

Name: HARISKUMAR S **RegNo:** GCTC1918111

Degree & Branch: B.Tech - Final Year - Information Technology

College: Government College of Technology, Coimbatore – 641 013

Subject: Professional Readiness for Innovation, Employability & Entrepreneurship (Nalaiya Thiran)

Assignment – 4 ESP32 with Ultrasonic Sensor and IoT Watson

Task – 1:

Write code and connections in wokwi for ultrasonic sensor. Whenever distance is less than 100 cms send "alert" to ibm cloud and display in device recent events. Upload document with wokwi share link and images of ibm cloud

Solution:

Program:

```
#include <WiFi.h> //library for wifi
#include <PubSubClient.h> //library for MQTT
#define TRIG_PIN 4 // ESP32 pin GPIO23 connected to Ultrasonic Sensor's TRIG pin
#define ECHO_PIN 2 // ESP32 pin GPIO22 connected to Ultrasonic Sensor's ECHO pin
#define DHTTYPE DHT11 // define type of sensor DHT 11

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

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

#define ORG "19gaqm" //IBM ORGANITION ID
#define DEVICE_TYPE "HK07" //Device type mentioned in ibm watson IOT Platform
#define DEVICE_ID "1234" //Device ID mentioned in ibm watson IOT Platform
#define TOKEN "123456789" //Token
String data3;
float duration_us, distance_cm;

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/test/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, port and wificredential
```

```

void setup()// configuring the ESP32
{
    Serial.begin(115200);
    pinMode(TRIG_PIN, OUTPUT);
    pinMode(ECHO_PIN, INPUT);
    delay(10);
    Serial.println();
    wificonnect();
    mqttconnect();
}

void loop()// Recursive Function
{
    digitalWrite(TRIG_PIN, HIGH);
    delayMicroseconds(10);
    digitalWrite(TRIG_PIN, LOW);
    // measure duration of pulse from ECHO pin
    duration_us = pulseIn(ECHO_PIN, HIGH);
    // calculate the distance
    distance_cm = 0.017 * duration_us;
    // print the value to Serial Monitor
    Serial.print("distance: ");
    Serial.print(distance_cm);
    Serial.println(" cm");
    delay(500);
    if(distance_cm<100){
        PublishData(distance_cm);
    }
    delay(1000);
    if (!client.loop()) {
        mqttconnect();
    }
}

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

void PublishData(float distance_cm)
{
    mqttconnect();//function call for connecting to ibm
    /*
        creating the String in in form JSon to update the data to ibm cloud
    */
    String payload = "{\"Alert\":\"ON\"";
    payload += ", \"Distance_cm\"";
    payload += distance_cm;
    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(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());
}

```

Output of Program:

Wokwi Project Link:

<https://wokwi.com/projects/346735894223389268>

Wokwi Website Screenshot:

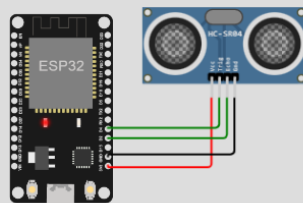
WOKWI

esp32-dht22.ino diagram.json libraries.txt Library Manager

```
1 #include <WiFi.h> //library for wifi
2 #include <PubSubClient.h> //library for MQTT
3 #define TRIG_PIN 4 // ESP32 pin GPIO23 connected to Ultrasonic Sensor's TRIG pin
4 #define ECHO_PIN 2 // ESP32 pin GPIO22 connected to Ultrasonic Sensor's ECHO pin
5
6
7 //-----credentials of IBM Accounts-----
8
9 #define ORG "19gaqm" //IBM ORGANITION ID
10 #define DEVICE_TYPE "HK07" //Device type mentioned in ibm watson IOT Platform
11 #define DEVICE_ID "1234" //Device ID mentioned in ibm watson IOT Platform
12 #define TOKEN "123456789" //Token
13 String data3;
14 float duration_us, distance_cm;
15
16
17 //----- Customise the above values -----
18 char server[] = ORG ".messaging.internetofthings.ibmcloud.com"; // Server Name
19 char publishTopic[] = "iot-2/evt/Data/fmt/json"; // topic name and type of event
20 char subscribetopic[] = "iot-2/cmd/test/fmt/String"; // cmd REPRESENT command
21 char authMethod[] = "use-token-auth"; // authentication method
22 char token[] = TOKEN;
23 char clientId[] = "d:" ORG ":" DEVICE_TYPE ":" DEVICE_ID; //client id
24
25
26 //-----
27 WiFiClient wificlient; // creating the instance for wificlient
28 PubSubClient client(server, 1883, wificlient); //calling the predefined client
29 void setup() // configuring the ESP32
30 {
```

Simulation

02:40.523 101%



Connecting to
WiFi connected
IP address:
10.10.0.2
Reconnecting client to 19gaqm.messaging.internetofthings.ibmcloud.com

IBM Cloud Screenshot:

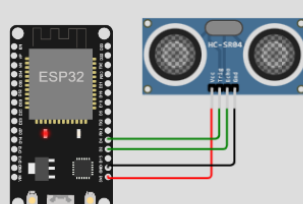
WOKWI

esp32-dht22.ino diagram.json libraries.txt Library Manager

```
1 #include <WiFi.h> //library for wifi
2 #include <PubSubClient.h> //library for MQTT
3 #define TRIG_PIN 4 // ESP32 pin GPIO23 connected to Ultrasonic Sensor's TRIG pin
4 #define ECHO_PIN 2 // ESP32 pin GPIO22 connected to Ultrasonic Sensor's ECHO pin
5
6
7 //-----credentials of IBM Accounts-----
8
9 #define ORG "19gaqm" //IBM ORGANITION ID
10 #define DEVICE_TYPE "HK07" //Device type mentioned in ibm watson IOT Platform
11 #define DEVICE_ID "1234" //Device ID mentioned in ibm watson IOT Platform
12 #define TOKEN "123456789" //Token
13 String data3;
14 float duration_us, distance_cm;
15
16
17 //----- Customise the above values -----
18 char server[] = ORG ".messaging.internetofthings.ibmcloud.com"; // Server Name
19 char publishTopic[] = "iot-2/evt/Data/fmt/json"; // topic name and type of event
20 char subscribetopic[] = "iot-2/cmd/test/fmt/String"; // cmd REPRESENT command
21 char authMethod[] = "use-token-auth"; // authentication method
22 char token[] = TOKEN;
23 char clientId[] = "d:" ORG ":" DEVICE_TYPE ":" DEVICE_ID; //client id
24
25
26 //-----
27 WiFiClient wificlient; // creating the instance for wificlient
28 PubSubClient client(server, 1883, wificlient); //calling the predefined client
29 void setup() // configuring the ESP32
30 {
```

Simulation

02:50.754 100%



Publish ok
distance: 47.96 cm
Sending payload: {"Alert":"ON","Distance_cm":47.96}
Publish ok
distance: 47.96 cm
Sending payload: {"Alert":"ON","Distance_cm":47.96}
Publish ok

WOKWI

SAVE

SHARE

esp32-dht22.ino

diagram.json

libraries.txt

Simulation

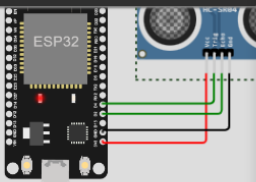
02:12.497 95%

Library Manager

```
1 #include <WiFi.h> //library for WiFi
2 #include <PubSubClient.h> //library
3 #define TRIG_PIN 4 // ESP32 pin G
4 #define ECHO_PIN 2 // ESP32 pin G
5
6
7 //-----credentials of IBM Account
8
9 #define ORG "19gaqm" //IBM ORGANIZATION
10 #define DEVICE_TYPE "HK07" //Device
11 #define DEVICE_ID "1234" //Device ID
12 #define TOKEN "123456789" //Token
13 String data3;
14 float duration_us, distance_cm;
15
16
17 //----- Customise the above values
18 char server[] = ORG ".messaging.internetofthings.ibmcloud.com";
19 char publishTopic[] = "iot-2/evt/";
20 char subscribetopic[] = "iot-2/cmd/";
21 char authMethod[] = "use-token-auth";
22 char token[] = TOKEN;
23 char clientId[] = "d:" ORG ":" DEVICE_ID;
24
25 //-----
26 WiFiClient wificlient; // creating
27 PubSubClient client(server, 1883, wificlient);
28 void setup() // configuring the ESP32
```

Editing Ultrasonic Distance Sensor

Distance: 46cm



Publish ok

distance: 45.95 cm

Sending payload:

{"Alert": "ON", "Distance_cm": 45.95}

Publish ok

distance: 46.00 cm

IBM Watson IoT Platform

hari.1918111@gct.ac.in

ID: 19gaqm

Browse

Action

Device Types

Interfaces

Add Device

Event	Value
Data	{"Alert": "ON", "Distance_cm": 45.95}
Data	{"Alert": "ON", "Distance_cm": 45.95}
Data	{"Alert": "ON", "Distance_cm": 45.95}
Data	{"Alert": "ON", "Distance_cm": 45.95}
Data	{"Alert": "ON", "Distance_cm": 45.95}

1

Explanation of Program:

Initially, we have imported the <Wifi.h> and <PubSubClient.h> header files as they are needed to connect wifi and MQTT Protocol. Then, Define Trigger pin and Echo pin values where the ultrasonic sensor is connected with ESP32 module. Then, Define the IBM Account Credentials such as ORG, Device_Type, Device_ID and Token. Also define the server, publishTopic, SubscribeTopic, authMethod, Token and ClientID. Create Object for WifiClient and PubSubClient.

Then, Start the void Setup() Function, Begin the Serial Monitor and set PinMode of Trigger Pin as Output and Echo Pin as Input and call wificonnect() and mqttconnect() to initialize wifi and mqtt Connection and Define their methods to make the Connection.

Then, Begin the void loop() function, digitalWrite HIGH to Trigger Pin and create a delay of 10 microseconds and write back LOW. Then, use pulseIn() function with Echo Pin to calculate the Duration and calculate the distance. If the Distance is less than 100cm call PublishData() Function to publish the Data to IoT Watson Device.

Finally, Define the PublishData() function with message as parameter. Then Define string that contains the payload with the message to be sent in the Json Format. Call Client.publish() function with publishTopic and payload as parameter. Also define the wificonnect() and mqttconnect() to make intial connection with wifi and mqtt connection.