

ASSIGNMENT-4

DISTANCEDETECTIONUSINGULTRASONICSENS OR

| | |
|--------|------------------|
| Date | 20 October2022 |
| TeamID | PNT2022TMID29941 |
| Name | S Divya(TL) |

Question:

Writecodeandconnections inwokwifor ultrasonicsensor. Whenever distance is less than 100cm
ssend" alert"toibmcloudanddisplay in 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 typr of dht
connected

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

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

#define ORG "ketslb" //IBM ORGANITION ID
#define DEVICE_TYPE "testbatchass1" //Device type mentioned in ibm watson IOT
Platform
```

```

#define DEVICE_ID "testass4">//Device ID mentioned in ibm watson IOT Platform
#define TOKEN "t4444cvn97m78mx4r3467rg0cq3" //Token

String data3;

float 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 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()// configureing the ESP32
{
    Serial.begin(115200);

    dht.begin();

    pinMode(LED, OUTPUT);

    delay(10);

    Serial.println();

    wificonnect();

    mqttconnect();

```

```
}
```

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

```
{
```

```
    t = dht.readTemperature();
```

```
    Serial.print("temperature:");
```

```
    Serial.println(t);
```

```
    PublishData(t);
```

```
    delay(1000);
```

```
    if (!client.loop()) {
```

```
        mqttconnect();
```

```
    }
```

```
}
```

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

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

```
    mqttconnect();//function call for connecting to ibm
```

```
    /*
```

```
        creating the String in in form JSON to update the data to ibm cloud
```

```
    */
```

```
    String payload = "{\"temperature\":";
```

```
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 = "";
```

}

OUTPUT:

The screenshot shows the WOKWI simulation interface. On the left, the code for the ESP32 is displayed in the 'divya.ino' file. The code includes libraries for WiFi, PubSubClient, and DHT. It defines the DHT pin as 15 and the DHT type as DHT22. A callback function is defined to handle incoming data. The code also sets up an MQTT client with credentials for an IBM Watson IoT Platform. The simulation output on the right shows the sensor reading a distance of 141.21 cm and sending a JSON payload to the cloud.

```
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 DHT dht (DHTPIN, DHTTYPE); // creating the instance by passing pin and type of
8
9 void callback(char* subscribetopic, byte* payload, unsigned int payloadLength)
10
11 //-----credentials of IBM Accounts-----
12
13 #define ORG "ketslb" // IBM ORGANIZATION ID
14 #define DEVICE_TYPE "testbatchass1" // Device type mentioned in IBM Watson IoT Platform
15 #define DEVICE_ID "testass4" // Device ID mentioned in IBM Watson IoT Platform
16 #define TOKEN "t4444cvm97m78mx4r3467rg0cq3" // Token
17 String data3;
18 float t;
19
20 //-----Customise the above values -----
21
22 char server[] = ORG ".messaging.internetofthings.ibmcloud.com"; // Server Name
23 char publishTopic[] = "iot-2/evt/Data/fmt/json"; // topic name and type of event
24 char subscribetopic[] = "iot-2/cmd/command/fmt/String"; // cmd REPRESENT command
25 char authMethod[] = "use-token-auth"; // authentication method
26 char token[] = TOKEN;
27 char clientId[] = "d:" ORG ":" DEVICE_TYPE ":" DEVICE_ID; // client id
28
```

Simulation output:

```
no object found
Sending payload: {"distance":141.21,"object":"No"}
Publish: ok
Distance in cm 141.21
no object found
Sending payload: {"distance":141.21,"object":"No"}
Publish ok
```

Data send to the IBM cloud device when the object is far:

The screenshot shows the IBM Watson IoT Platform dashboard. The 'Recent Events' tab is selected, displaying a table of data events. The table has columns for Event, Value, Format, and Last Received. The events show a distance of 141.21 cm and an object status of 'No'.

| Event | Value | Format | Last Received |
|-------|-----------------------------------|--------|-------------------|
| Data | ["distance":141.21,"object":"No"] | json | a few seconds ago |
| Data | ["distance":141.21,"object":"No"] | json | a few seconds ago |
| Data | ["distance":141.21,"object":"No"] | json | a few seconds ago |
| Data | ["distance":141.18,"object":"No"] | json | a few seconds ago |
| Data | ["distance":141.2,"object":"No"] | json | a few seconds ago |