**Assignment -4**

**Wokwi and IBM cloud connection**

|  |  |
| --- | --- |
| Student Name | GOKUL A R |
| Student Roll Number | 110119205016 |
| Maximum Marks | 2 Marks |

**Question-1:**

Write code and connections in wokwi for ultrasonic sensor. Whenever distance is less than 100cms send "alert" to IBM cloud and display in device recent events (Upload document with wokwi share link and images of IBM cloud).

**Solution:**

**Wokwi link:** <https://wokwi.com/projects/347570405608784468>

**Code:**

#include <WiFi.h>//library for wifi

#include <PubSubClient.h>//library for MQtt

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

#define ORG "b59mry"//IBM ORGANITION ID

#define DEVICE\_TYPE "Board"//Device type mentioned in ibm watson IOT Platform

#define DEVICE\_ID "1111"//Device ID mentioned in ibm watson IOT Platform

#define TOKEN "12345678"     //Token

String data3;

//-------- 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

#define echoPin 10

#define trigPin 11

float duration;

float distance;

void setup() {

  pinMode(trigPin, OUTPUT);

  pinMode(echoPin, INPUT);

**Serial**.begin(9600);

}

void loop() {

  // Clears the trigPin condition

  digitalWrite(trigPin, LOW);

  delayMicroseconds(2);

  // Sets the trigPin HIGH (ACTIVE) for 10 microseconds

  digitalWrite(trigPin, HIGH);

  delayMicroseconds(10);

  digitalWrite(trigPin, LOW);

  // Reads the echoPin, returns the sound wave travel time in microseconds

  duration = pulseIn(echoPin, HIGH);

  // Calculating the distance

  distance = duration \* 0.034/2; // Speed of sound wave divided by 2 (go and back)

  // Displays the distance on the Serial Monitor

**Serial**.print("Distance: ");

**Serial**.print(distance);

**Serial**.println(" cm");

  if(distance<100)

  {

**Serial**.println("Alert:Object is between 100cm");

  }

}

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

void PublishData(float distance) {

  mqttconnect();//function call for connecting to ibm

  /\*

     creating the String in in form JSon to update the data to ibm cloud

  \*/

  String payload = "{\"distance\":";

  payload += distance;

  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];

  }

**Serial**.println("data: "+ data3);

data3="";

}

