Assignment -4

|  |  |
| --- | --- |
| Assignment Date | 19 September 2022 |
| Student Name | Mr. KRISHANMOORTHY K |
| Student Roll Number | 310119205013 |
| Maximum Marks | 2 Marks |

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.

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

#include <PubSubClient.h>//library for MQtt #define buzzerPin 4

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

//-------credentials of IBM Accounts------

#define ORG "nw3318"//IBM ORGANITION ID

#define DEVICE\_TYPE "123"//Device type mentioned in ibm watson IOT Platform #define DEVICE\_ID "1234567"//Device ID mentioned in ibm watson IOT Platform #define TOKEN "12345678" //Token

String data3; float h, t;

int d;

//-------- 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

long readUltrasonicDistance(int triggerPin, int echoPin)

{

pinMode(triggerPin, OUTPUT); digitalWrite(triggerPin, LOW); delayMicroseconds(2); digitalWrite(triggerPin, HIGH); delayMicroseconds(10); digitalWrite(triggerPin, LOW); pinMode(echoPin, INPUT); return pulseIn(echoPin, HIGH);

}

void setup()// configureing the ESP32

{

**Serial**.begin(115200); pinMode(buzzerPin, OUTPUT); delay(10);

**Serial**.println(); wificonnect(); mqttconnect();

}

void loop()// Recursive Function

{

d=(0.01723 \* readUltrasonicDistance(18, 19));

**Serial**.print("distance:"); **Serial**.println(d);

if(d<100)

{

tone(buzzerPin, 31); delay(1000); noTone(buzzerPin); delay(1000); tone(buzzerPin, 100, 1000); delay(2000); **Serial**.print("buzzer on"); **Serial**.println();

digitalWrite(buzzerPin,HIGH);

}

else

{

**Serial**.print("buzzer off"); **Serial**.println(); digitalWrite(buzzerPin,LOW);

}

PublishData(d);

delay(1000);

if (!client.loop()) { mqttconnect();

}

}

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

void PublishData(int 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); if(data3=="lighton")

{

**Serial**.println(data3); digitalWrite(LED,HIGH);

}

else

{

**Serial**.println(data3); digitalWrite(LED,LOW);

}

data3="";

}



