

Wokwi code:

```
#include <WiFi.h>
```

```
#include <PubSubClient.h>
```

```
#include "DHT.h"
```

```
#define DHTPIN 15
```

```
#define DHTTYPE DHT22
```

```
#define LED 2
```

```
DHT dht (DHTPIN, DHTTYPE);
```

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

```
#define ORG "h4s7t8"
```

```
#define DEVICE_TYPE "IOT"
```

```
#define DEVICE_ID "12345"
```

```
#define TOKEN "12345678"
```

```
String data3;
```

```
float h, t;
```

```
char server[] = ORG ".messaging.internetofthings.ibmcloud.com";

char publishTopic[] = "iot-2/evt/Data/fmt/json";

char subscribetopic[] = "iot-2/cmd/command/fmt/String";

char authMethod[] = "use-token-auth";

char token[] = TOKEN;

char clientId[] = "d:" ORG ":" DEVICE_TYPE ":" DEVICE_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()// configureing the ESP32
```

```
{

  Serial.begin(115200);

  dht.begin();

  pinMode(LED,OUTPUT);

  delay(10);

  Serial.println();

  wificonnect();

  mqttconnect();

}
```

```
void loop()// Recursive Function
```

```
{
```

```
h = dht.readHumidity();

t = dht.readTemperature();

Serial.print("temp:");

Serial.println(t);

Serial.print("Humid:");

Serial.println(h);
```

```
PublishData(t, h);

delay(1000);

if (!client.loop()) {

    mqttconnect();

}

}
```

/.....retrieving to Cloud...../

```
void PublishData(float temp, float humid) {

    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 += "}";
```

```
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=="lighton")

{

Serial.println(data3);

digitalWrite(LED,HIGH);

}

else

{

Serial.println(data3);

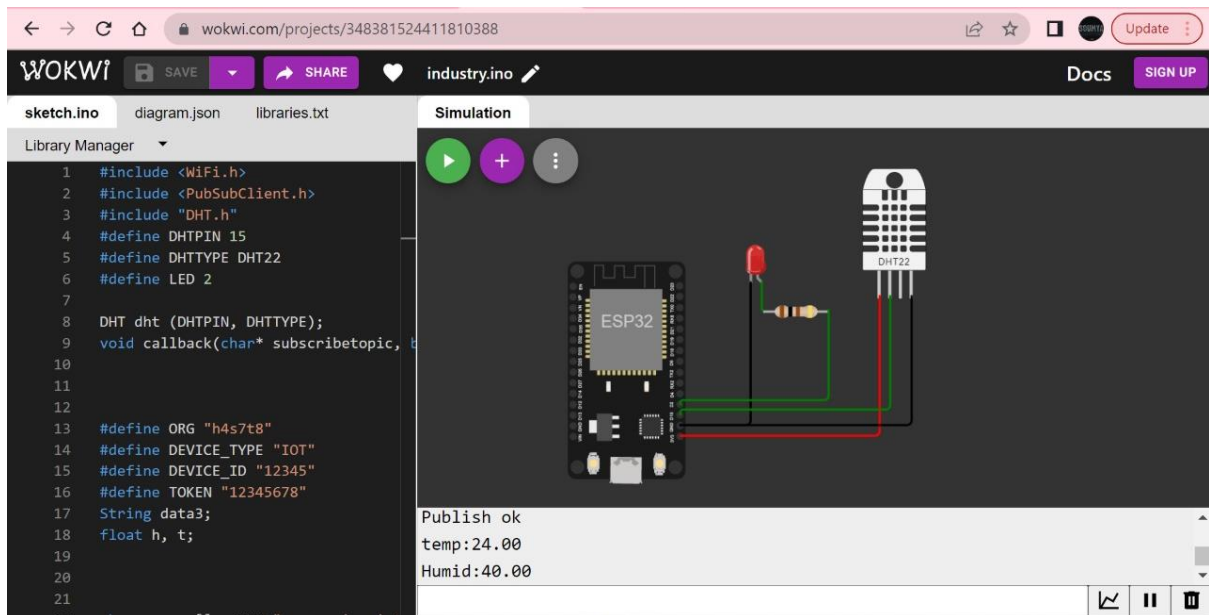
digitalWrite(LED,LOW);

}

data3="";

}
```

Wokwi Outputs:

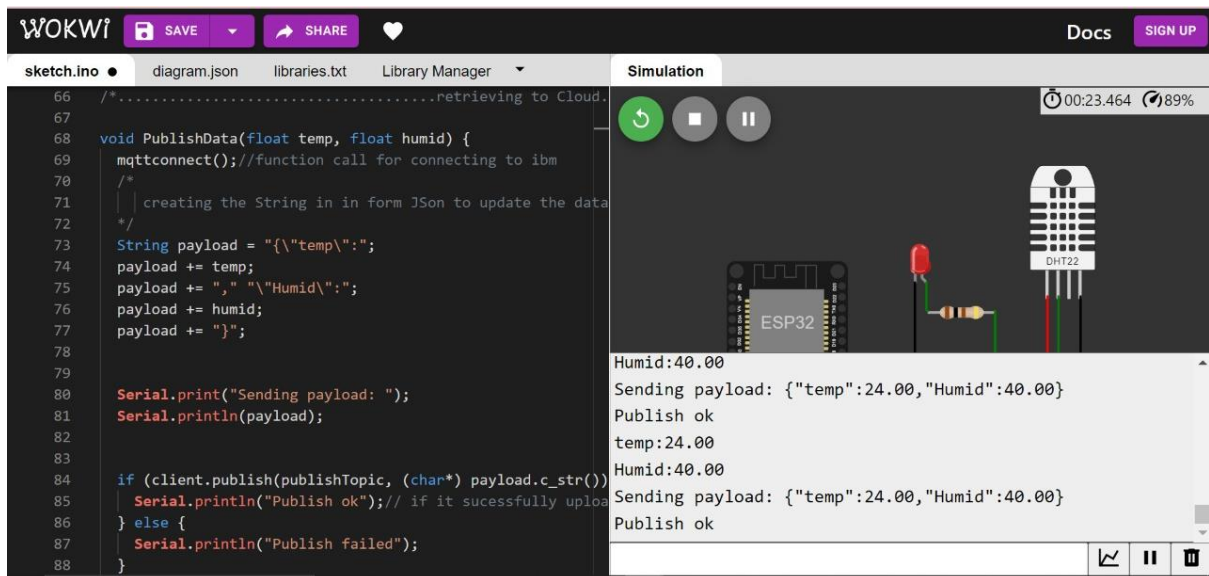


The screenshot shows the Wokwi web interface. The left pane displays a sketch with the following code:

```
1 #include <Wifi.h>
2 #include <PubSubClient.h>
3 #include "DHT.h"
4 #define DHTPIN 15
5 #define DHTTYPE DHT22
6 #define LED 2
7
8 DHT dht (DHTPIN, DHTTYPE);
9 void callback(char* subscribetopic, t
10
11
12
13 #define ORG "h4s7t8"
14 #define DEVICE_TYPE "IOT"
15 #define DEVICE_ID "12345"
16 #define TOKEN "12345678"
17 String data3;
18 float h, t;
19
20
21
```

The right pane shows a simulation of an ESP32 microcontroller connected to a DHT22 temperature and humidity sensor. The output window displays:

```
Publish ok
temp:24.00
Humid:40.00
```



The screenshot shows the Wokwi web interface with a timer and battery level indicator in the top right corner of the simulation pane. The sketch code is as follows:

```
66 /*.....retrieving to Cloud.
67
68 void PublishData(float temp, float humid) {
69   mqttconnect();//function call for connecting to ibm
70   /*
71   | creating the String in in form JSon to update the data
72   */
73   String payload = "{\"temp\":";
74   payload += temp;
75   payload += "," " \"Humid\":";
76   payload += humid;
77   payload += "}";
78
79
80   Serial.print("Sending payload: ");
81   Serial.println(payload);
82
83
84   if (client.publish(publishTopic, (char*) payload.c_str()))
85     Serial.println("Publish ok");// if it sucessfully uploa
86   } else {
87     Serial.println("Publish failed");
88   }

```

The simulation pane shows the same ESP32 and DHT22 setup. The output window displays the following sequence of messages:

```
Humid:40.00
Sending payload: {"temp":24.00,"Humid":40.00}
Publish ok
temp:24.00
Humid:40.00
Sending payload: {"temp":24.00,"Humid":40.00}
Publish ok
```