

# Personal Assistance for Seniors Who Are Self-Reliant

## Assignment – IV

**Name:** REVATHI

**RegisterNumber:** 921019104042

**E-Mail Id :** srirevathi670@gmail.com

**Question :** Write code and connections in wokwi for the ultrasonic sensor. Whenever the distance is less than 100 cms send an "alert" to the IBM cloud and display in the device recent events

## Code:

```
#include <WiFi.h> //library for wifi
#include <PubSubClient.h> //library for MQTT
#include "DHT.h" // Library for dht11
#define DHTPIN 15 // what pin we're connected to
#define DHTTYPE DHT22 // define type of sensor DHT 11
#define LED 2

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

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

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

#define ORG "q3s9e0" //IBM ORGANIZATION ID
#define DEVICE_TYPE "device" //Device type mentioned in ibm watson IOT Platform
#define DEVICE_ID "deviceId" //Device ID mentioned in ibm watson IOT Platform
#define TOKEN "LWXj2CFxf@paR)qJUm" //Token
String data3;
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 subscribtopic[] = "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()// 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("Alert Distance :");
  Serial.println(t);

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

  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)
{
```

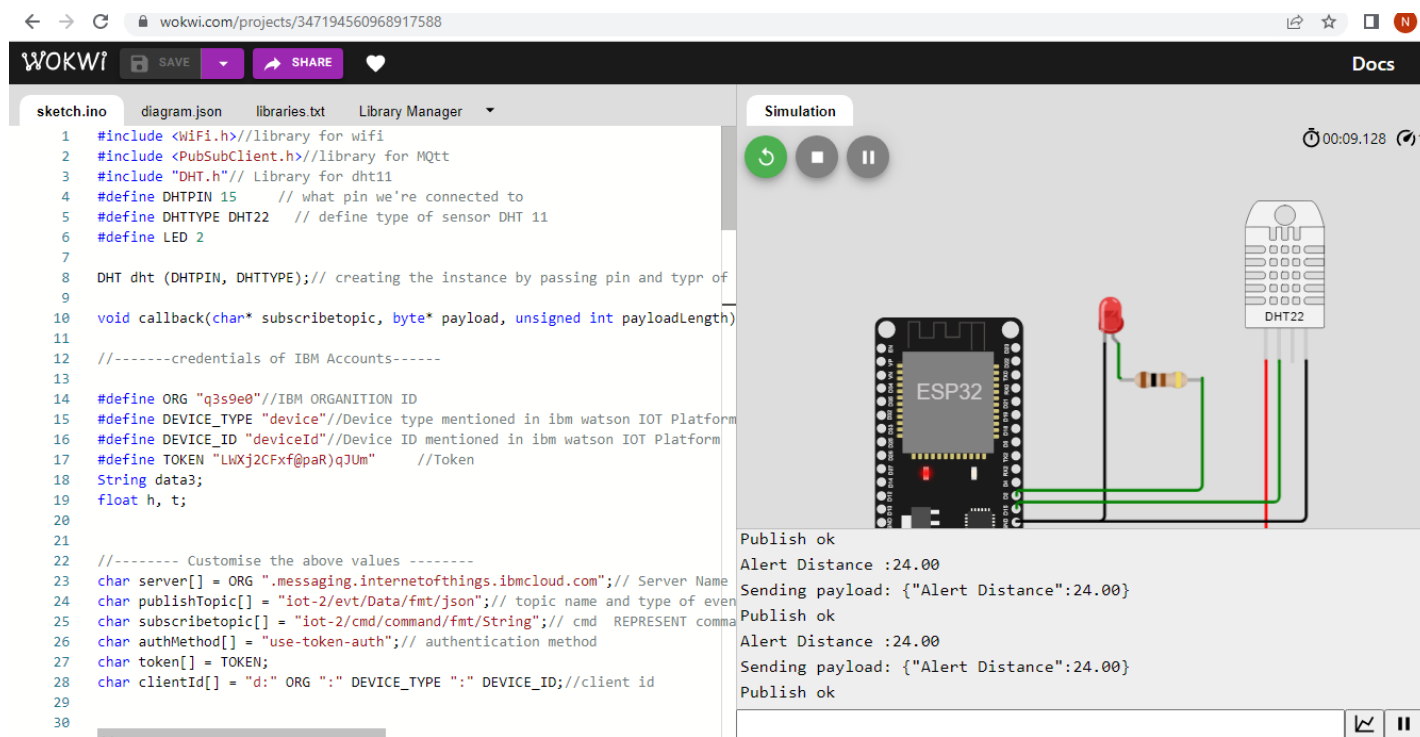
```
    Serial.print("callback invoked for topic: ");
```

```

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="";
}

```

## Simulation Output:



The screenshot displays the Wokwi IDE interface. On the left, the 'sketch.ino' file contains the following code:

```

1 #include <WiFi.h> //library for wifi
2 #include <PubSubClient.h> //library for MQTT
3 #include "DHT.h" // Library for dht11
4 #define DHTPIN 15 // what pin we're connected to
5 #define DHTTYPE DHT22 // define type of sensor DHT 11
6 #define LED 2
7
8 DHT dht (DHTPIN, DHTTYPE); // creating the instance by passing pin and type of
9
10 void callback(char* subscribetopic, byte* payload, unsigned int payloadLength)
11
12 //-----credentials of IBM Accounts-----
13
14 #define ORG "q3s9e0" //IBM ORGANIZATION ID
15 #define DEVICE_TYPE "device" //Device type mentioned in ibm watson IOT Platform
16 #define DEVICE_ID "deviceId" //Device ID mentioned in ibm watson IOT Platform
17 #define TOKEN "LwXj2CFxf@paR)qJUu" //Token
18 String data3;
19 float h, t;
20
21
22 //----- Customise the above values -----
23 char server[] = ORG ".messaging.internetofthings.ibmcloud.com"; // Server Name
24 char publishTopic[] = "iot-2/evt/Data/fmt/json"; // topic name and type of event
25 char subscribetopic[] = "iot-2/cmd/command/fmt/String"; // cmd REPRESENT comma
26 char authMethod[] = "use-token-auth"; // authentication method
27 char token[] = TOKEN;
28 char clientId[] = "d:" ORG ":" DEVICE_TYPE ":" DEVICE_ID; //client id
29
30

```

On the right, the 'Simulation' window shows a visual representation of the ESP32 microcontroller connected to a DHT22 temperature and humidity sensor and a red LED. The output log on the right side of the simulation window shows the following sequence of events:

```

Publish ok
Alert Distance :24.00
Sending payload: {"Alert Distance":24.00}
Publish ok
Alert Distance :24.00
Sending payload: {"Alert Distance":24.00}
Publish ok

```

## Cloud Output:

