

Assignment -4

SmartFarmer - IoT Enabled Smart Farming Application

Assignment Date	17 October 2022
Student Name	Manivel pandian . R
Student Roll Number	910619106029
Maximum Marks	2 Marks

Question-1:

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 1 :

```
#include <WiFi.h>
#include <PubSubClient.h>
void callback(char* subscribetopic, byte* payload, unsigned int
payloadLength);
//-----credentials of IBM Accounts-----
#define ORG "9p3i98"//IBM ORGANITION ID
#define DEVICE_TYPE "ESP32"//Device type mentioned in ibm watson IOT Platform
#define DEVICE_ID "12345"//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!!\": \"\" \"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="";
}

```

Wokwi Link :

<https://wokwi.com/projects/347023409599742548>

Output and Simulation :

The screenshot displays the Wokwi IDE interface. On the left, the 'sketch.ino' file is open, showing a C++ code snippet for an ESP32 device connected to an IBM Watson IoT Platform. The code includes headers for WiFi and PubSubClient, defines device credentials, and sets up a loop that triggers an ultrasonic sensor and publishes distance data to the cloud. On the right, the 'Simulation' window shows a visual representation of the ESP32 and an HC-SR04 ultrasonic sensor. Below the simulation, the console output shows the device publishing distance data (22.95 cm) and sending an alert message when the distance is less than 100 cm.

```
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 "9p3l98"//IBM ORGANIZATION ID
7 #define DEVICE_TYPE "ESP32"//Device type mentioned in ibm watson IoT Platform
8 #define DEVICE_ID "12345"//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 subscribeTopic[] = "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 }
31 void loop()
32 {
33   digitalWrite(trigPin, LOW);
34   delayMicroseconds(2);
35   digitalWrite(trigPin, HIGH);
36   delayMicroseconds(10);
37   digitalWrite(trigPin, LOW);
```

Simulation Output:

```
Publish ok
Distance (cm): 22.95
ALERT!!
Sending payload: {"Distance":22.95,"ALERT!!":"Distance less than 100cms"}
Publish ok
Distance (cm): 22.98
ALERT!!
```

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. The 'Browse' tab is selected, displaying a list of devices. The device with ID '12345' is highlighted, and its 'Recent Events' are shown in a table below. The table lists three events, each representing a distance measurement and an alert message sent to the cloud.

Event	Value	Format	Last Received
Data	{"Distance":22.95,"ALERT!!":"Distance less than ...	json	a few seconds ago
Data	{"Distance":79.97,"ALERT!!":"Distance less than ...	json	a few seconds ago
Data	{"Distance":80.04,"ALERT!!":"Distance less than ...	json	a few seconds ago