

Personal Assistance for Seniors Who Are Self-Reliant

Assignment – IV

Name: Parkavi V

RegisterNumber: 921019104035

E-Mail Id : parkaviparkavi996@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 "p88gsn" //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 "oTPYNg?nxxj0&eBelI" //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)
{
```

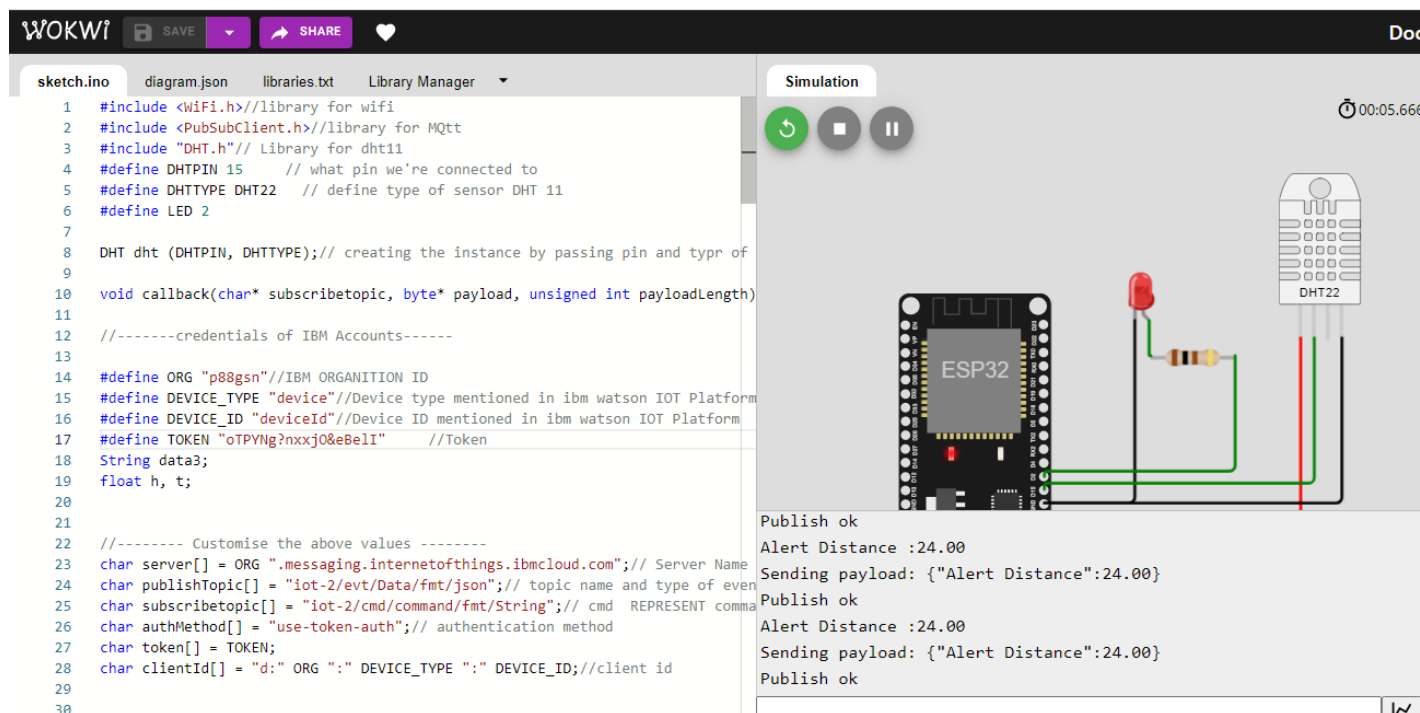
```
    Serial.print("callback invoked for topic: ");
```

```

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 shows the WOKWI simulation environment. On the left, the sketch code is displayed, which includes libraries for WiFi, PubSubClient, and DHT. The code defines pins for the DHT22 sensor (DHTPIN 15, DHTTYPE DHT22) and an LED (LED 2). It creates a DHT instance and defines a callback function. The code also includes IBM Watson IoT Platform credentials and a client ID. The simulation output on the right 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:

← → ↺

p88gsn.internetofthings.ibmcloud.com/dashboard/devices/browse

IBM Watson IoT Platform

?

par ID:

⋮

⚙

👤

📡

📊

📈

🔒

⚙

Browse

Action

Device Types

Interfaces

🔍 Search by Device ID

Device Simulator

| <input type="checkbox"/> | Device ID | Status | Device Type | Class ID | Date Added |
|----------------------------|-----------|-----------|-------------|----------|----------------------|
| ▼ <input type="checkbox"/> | deviceId | Connected | device | Device | Nov 2, 2022 12:25 PM |

Identity

Device Information

Recent Events

State

Logs

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

| Event | Value | Format | Last Received |
|-------|-----------------------|--------|-------------------|
| Data | {"Alert Distance":24} | json | a few seconds ago |
| Data | {"Alert Distance":24} | json | a few seconds ago |
| Data | {"Alert Distance":24} | json | a few seconds ago |

Link:

<https://wokwi.com/projects/347197534358209106>