

IBM ASSIGNMENT – 4

Student Name	Salman zahir R
Student Roll Number	710019106037
Team ID	PNT2022TMID42240
Assignment Date	06 NOVEMBER 2022

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.

CODE:

```
#include <WiFi.h>
#include <PubSubClient.h>
void callback(char* subscribetopic, byte* payload, unsigned int
payloadLength);
//-----credentials of IBM Accounts-----
#define ORG "jwl2wf"//IBM ORGANITION ID
#define DEVICE_TYPE "salman_device"//Device type mentioned in ibm watson IOT
Platform
#define DEVICE_ID "2001"//Device ID mentioned in ibm watson IOT Platform
#define TOKEN "123456789" //Token
String data3;
char server[] = ORG ".messaging.internetofthings.ibmcloud.com";
char publishTopic[] = "iot-2/evt/Data/fmt/json";
char subscribetopic[] = "iot-2/cmd/test/fmt/String";
char authMethod[] = "use-token-auth";
char token[] = TOKEN;
char clientId[] = "d:" ORG ":" DEVICE_TYPE ":" DEVICE_ID;
WiFiClient wifiClient;
PubSubClient client(server, 1883, callback ,wifiClient);
const int trigPin = 5;
const int echoPin = 18;
#define SOUND_SPEED 0.034
long duration;
float distance;
void setup() {
Serial.begin(115200);
pinMode(trigPin, OUTPUT);
pinMode(echoPin, INPUT);
wificonnect();
mqttconnect();
}
void loop()
```

```

{
digitalWrite(trigPin, LOW);
delayMicroseconds(2);
digitalWrite(trigPin, HIGH);
delayMicroseconds(10);
digitalWrite(trigPin, LOW);
duration = pulseIn(echoPin, HIGH);
distance = duration * SOUND_SPEED/2;
Serial.print("Distance (cm): ");
Serial.println(distance);
if(distance<100)
{
Serial.println("ALERT!!");
delay(1000);
PublishData(distance);
delay(1000);
if (!client.loop()) {
mqttconnect();
}
}
delay(1000);
}
void PublishData(float dist) {
mqttconnect();
String payload = "{\"Distance\":";
payload += dist;
payload += ", \"ALERT!!\":";
payload += "\"Distance less than 100cms\"";
payload += "}";
Serial.print("Sending payload: ");
Serial.println(payload);

if (client.publish(publishTopic, (char*) payload.c_str())) {
Serial.println("Publish ok");
} 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();
}

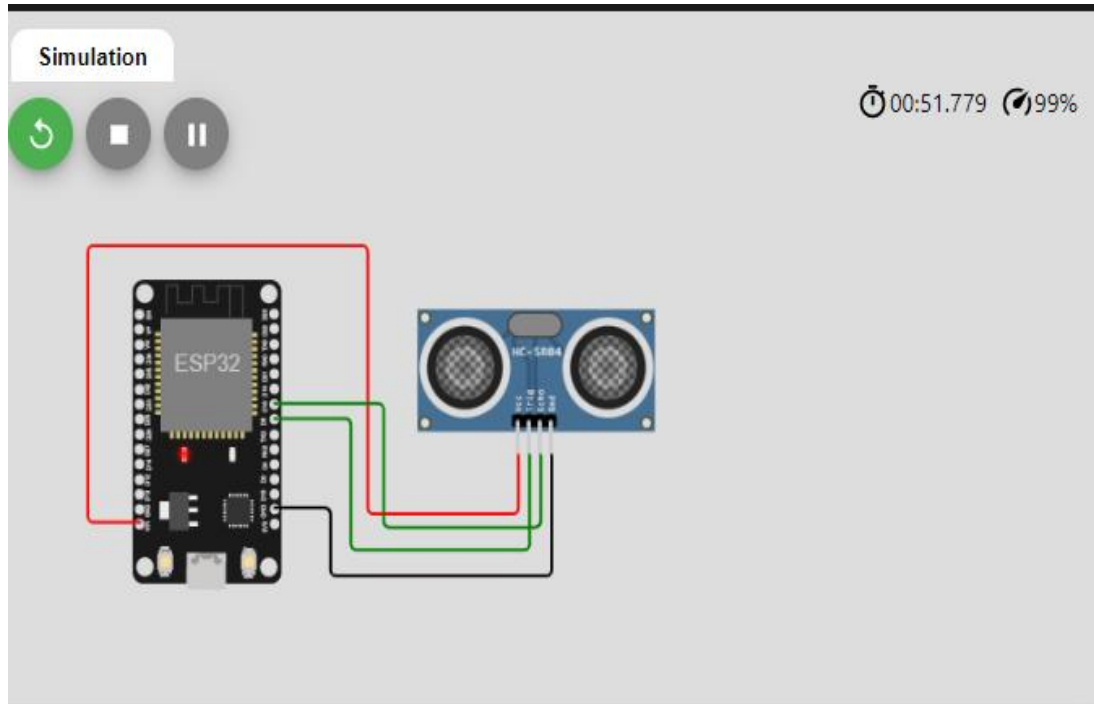
```

```

}
}
void wificonnect()
{
  Serial.println();
  Serial.print("Connecting to ");
  WiFi.begin("Wokwi-GUEST", "", 6);
  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);
  data3="";
}

```

SCHEMATIC/CIRCUIT DIAGRAM:



WOKWI LINK:

<https://wokwi.com/projects/347659562167304786>

WOKWI OUTPUT:

The screenshot shows the Wokwi IDE interface. On the left, the code for `esp32-dht22.ino` is displayed. The code includes headers for `WiFi` and `PubSubClient`, and defines a callback function. It sets up an ESP32 board with an HC-SR04 sensor. The code includes comments for IBM Watson IoT Platform credentials and device information. The `setup` function initializes the serial port and pins. The `loop` function publishes data to the IoT platform and triggers the sensor.

```
1 #include <WiFi.h>
2 #include <PubSubClient.h>
3 void callback(char* topic, byte* payload, unsigned int
4   payloadlength);
5 //-----credentials of IBM Accounts-----
6 #define ORG "jw12w4f"//IBM ORGANIZATION ID
7 #define DEVICE_TYPE "salman_device"//Device type mentioned in ibm watson IOT Platform
8 #define DEVICE_ID "2001"//Device ID mentioned in ibm watson IOT Platform
9 #define TOKEN "123456789" //Token
10 String data3;
11 char server[] = ORG ".messaging.internetofthings.ibmcloud.com";
12 char publishTopic[] = "iot-2/evt/data/fmt/json";
13 char subscribTopic[] = "iot-2/cmd/test/fmt/String";
14 char authMethod[] = "use-token-auth";
15 char token[] = TOKEN;
16 char clientId[] = "d:" ORG ":" DEVICE_TYPE ":" DEVICE_ID;
17 WiFiClient wifiClient;
18 PubSubClient client(server, 1883, callback, wifiClient);
19 const int trigPin = 5;
20 const int echoPin = 18;
21 #define SOUND_SPEED 0.034
22 long duration;
23 float distance;
24 void setup() {
25   Serial.begin(115200);
26   pinMode(trigPin, OUTPUT);
27   pinMode(echoPin, INPUT);
28   wifiConnect();
29   mqttConnect();
30 }
```

On the right, the simulation window shows the same circuit diagram. Below the diagram, the output log displays the following messages:

```
than 100cms"}
Publish ok
Distance (cm): 57.95
ALERT!!
Sending payload: {"Distance":57.95,"ALERT!!":"Distance less
than 100cms"}
Publish ok
```

IBM CLOUD OUTPUT :

The screenshot shows the IBM Watson IoT Platform interface. The top navigation bar includes 'Browse', 'Action', 'Device Types', and 'Interfaces'. The main content area displays a list of devices, with the first device '2001' selected. The 'Recent Events' tab is active, showing a table of data events. The table has columns for 'Event', 'Value', 'Format', and 'Last Received'. The events are JSON objects containing distance and alert information. A notification at the bottom right states '1 Simulation running'.

Event	Value	Format	Last Received
Data	{"Distance":98.94,"ALERT!!":"Distance less than 1..."}	json	a few seconds ago
Data	{"Distance":98.94,"ALERT!!":"Distance less than 1..."}	json	a few seconds ago
Data	{"Distance":81.97,"ALERT!!":"Distance less than 1..."}	json	a minute ago

1 Simulation running