ASSIGNMENT – 4

Ultrasonic sensor simulation

Date	29 October 2022
Team ID	PTN2022TMID51402
Project Name	Real- Time River Water Quality Monitoring and Control System
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

CODE:

Sketch.ino

```
#include <WiFi.h>//library for wifi
#include <PubSubClient.h>//library for MQtt

#define ECHO_GPIO 12
#define TRIGGER_GPIO 14
#define MAX_DISTANCE_CM 100 // Maximum of 5 meters
#include "Ultrasonic.h"

Ultrasonic ultrasonic(14, 12);
int distance;

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

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

#define ORG "nnj60r"//IBM ORGANITION ID
#define DEVICE_TYPE "iotdev"//Device type mentioned in ibm watson IOT Platform
#define DEVICE_ID "213432"//Device ID mentioned in ibm watson IOT Platform
#define TOKEN "qHtRu6F*_QH)YEVFh8"      //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);
  delay(10);
  Serial.println();
  wificonnect();
  mqttconnect();
}

void loop()// Recursive Function
{

  distance = ultrasonic.read(CM);
  if(distance < 100){
  Serial.print("Distance in CM: ");
  Serial.println(distance);
  PublishData(distance);
  delay(1000);
  if (!client.loop()) {
    mqttconnect();
  }

  }

  delay(1000);


}

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

void PublishData(float temp) {
  mqttconnect();//function call for connecting to ibm
  /*
     creating the String in in form JSon to update the data to ibm cloud
  */
  String payload = "{\"Alert Distance:\":";
  payload += temp;
  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);
  }
  else
  {
Serial.println(data3);
  }
data3="";
}
```

Ultrasonic.h

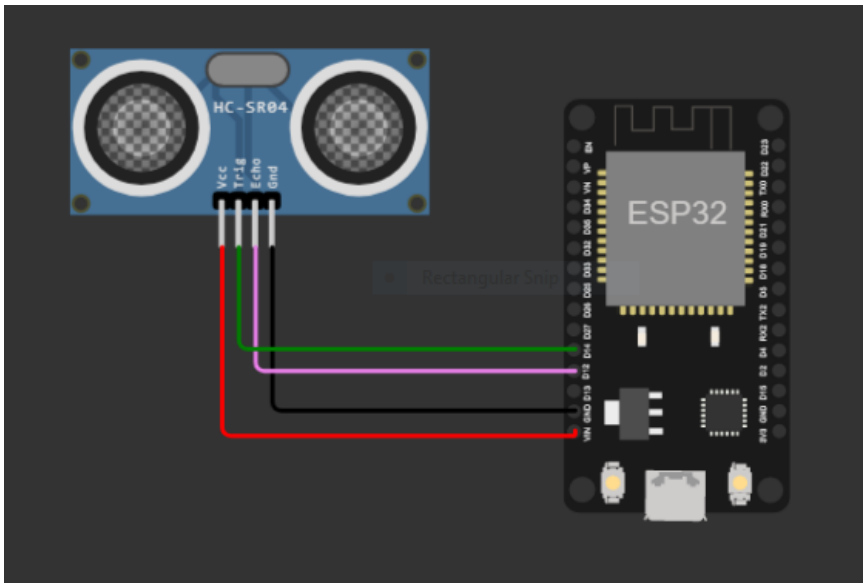
```
#ifndef Ultrasonic_h
#define Ultrasonic_h

/*
 * Values of divisors
 */
#define CM 28
#define INC 71

class Ultrasonic {
  public:
    Ultrasonic(uint8_t sigPin) : Ultrasonic(sigPin, sigPin) {};
    Ultrasonic(uint8_t trigPin, uint8_t echoPin, unsigned long timeOut = 20000UL);
    unsigned int read(uint8_t und = CM);
    unsigned int distanceRead(uint8_t und = CM) __attribute__ ((deprecated ("This
method is deprecated, use read() instead.")));
    void setTimeout(unsigned long timeOut) {timeout = timeOut;}
    void setMaxDistance(unsigned long dist) {timeout = dist*CM*2;}

  private:
    uint8_t trig;
    uint8_t echo;
    boolean threePins = false;
    unsigned long previousMicros;
    unsigned long timeout;
    unsigned int timing();
};

#endif // Ultrasonic_h
```

Ultrasonic.cpp

```
#if ARDUINO >= 100
  #include <Arduino.h>
#else
  #include <WProgram.h>
#endif

#include "Ultrasonic.h"

Ultrasonic::Ultrasonic(uint8_t trigPin, uint8_t echoPin, unsigned long timeOut) {
  trig = trigPin;
  echo = echoPin;
  threePins = trig == echo ? true : false;
  pinMode(trig, OUTPUT);
  pinMode(echo, INPUT);
  timeout = timeOut;
}

unsigned int Ultrasonic::timing() {
  if (threePins)
    pinMode(trig, OUTPUT);

  digitalWrite(trig, LOW);
  delayMicroseconds(2);
  digitalWrite(trig, HIGH);
  delayMicroseconds(10);
  digitalWrite(trig, LOW);

  if (threePins)
    pinMode(trig, INPUT);

  previousMicros = micros();
  while(!digitalRead(echo) && (micros() - previousMicros) <= timeout); // wait for
the echo pin HIGH or timeout
  previousMicros = micros();
  while(digitalRead(echo)  && (micros() - previousMicros) <= timeout); // wait for
the echo pin LOW or timeout

  return micros() - previousMicros; // duration
}

/*
 * If the unit of measure is not passed as a parameter,
 * sby default, it will return the distance in centimeters.
 * To change the default, replace CM by INC.
 */
unsigned int Ultrasonic::read(uint8_t und) {
  return timing() / und / 2;  //distance by divisor
}

/*
```

```
 * This method is too verbal, so, it's deprecated.
 * Use read() instead.
 */
unsigned int Ultrasonic::distanceRead(uint8_t und) {
  return read(und);
}
```

CIRCUIT DIAGRAM:



Wokwi simulation link:

https://wokwi.com/projects/348318092902793810

WOKWI OUTPUT:

```
Sending payload: {"Alert Distance:":69.00}
Publish ok
Distance in CM: 69
Sending payload: {"Alert Distance:":69.00}
Publish ok
Distance in CM: 72
Sending payload: {"Alert Distance:":72.00}
Publish ok
Distance in CM: 9
Sending payload: {"Alert Distance:":9.00}
Publish ok
Distance in CM: 9
Sending payload: {"Alert Distance:":9.00}
Publish ok
Distance in CM: 9
Sending payload: {"Alert Distance:":9.00}
Publish ok
```

IBM CLOUD OUTPUT: