

## Assignment - 4

### IoT Based Safety Gadget for Child Safety Monitoring and Notification

Student Name	SATHISHKUMAR .M
Student Roll Number	611719106022

#### 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. Upload document with wokwi share link and images of IBM cloud

#### CODE :

```
#include <WiFi.h> //library for wifi
#include <PubSubClient.h> //library for MQTT

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

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

#define ORG "jcs25v" //IBM ORGANITION ID
#define DEVICE_TYPE "123" //Device type mentioned in ibm watson IOTPlatform
#define DEVICE_ID "12345" //Device ID mentioned in ibm watson IOT
Platform#define TOKEN "123456789" //Token
String data3;
float dist;
//----- 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/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,portand wificredential

int LED = 4;
int trig = 5;
int echo = 18;
void setup()
{
  Serial.begin(115200);
  pinMode(trig,OUTPUT);
```

```

pinMode(echo,INPUT);
pinMode(LED, OUTPUT);
delay(10);
wificonnect();
mqttconnect();
}
void loop()// Recursive Function
{

digitalWrite(trig,LOW);
digitalWrite(trig,HIGH);
delayMicroseconds(10);
digitalWrite(trig,LOW);
float dur = pulseIn(echo,HIGH);
float dist = (dur * 0.0343)/2;
Serial.print ("Distancein cm");
Serial.println(dist);


PublishData(dist);
delay(1000);
if (!client.loop()) {
    mqttconnect();
}
}

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

void PublishData(float dist) {
    mqttconnect();//function call for connecting to ibm
    /*
        creating the String in in form JSON to update the data to ibm cloud
    */
    String object;
    if (dist <100)
    {
        digitalWrite(LED,HIGH);
        Serial.println("object is near");
        object = "Alert: Person Detected";
    }
    else
    {
        digitalWrite(LED,LOW);
        Serial.println("no object found");
        object = "No";
    }

    String payload = "{\"distance\":\"";
    payload += dist;
    payload += "," " \"object\":\"";
    payload += object;
    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];
    }

    data3="";
}

```

## Wokwi Link:

<https://wokwi.com/projects/347204840687927891>

## Output and Simulation:

The screenshot displays the Wokwi web-based IDE. On the left, the 'sketch.ino' file is open, showing the following code:

```

1 #include <WiFi.h>
2 #include <PubSubClient.h>
3 void callback(char* subscribetopic, byte* payload, unsigned int payloadLength)
4 #define ORG "ocknc6"
5 #define DEVICE_TYPE "esp32"
6 #define DEVICE_ID "12345"
7 #define TOKEN "12345678"
8 String data3;
9 char server[] = ORG ".messaging.internetofthings.ibmcloud.com";
10 char publishTopic[] = "iot-2/evt/Data/fmt/json";
11 char subscribetopic[] = "iot-2/cmd/test/fmt/String";
12 char authMethod[] = "use-token-auth";
13 char token[] = TOKEN;
14 char clientId[] = "d:" ORG ":" DEVICE_TYPE ":" DEVICE_ID;
15 WiFiClient wificlient;
16 PubSubClient client(server, 1883, callback, wificlient);
17 const int trigPin = 5;
18 const int echoPin = 18;
19 #define SOUND_SPEED 0.034
20 long duration;
21 float distance;
22 void setup() {
23   Serial.begin(115200);
24   pinMode(trigPin, OUTPUT);
25   pinMode(echoPin, INPUT);
26   wificlient.connect();
27   mqttconnect();
28 }
29 void loop()

```

On the right, the 'Simulation' window is active, showing a visual representation of the ESP32 microcontroller and the HC-SR04 ultrasonic sensor. A slider for the sensor's distance is set to 73cm. Below the simulation, the output log shows the following sequence of events:

```

Distance (cm): 135.97
Distance (cm): 89.98
ALERT!!
Sending payload: {"Distance":89.98,"ALERT!!":"Distance less than 100cms"}
Publish ok
Distance (cm): 72.96
ALERT!!

```

The bottom of the image shows the Windows taskbar with the system clock indicating 5:39 PM on 11/12/2022.

## IBM cloud:

### Device Information:

The screenshot shows the IBM Watson IoT Platform dashboard. The top navigation bar includes 'Browse', 'Action', 'Device Types', and 'Interfaces'. A search bar is present with the text 'Search by Device ID'. The main content area displays a table with columns: Device ID, Status, Device Type, Class ID, and Date Added. The device 123\_1 is listed with a status of 'Connected'. Below the table, a detailed view of the device information is shown, including fields like Device ID, Device Type, Date Added, Added By, and Connection Status. A notification at the bottom right indicates '2 Simulations running'.

Device ID	Status	Device Type	Class ID	Date Added
123_1	Connected	123	Device	Nov 12, 2022 5:21 PM

  

Device ID	Device Information
123_1	<p>Device ID: 123_1</p> <p>Device Type: 123</p> <p>Date Added: Nov 12, 2022 5:21 PM</p> <p>Added By: 611719106022@smartinternz.com</p> <p>Connection Status: Connected</p> <p>Connection Time: Nov 12, 2022 5:45 PM</p> <p>Client Address: 157.49.154.49 SecureToken</p>

### Device Recent Events:

Whenever the distance is less than 100 cms send an "alert" to the IBM cloud and display in the device recent events.

The screenshot shows the IBM Watson IoT Platform dashboard with the 'Recent Events' tab selected. A message states: 'The recent events listed show the live stream of data that is coming and going from this device.' Below this, a table displays the recent events. The table has columns: Event, Value, Format, and Last Received. The events are listed as 'event\_1' with values like '{"distance":102}', '{"distance":221}', '{"distance":133}', '{"distance":257}', and '{"distance":185}', all in 'json' format, received 'a few seconds ago'. A notification at the bottom right indicates '2 Simulations running'.

Event	Value	Format	Last Received
event_1	{"distance":102}	json	a few seconds ago
event_1	{"distance":221}	json	a few seconds ago
event_1	{"distance":133}	json	a few seconds ago
event_1	{"distance":257}	json	a few seconds ago
event_1	{"distance":185}	json	a few seconds ago