

Assignment -4

Student name	s.karthickraj
Student rollnumber	821719104014
Maximum mark	2mark

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 :

```
#include <WiFi.h>
#include <PubSubClient.h> void callback(char* subscribetopic, byte* payload,unsigned int
payloadLength);

#define ORG " j3bgcj"
#define DEVICE_TYPE "esp32"#define
DEVICE_ID "1234"
#define TOKEN "12345678" 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++)
  {
  data3 += (char)payload[i];
  }
  Serial.println("data: "+ data3); data3="";
  }

```

Wokwi Link :

<https://wokwi.com/projects/348293960317272660>

Output and Simulation :

The screenshot displays the Wokwi web interface. On the left, the 'sketch.ino' file is open, showing C++ code for an ESP32 microcontroller. The code includes libraries for WiFi and MQTT, defines pins and sensor types, and sets up an MQTT client to connect to IBM Watson IoT Platform. The right side of the interface shows a 'Simulation' window with a visual representation of the ESP32 board, a red LED, and a DHT22 temperature and humidity sensor. Below the simulation, a console window shows the output of the program, indicating successful connection to the WiFi and the MQTT broker.

```
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
8 DHT dht (DHTPIN, DHTTYPE); // creating the instance by passing pin and type of
9
10 void callback(char* topic, byte* payload, unsigned int payloadLength)
11
12 //-----credentials of IBM Accounts-----
13
14 #define ORG "j3bgcj" //IBM ORGANIZATION ID
15 #define DEVICE_TYPE "nodeMCU" //Device type mentioned in IBM Watson IoT Platform
16 #define DEVICE_ID "1234" //Device ID mentioned in IBM Watson IoT Platform
17 #define TOKEN "12345678" //Token
18 String data3;
19 float h, t;
20
21 //----- Customise the above values -----
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
28 char mqttId[] = "4-" ORG "-" DEVICE_TYPE "-" DEVICE_ID "-" j3bgcj - 14
```

Simulation window output:

```
Connecting to ..
WiFi connected
IP address:
10.10.0.2
Reconnecting client to j3bgcj.messaging.internetofthings.ibmcloud.com
..
```

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 top navigation bar includes tabs for 'Browse', 'Action', 'Device Types', and 'Interfaces'. A search bar labeled 'Search by Device ID' is present. The main content area displays a table of devices. The first device, ID 1234, is highlighted. Below the device list, a tabbed interface shows 'Recent Events' 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 lists recent events.

Event	Value	Format	Last Received
status	{"name":"Train1","lat":17.6248626,"lon":78.472...	json	a few seconds ago
status	{"name":"Train1","lat":17.6340889,"lon":78.474...	json	a few seconds ago
status	{"name":"Train1","lat":17.6341908,"lon":78.474...	json	a few seconds ago
status	{"name":"Train1","lat":17.6387448,"lon":78.475...	json	a few seconds ago