

ASSIGNMENT - 4

Date	28 September 2022
Student Name	Harni V
Student Roll Number	917719D028
Maximum Marks	2 marks

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.

Link: <https://wokwi.com/projects/347147230691459666>

Circuit Diagram:

The screenshot displays the Wokwi web interface for a project. On the left, the 'diagram.json' file is open, showing the configuration for an ESP32 devkit and an HC-SR04 ultrasonic sensor. The sensor is configured with a distance of 173 cm. The connections section shows the wiring between the sensor and the ESP32: VCC to VIN (red), GND to GND (black), ECHO to D12 (green), and TRIG to D13 (green). On the right, the 'Simulation' tab is active, showing a visual representation of the circuit. The ESP32 devkit is connected to the HC-SR04 sensor. Below the simulation, the terminal output shows the following sequence of events: 'Connecting to', 'WiFi connected', 'IP address: 10.10.0.2', 'Reconnecting client to k54tgp.messaging.internetofthings.ibmcloud.com', 'iot-2/cmd/command/fmt/String', 'subscribe to cmd OK', 'Sending payload: {"MESSAGE":"ALERT"}', and 'Publish ok'.

```
1 {
2   "version": 1,
3   "author": "Harni V",
4   "editor": "wokwi",
5   "parts": [
6     { "type": "wokwi-esp32-devkit-v1", "id": "esp", "top": -160.68, "x": 100, "y": 100 },
7     {
8       "type": "wokwi-hc-sr04",
9       "id": "ultrasonic1",
10      "top": -147.27,
11      "left": -21.68,
12      "attrs": { "distance": "173" }
13    }
14  ],
15  "connections": [
16    [ "esp:TX0", "$serialMonitor:RX", "", [] ],
17    [ "esp:RX0", "$serialMonitor:TX", "", [] ],
18    [ "ultrasonic1:GND", "esp:GND.2", "black", [ "v105.33", "h219.57" ] ],
19    [ "ultrasonic1:ECHO", "esp:D12", "green", [ "v85.34", "h205.47" ] ],
20    [ "ultrasonic1:TRIG", "esp:D13", "green", [ "v95.73", "h232.95" ] ],
21    [ "ultrasonic1:VCC", "esp:VIN", "red", [ "v112.53", "h241.64" ] ]
22  ]
23 }
```

Connecting to
WiFi connected
IP address:
10.10.0.2
Reconnecting client to k54tgp.messaging.internetofthings.ibmcloud.com
iot-2/cmd/command/fmt/String
subscribe to cmd OK

Sending payload: {"MESSAGE":"ALERT"}
Publish ok

```

#include <WiFi.h>//library for wifi
#include <PubSubClient.h>//library for MQTT
#define TrigPIN 15
#define EchoPIN 4
#define MINDIST 100
void callback(char* subscribetopic, byte* payload, unsigned int payloadLength);
//-----credentials of IBM Accounts-----
#define ORG "k54tgp"//IBM ORGANITION ID
#define DEVICE_TYPE "4545"//Device type mentioned in ibm watson IOT Platform
#define DEVICE_ID "9999"//Device ID mentioned in ibm watson IOT Platform
#define TOKEN "12345678" //Token
String data3;
float h, 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);
  pinMode(TrigPIN, OUTPUT);
  digitalWrite(TrigPIN, LOW);
  pinMode(EchoPIN, INPUT);
  delay(10);
  Serial.println();
  wificonnect();
  mqttconnect();
}

void loop()// Recursive Function
{
  unsigned long t1;
  unsigned long t2;
  unsigned long pulse_Width;
  float distance;

  digitalWrite(TrigPIN, HIGH);
  delayMicroseconds(10);
  digitalWrite(TrigPIN, LOW);

```

```

pulse_Width = pulseIn(EchoPIN,HIGH);

distance= pulse_Width *0.034 / 2;

if(distance<100)
{
  PublishData();
}

delay(1000);
if (!client.loop()) {
  mqttconnect();
}
}
/*...retrieving to Cloud...*/

void PublishData() {
  mqttconnect();//function call for connecting to ibm
  /*
    creating the String in form JSon to update the data to ibm cloud
  */
  String payload = "{\"MESSAGE\":\"ALERT\"}";
  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)
{
}

```

OUTPUT:

The screenshot shows the Wokwi IoT dashboard interface. At the top, there's a navigation bar with 'Browse', 'Action', 'Device Types', and 'Interfaces'. A sidebar on the left contains various icons for device management. The main content area displays details for a device named 'assign_4', which is currently 'Disconnected'. Below this, there's a tabbed interface with 'Identity', 'Device Information', 'Recent Events', 'State', and 'Logs'. The 'Recent Events' tab is active, showing a table of events.

Event	Value	Format	Last Received
Data	{"MESSAGE":"ALERT"}	json	a few seconds ago
Data	{"MESSAGE":"ALERT"}	json	a few seconds ago
Data	{"MESSAGE":"ALERT"}	json	3 minutes ago
Data	{"MESSAGE":"ALERT"}	json	3 minutes ago
Data	{"MESSAGE":"ALERT"}	json	4 minutes ago