# Assignment -4

|  |  |
| --- | --- |
| Assignment Date | 31 October 2022 |
| Student Name | Vinmathi k |
| Student Roll Number | 422019106002 |
| Maximum Marks | 2 Marks |

**Question-1:**

Write code and connections in wokwi for ultrasonic sensor. Whenever distance is less than 100 cm send "alert" to IBM cloud and display in device recent events.

WOWKI LINK:

https://wokwi.com/projects/347118252509364819

**Solution:**

#include <WiFi.h> #include <WiFiClient.h>

#include <PubSubClient.h> const int trigPin = 5; const int echoPin = 18;

//define sound speed in cm/uS #define SOUND\_SPEED 0.034

#define CM\_TO\_INCH 0.393701 long duration;

float distanceCm; float distanceInch;

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

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

#define ORG "gs8md6"//IBM ORGANITION ID

#define DEVICE\_TYPE "Assignment-4"//Device type mentioned in ibm watson IOT Platform #define DEVICE\_ID "DuraishanmugamID"//Device ID mentioned in ibm watson IOT Platform #define TOKEN "w7F\*gO8p7gFNvPfX6j" //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/test/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);

void setup() {

**Serial**.begin(115200); // Starts the serial communication pinMode(trigPin, OUTPUT); // Sets the trigPin as an Output pinMode(echoPin, INPUT); // Sets the echoPin as an Input **Serial**.println();

wificonnect(); mqttconnect();

}

void loop() {

// Clears the trigPin digitalWrite(trigPin, LOW); delayMicroseconds(2);

// Sets the trigPin on HIGH state for 10 micro seconds digitalWrite(trigPin, HIGH);

delayMicroseconds(10); digitalWrite(trigPin, LOW);

// Reads the echoPin, returns the sound wave travel time in microseconds duration = pulseIn(echoPin, HIGH);

// Calculate the distance

distanceCm = duration \* SOUND\_SPEED/2;

// Convert to inches

distanceInch = distanceCm \* CM\_TO\_INCH;

// Prints the distance in the Serial Monitor **Serial**.print("Distance (cm): "); **Serial**.println(distanceCm); **Serial**.print("Distance (inch): "); **Serial**.println(distanceInch);

PublishData(distanceCm); delay(1000);

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

}

}

void PublishData(float Cm) { mqttconnect();//function call for connecting to ibm

/\*

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

\*/

String payload = "{\"Distance (cm)\":"; payload += Cm;

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];

}

}

**OUTPUT:**



