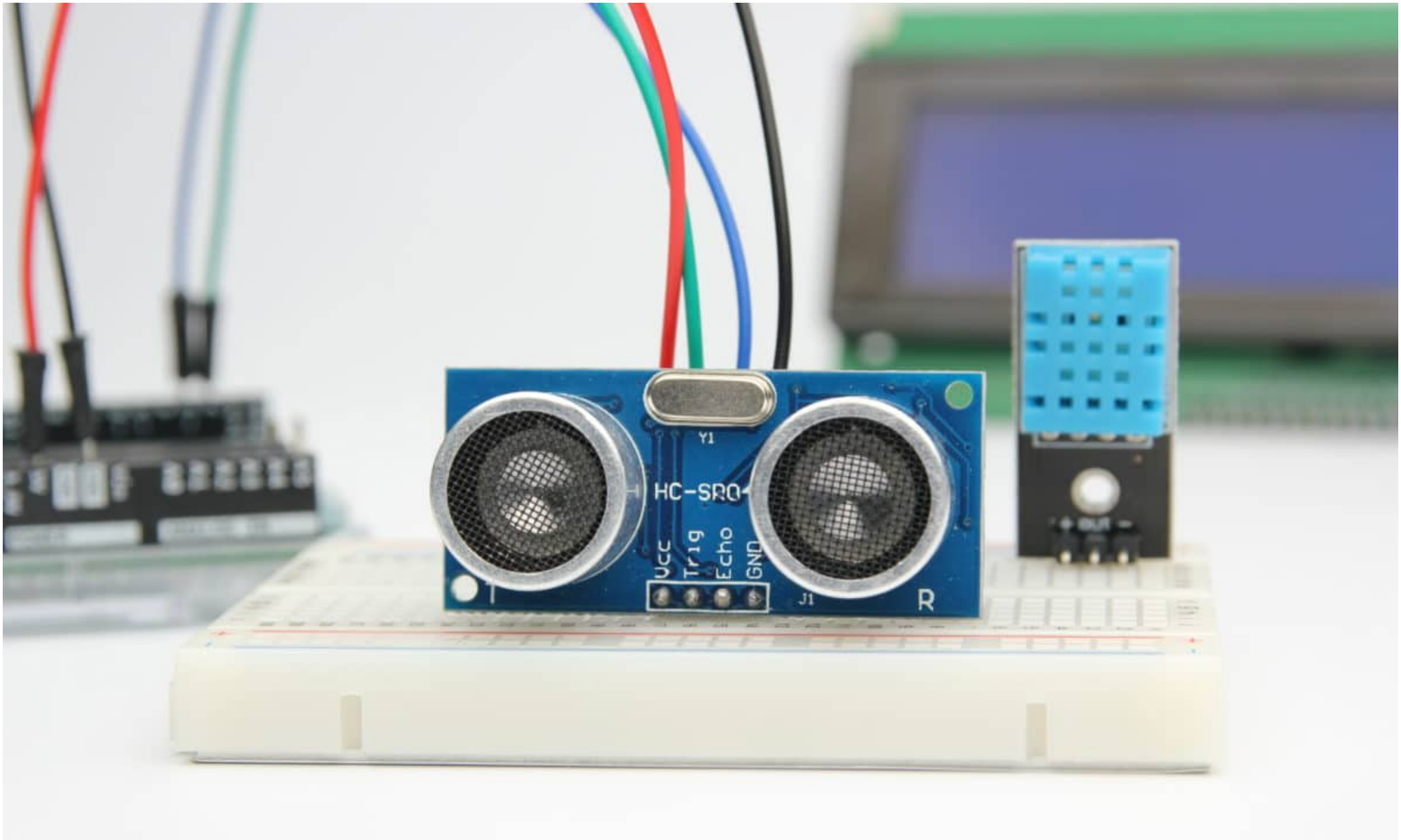
Makerguides.com

How to use a HC-SR04 Ultrasonic Distance Sensor with Arduino

Written by Benne de Bakker (https://www.makerguides.com/author/benne-de-bakker/)


Y1
HC-SR04
Vcc
Trig
Echo
GND
J1
R

The HC-SR04 is an inexpensive, easy to use distance sensor, with a range of 2 to 400cm. It is commonly used in obstacle avoiding robots and automation projects. And did you know that your car uses similar sensors in the parking assist system?

In this tutorial you will learn how the sensor works and how to use it with Arduino. I have included 5 examples with a wiring diagram and code so you can start experimenting with your sensor. We will first look at an example that does not use an Arduino library. Next, I will show you how you can use the **NewPing** library to create more compact code.

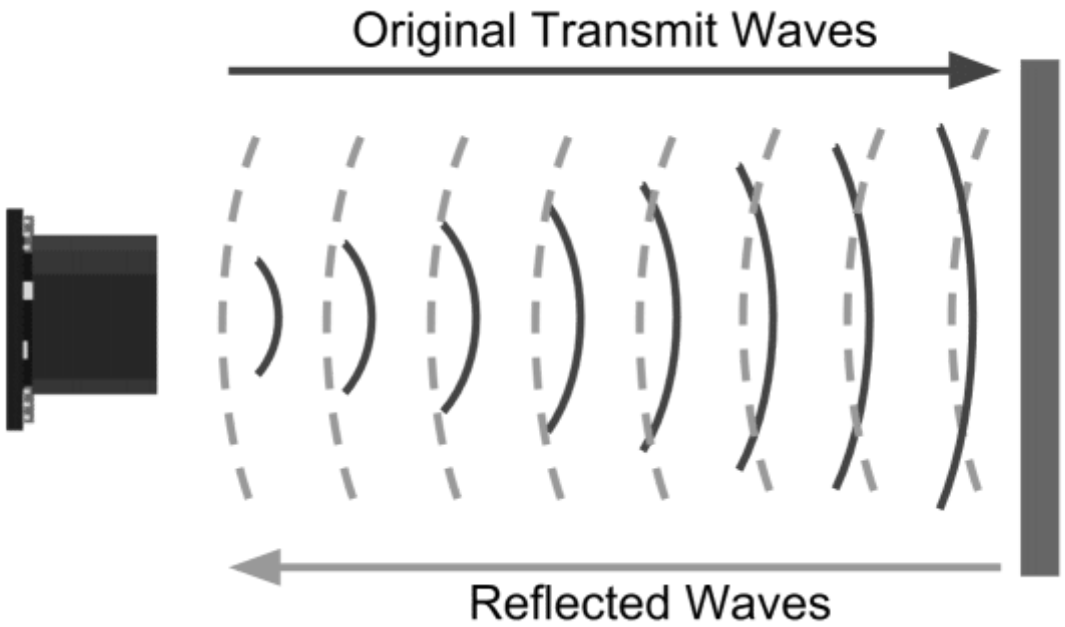
Cheap ultrasonic distance/proximity sensors are great but in some projects you might need a waterproof sensor like the JSN-SR04T or an IR sensor that isn't influenced by temperature changes. In that case, the articles below might be useful:

- Waterproof JSN-SR04T Ultrasonic Distance Sensor with Arduino Tutorial (https://www.makerguides.com/jsn-sr04t-arduino-tutorial/)
- How to use a SHARP GP2Y0A21YK0F IR Distance Sensor with Arduino (https://www.makerguides.com/sharp-gp2y0a21yk0f-ir-distance-sensor-arduino-tutorial/)
- How to use a SHARP GP2Y0A710K0F IR Distance Sensor with Arduino (https://www.makerguides.com/sharp-gp2y0a710k0f-ir-distance-sensor-arduino-tutorial/)
- MaxBotix MB7389 weather-resistant distance sensor tutorial (https://www.makerguides.com/maxbotix-mb7389-arduino-tutorial/)

- MaxBotix MB1240 ultrasonic distance sensor Arduino tutorial (https://www.makerguides.com/maxbotix-mb1240-arduino-tutorial/)

How does an ultrasonic distance sensor work?

Ultrasonic sensors work by emitting sound waves with a frequency that is too high for a human to hear. These sound waves travel through the air with the speed of sound, roughly 343 m/s. If there is an object in front of the sensor, the sound waves get reflected back and the receiver of the ultrasonic sensor detects them. By measuring how much time passed between sending and receiving the sound waves, the distance between the sensor and the object can be calculated.



Ultrasonic distance sensors working principle. Source: https://www.maxbotix.com/ (https://www.maxbotix.com/)

At 20°C the speed of sound is roughly 343 m/s or 0.034 cm/µs. Let's say that the time between sending and receiving the sound waves is 2000 microseconds. If you multiply the speed of sound by the time the sound waves traveled, you get the distance that the sound waves traveled.

$$\text{Distance} = \text{Speed} \times \text{Time}$$

But that is not the result we are looking for. The distance between the sensor and the object is actually only half this distance because the sound waves traveled from the sensor to the object and back from the object to the sensor. So you need to divide the result by two.

$$\text{Distance (cm)} = \text{Speed of sound (cm/µs)} \times \text{Time (µs)} / 2$$

And so for the example this becomes:

$$\text{Distance (cm)} = 0.0343 \text{ (cm/µs)} \times 2000 \text{ (µs)} / 2 = 34.3 \text{ cm}$$

Temperature dependence of the speed of the speed of sound

The speed of sound actually depends strongly on temperature and to a far lesser degree on the humidity of the air. Wikipedia states that the speed of sound (https://en.wikipedia.org/wiki/Speed_of_sound) increases with roughly 0.6 m/s per degree Celsius. For most cases at 20°C you can just use 343 m/s but if you want to get more accurate readings, you can calculate the speed of sound with the following formula:

$$V \text{ (m/s)} = 331.3 + (0.606 \times T)$$

$$V = \text{Speed of sound (m/s)}$$
$$T = \text{Air Temperature (°C)}$$

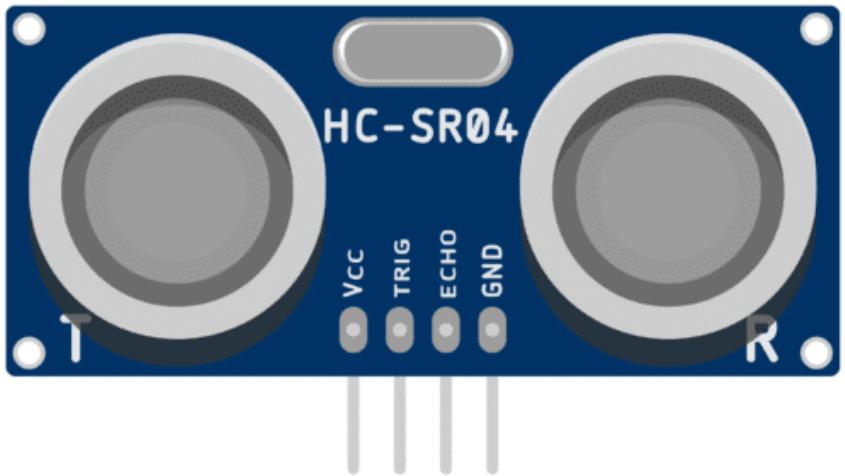
This formula doesn't include the humidity since its effect on the speed of sound is only very small.

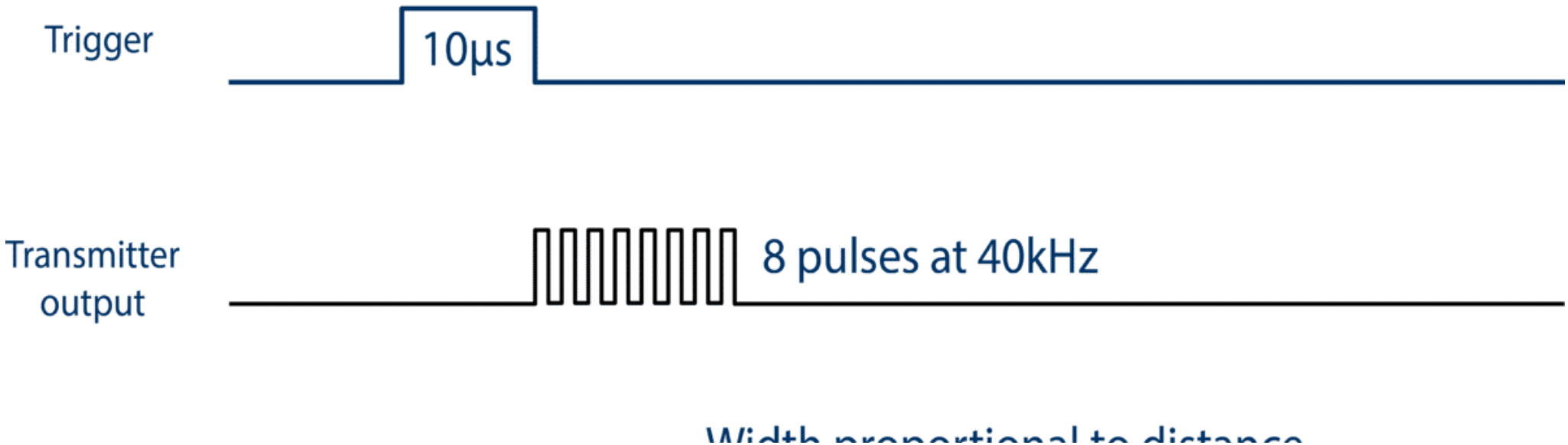
Below you can find a tutorial on how to use a DHT11 (https://amzn.to/2RlYfdw) temperature and humidity sensor to calibrate the speed of sound and get a more accurate distance reading with the HC-SR04.

How the HC-SR04 works

At the front of the HC-SR04 (https://amzn.to/34TdZhV) sensor you can find two silver cylinders (ultrasonic transducers), one is the transmitter of the sound waves and the other is the receiver. To let the sensor generate a sonic burst, you need to set the Trig pin high for at least 10 µs. The sensor then creates an 8 cycle burst of ultrasound at 40 kHz.

This sonic burst travels at the speed of sound, bounces back and gets received by the receiver of the sensor. The Echo pin then outputs the time that the sound waves traveled in microseconds.


HC-SR04
Vcc
TRIG
ECHO
GND
T
R


Trigger
10µs
Transmitter output
8 pulses at 40kHz
Width proportional to distance

You can use the `pulseIn()` function in the Arduino code to read the length of the pulse from the Echo pin. After that, you can use the formula mentioned above to calculate the distance between the sensor and the object.

HC-SR04 Specifications

Operating voltage	5 V
Operating current	15 mA
Frequency	40 kHz
Measuring range	2 – 400 cm
Resolution	3 mm
Measuring angle	15 degrees
Trigger input signal	10 µs high pulse
Cost	Check price (https://amzn.to/2HCqxjs)

For more information you can check out the datasheet here.

HC-SR04 Datasheet **(https://www.makerguides.com/wp-content/uploads/2019/02/HCSR04-Datasheet.pdf)**

Things used in this tutorial:

To follow this tutorial you need the following components:

Hardware components

HC-SR04 sensor (https://amzn.to/2REhpkT)	x 1	Amazon (https://amzn.to/2REhpkT)
Arduino UNO R3 (https://amzn.to/2L9w4SS) You can also use other microcontrollers like an Arduino Mega (https://amzn.to/2J3gbuz) or esp32.	x 1	Amazon (https://amzn.to/2L9w4SS)
Breadboard (https://amzn.to/2MQHicc) I highly recommend to buy at least 1 good quality breadboard like the BusBoard Prototype Systems BB400 (https://amzn.to/2MQHicc) or BB830 (https://amzn.to/2GM261Q).	x 1	Amazon (https://amzn.to/2MQHicc)
Jumper wires (https://amzn.to/2DCt2R8)	~ 10	Amazon (https://amzn.to/2DCt2R8)

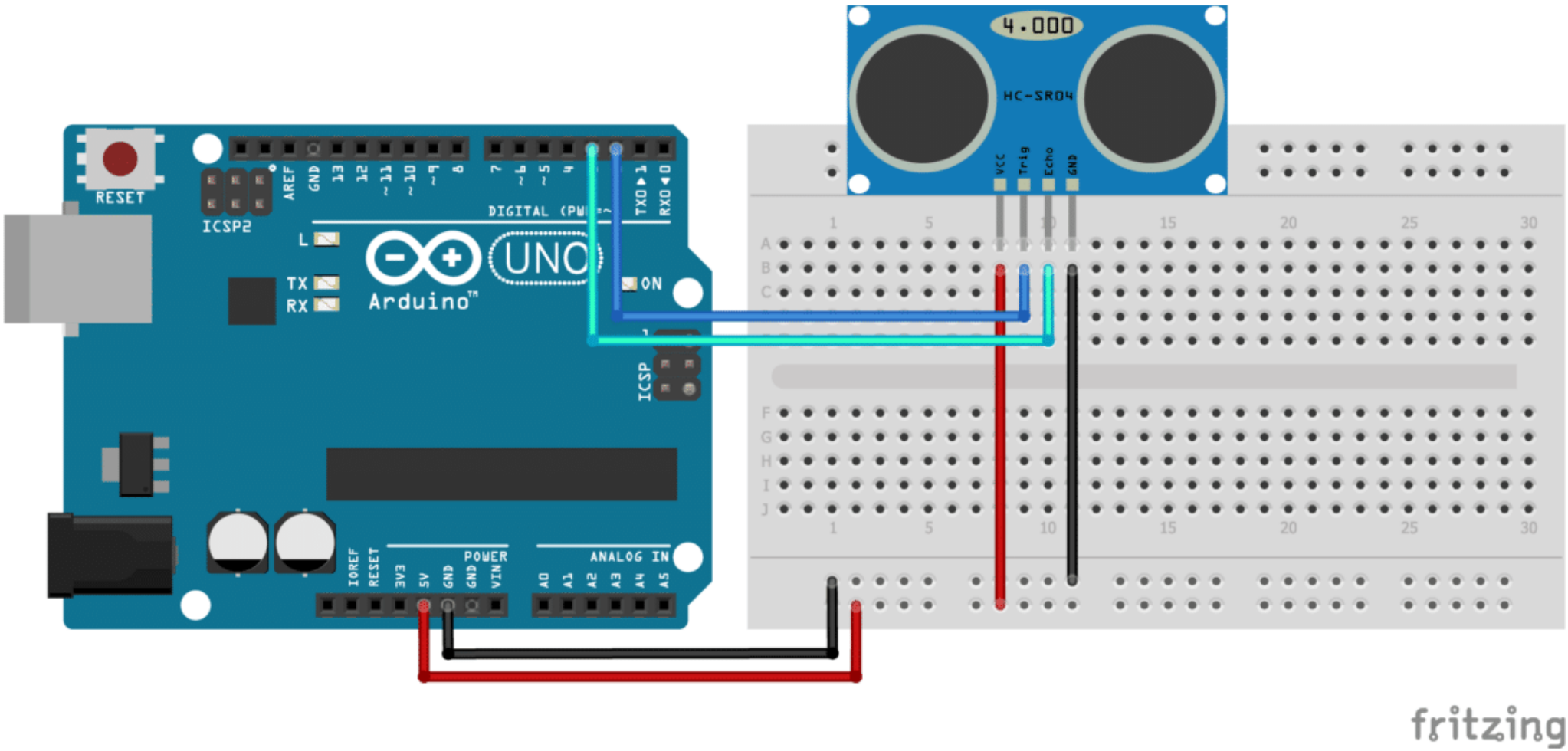
USB Type-B cable (https://amzn.to/2GR73aX)	x 1	Amazon (https://amzn.to/2GR73aX)
DHT11 sensor (https://amzn.to/2RIYfdw) (optional)	x 1	Amazon (https://amzn.to/2RIYfdw)
2004 I2C LCD (https://amzn.to/2srlysl) (optional) 1602 I2C LCD (https://amzn.to/2HjxKoQ) can also be used.	x 1	Amazon (https://amzn.to/2srlysl)

Software

Arduino IDE (https://www.arduino.cc/en/Main/Software)		

Wiring – Connecting HC-SR04 to Arduino UNO

The wiring diagram below shows you how to connect the HC-SR04 sensor to the Arduino.



HC-SR04 with Arduino wiring diagram

The code examples below use digital pin 2 and 3 for the trigger and echo pin, but of course you can change this to any digital pin you want.

HC-SR04 Connections

HC-SR04	Arduino
VCC	5 V
Trig	Pin 2
Echo	Pin 3
GND	GND

Example code for HC-SR04 with Arduino

Now that you have wired up the sensor it is time to connect the Arduino to the computer and upload some code. You can upload the following example code to your Arduino using the Arduino IDE (https://www.arduino.cc/en/main/software). Next, I will explain you how the code works.

```
1.  /* Example code for HC-SR04 ultrasonic distance sensor with Arduino. No library required. More info:
    https://www.makerguides.com */
2.
```

```
// Define Trig and Echo pin:
#define trigPin 2
#define echoPin 3

// Define variables:
long duration;
int distance;

void setup() {
  // Define inputs and outputs:
  pinMode(trigPin, OUTPUT);
  pinMode(echoPin, INPUT);

  //Begin Serial communication at a baudrate of 9600:
  Serial.begin(9600);
}

void loop() {
  // Clear the trigPin by setting it LOW:
  digitalWrite(trigPin, LOW);
  delayMicroseconds(5);

  // Trigger the sensor by setting the trigPin high for 10 microseconds:
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);

  // Read the echoPin, pulseIn() returns the duration (length of the pulse) in microseconds:
  duration = pulseIn(echoPin, HIGH);
  // Calculate the distance:
```

```
33.    distance= duration*0.034/2;
34.
35.    // Print the distance on the Serial Monitor (Ctrl+Shift+M):
36.    Serial.print("Distance = ");
37.    Serial.print(distance);
38.    Serial.println(" cm");
39.
40.    delay(50);
41.  }
```

How the code works

First, the trigger pin and the echo pin are defined. I call them `trigPin` and `EchoPin`. The trigger pin is connected to digital pin 2 and the echo pin to digital pin 3 on the Arduino.

The statement `#define` is used to give a name to a constant value. The compiler will replace any references to this constant with the defined value when the the program is compiled. So everywhere you mention `trigPin`, the compiler will replace it with the value 2 when the program is compiled.

```
3.  // Define Trig and Echo pin:
4.  #define trigPin 2
5.  #define echoPin 3
```

Next I defined two variables: `duration` and `distance`. Duration stores the time between sending and receiving the sound waves. The distance variable is used to store the calculated distance.

```
// Define variables:
long duration;
int distance;
```

In the `setup()`, you start by setting the trigPin as an output and the echoPin as an input. Next you initialize serial communication at a baud rate of 9600. Later you will display the measured distance in the serial monitor, which can be accessed with Ctrl+Shift+M or Tools > Serial Monitor. Make sure the baud rate is also set to 9600 in the serial monitor.

```
void setup() {
  // Define inputs and outputs:
  pinMode(trigPin, OUTPUT);
  pinMode(echoPin, INPUT);

  //Begin Serial communication at a baudrate of 9600:
  Serial.begin(9600);
}
```

In the `loop()`, you trigger the sensor by setting the trigPin HIGH for 10 µs. Note that to get a clean signal you start by clearing the trigPin by setting it LOW for 5 microseconds.

```
void loop() {
  // Clear the trigPin by setting it LOW:
  digitalWrite(trigPin, LOW);
```

```
    delayMicroseconds(5);

    // Trigger the sensor by setting the trigPin high for 10 microseconds:
    digitalWrite(trigPin, HIGH);
    delayMicroseconds(10);
    digitalWrite(trigPin, LOW);
```

Next, you need to read the length of the pulse sent by the echoPin. I use the function `pulseIn()` for this. This function waits for the pin to go from LOW to HIGH, starts timing, then waits for the pin to go LOW and stops timing.

After that you calculate the distance by using the formula mentioned in the introduction of this tutorial.

```
    // Read the echoPin, pulseIn() returns the duration (length of the pulse) in microseconds:
    duration = pulseIn(echoPin, HIGH);
    // Calculate the distance:
    distance= duration*0.034/2;
```

Finally, print the calculated distance in the serial monitor.

```
    // Print the distance on the Serial Monitor (Ctrl+Shift+M):
    Serial.print("Distance = ");
    Serial.print(distance);
    Serial.println(" cm");

    delay(50);
}
```

Example code HC-SR04 with Arduino and NewPing library

The **NewPing** library written by Tim Eckel can be used with many ultrasonic distance sensors. The latest version of this library can be downloaded here on bitbucket.org. (https://bitbucket.org/teckel12/arduino-new-ping/downloads/) You might notice that the code below, which uses the NewPing library, is a lot shorter than the code we used before. Besides that, the NewPing library does include some other nice features. It allows you to set a max distance to read, it won't lag for a full second when no echo is received and it has a built-in median filter.

NewPing_v1.9.1.zip **(https://www.makerguides.com/wp-content/uploads/2019/02/NewPing_v1.9.1.zip)**

You can install the library by going to **Sketch > Include Library > Add .ZIP Library** in the Arduino IDE.

The library does include some examples that you can use, but you will have to modify them to match your hardware setup. I have included a modified example code below that can be used with the same wiring setup as before.

```
/* HC-SR04 ultrasonic distance sensor with NewPing library example code. More info: www.makerguides.com */

// Include the library:
#include <NewPing.h>

// Define pins and max distance:
#define trigPin  2
```

```
#define echoPin  3
#define MAX_DISTANCE 350 // Maximum distance we want to ping for (in centimeters). Maximum sensor distance is rated at 400-500cm.

NewPing sonar(trigPin, echoPin, MAX_DISTANCE); // NewPing setup of pins and maximum distance.
float duration, distance;

void setup() {
  Serial.begin(9600); // Open serial monitor at 9600 baud to see ping results.
}

void loop() {
  delay(50); // Wait 50ms between pings (about 20 pings/sec). 29ms should be the shortest delay between pings.

  duration = sonar.ping();
  distance = (duration / 2) * 0.0343;

  Serial.print("Distance = ");
  Serial.print(distance); // Distance will be 0 when out of set max range.
  Serial.println(" cm");
}
```

You can also use `distance = sonar.ping_cm()` or `distance = sonar.ping_in()` which returns the measured distance in whole centimeters or inches. With this function you do not need to take a duration measurement and calculate the distance.

Interfacing ultrasonic sensors in 3 pin mode

The NewPing library also makes it easy to interface with ultrasonic sensors while using only 1 I/O pin. This can be handy if you have very few I/O pins available or if you want to use a 3 pin ultrasonic sensor like the Parallax Ping (https://amzn.to/2D654fR).

To create a 3 pin setup (GND, 5V and SIG) you have to connect both the trigger pin and the echo pin to the same digital pin on the Arduino. In the code, the only thing you have to change is line 6-7 and define the same pin for both the trigPin and the echoPin. For example digital pin 2.

```
6.  //Define Trig and Echo pin
7.  #define trigPin 2
8.  #define echoPin 2
```

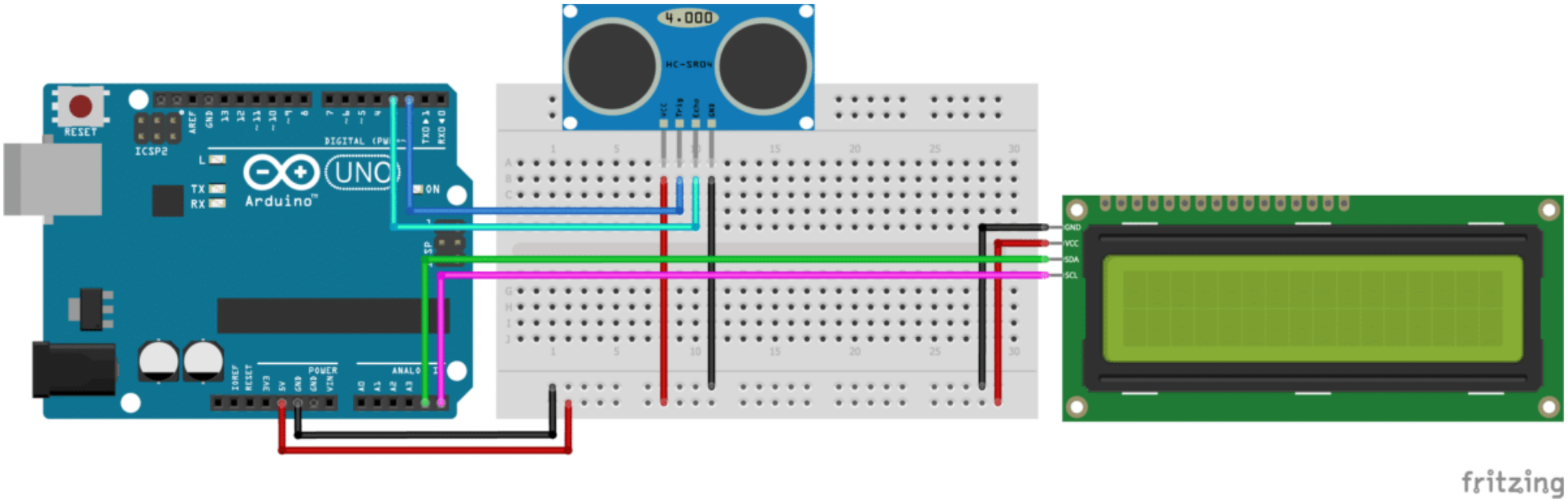
How to use ping_median() digital filter

The main thing I like about the NewPing library is that it has a built-in median filter. This filter can greatly improve the accuracy of your HC-SR04 readings. The `ping_median()` function takes many duration measurements in a row, throws away the out of range readings and then averages the remaining ones. By default it will take 5 readings but you can specify how many it should take. Replace line 19 with below lines.

```
21.  int iterations = 5;
22.  duration = sonar.ping_median(iterations);
```

Example code HC-SR04 with I2C LCD and Arduino

To display the measured distance on a 2004 or 1602 I2C LCD (https://amzn.to/2srlysl), all you have to do is make the following connections and upload the code below. The HC-SR04 sensor is connected in the same way as before.



HC-SR04 with Arduino and I2C LCD wiring diagram.

I2C LCD Connections

I2C LCD	Arduino
GND	GND
VCC	5 V
SDA	A4
SCL	A5

If you are not using an Arduino Uno, the SDA and SCL pins can be at a different location. An Arduino UNO with the R3 layout (1.0 pinout), also has the SDA (data line) and SCL (clock line) pin headers close to the AREF pin. Check the table below for more details.

I2C pin locations for different Arduino boards

Board	SDA	SCL
Arduino Uno (https://amzn.to/2N80aGy)	A4	A5
Arduino Nano (https://amzn.to/2N7I9bg)	A4	A5
Arduino Micro (https://amzn.to/32zK3oQ)	2	3
Arduino Mega 2560 (https://amzn.to/2N6aheY)	20	21
Arduino Leonardo (https://amzn.to/2HZdDMD)	2	3
Arduino Due (https://amzn.to/2NTFtOn)	20	21

The code uses the **LiquidCrystal_I2C** library, which you can download here on GitHub (https://github.com/johnrickman/LiquidCrystal_I2C). Make sure that you have this exact library installed! It also includes the **Wire.h** library, which allows you to communicate with I2C devices. This library should come pre-installed with the Arduino IDE.

LiquidCrystal_I2C-master.zip **(https://www.makerguides.com/wp-content/uploads/2019/02/LiquidCrystal_I2C-master.zip)**

If you want to learn more about how to control a I2C LCD with Arduino, you can check out the full tutorial here.

- How to control a character I2C LCD with Arduino (https://www.makerguides.com/character-i2c-lcd-arduino-tutorial/)

You can click the button in the top right corner of the code field to open the complete code in a new window.

```
/*
HC-SR04 ultrasonic distance sensor with Arduino and I2C LCD example code.
More info: https://www.makerguides.com
*/

// Include the libraries:
#include <Wire.h>
#include <LiquidCrystal_I2C.h>


// Define Trig and Echo pin:
#define trigPin 2
#define echoPin 3
```

```
14.
15.  // Define SDA and SCL pin for LCD:
16.  #define SDAPin A4 // Data pin
17.  #define SCLPin A5 // Clock pin
18.
19.  // Connect to LCD via I2C, default address 0x27 (A0-A2 not jumpered):
20.  LiquidCrystal_I2C lcd = LiquidCrystal_I2C(0x27,20,4); //Change to (0x27,16,2) for 1602 LCD
21.
22.  // Define variables:
23.  long duration;
24.  int distance;
25.
26.  void setup() {
27.    // Define inputs and outputs:
28.    pinMode(trigPin, OUTPUT);
29.    pinMode(echoPin, INPUT);
30.
31.    // Initiate the LCD:
32.    lcd.init();
33.    lcd.backlight();
34.  }
35.
36.  void loop() {
37.    // Clear the trigPin by setting it LOW:
38.    digitalWrite(trigPin, LOW);
39.    delayMicroseconds(5);
40.
41.    // Trigger the sensor by setting the trigPin high for 10 microseconds:
42.    digitalWrite(trigPin, HIGH);
43.    delayMicroseconds(10);
```

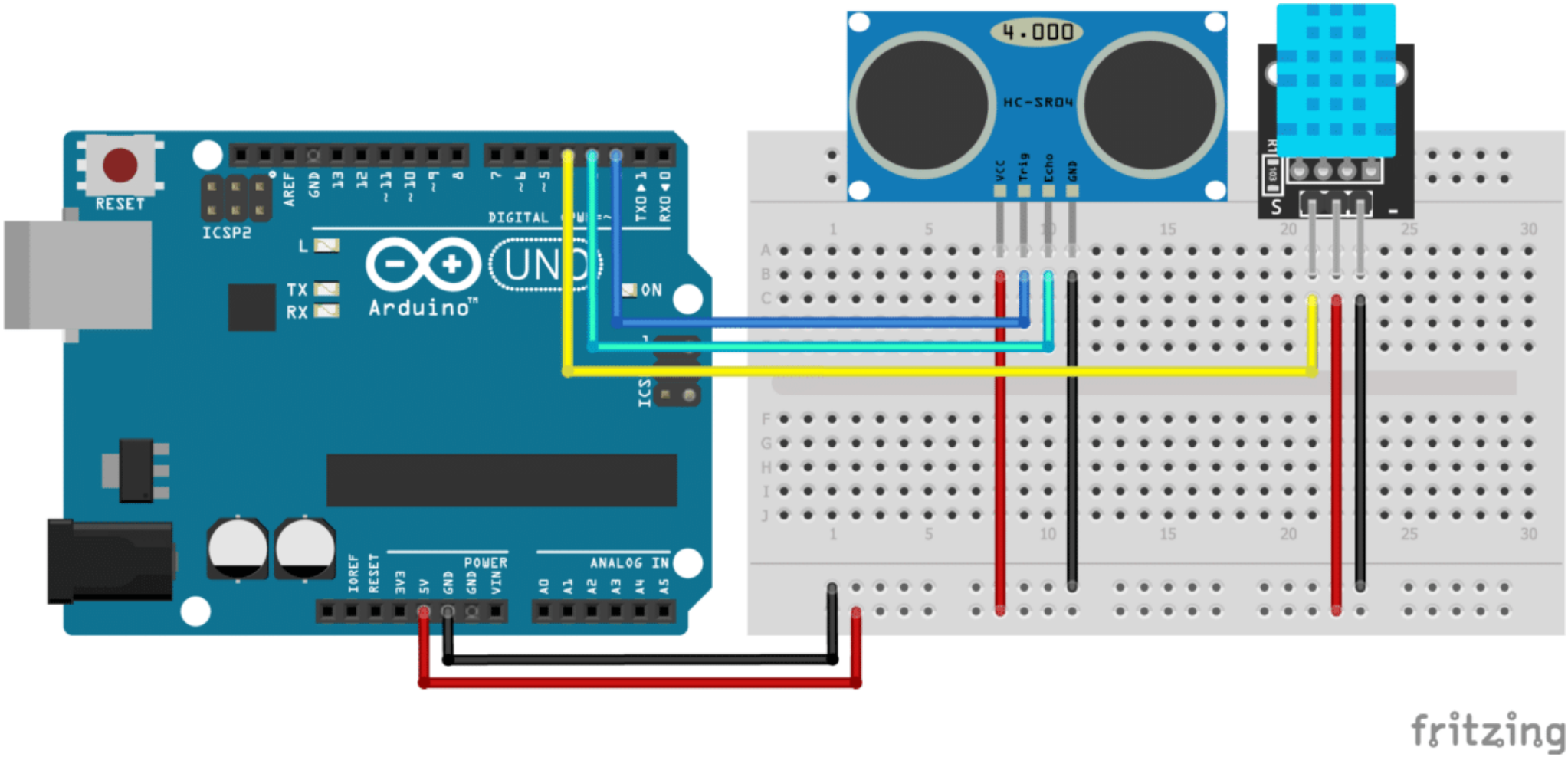
```
  digitalWrite(trigPin, LOW);

  // Read the echoPin. This returns the duration (length of the pulse) in microseconds:
  duration = pulseIn(echoPin, HIGH);

  // Calculate the distance:
  distance = duration*0.034/2;

  // Display the distance on the LCD:
  lcd.setCursor(0,0); // Set the cursor to column 1, line 1 (counting starts at zero)
  lcd.print("Distance = "); // Prints string "Display = " on the LCD
  lcd.print(distance); // Prints the measured distance
  lcd.print(" cm  "); // Prints "cm" on the LCD, extra spaces are needed to clear previously displayed characters

  delay(50);
}
```

Note that I used a 20 x 4 LCD display. If you have a different size LCD (16 x 2 is also common) you need to change line 20 to `LiquidCrystal_I2C lcd(0x27,16,2);` . If your LCD doesn't have the default I2C address, 0x27, check out the complete I2C tutorial where I explain how you can find out what the address is.

Example code HC-SR04 with DHT11 temperature sensor and Arduino

As mentioned earlier, the speed of sound strongly depends on the air temperature. If you want to measure long distances (3-4 m) it can be a good idea to add a DHT11 or DHT22 (https://amzn.to/2FMlC00) temperature and humidity sensor to your setup. This will allow you to calibrate the speed of sound in real time and thereby increase the accuracy of your measurements.

Adding a DHT11 sensor is really simple. The wiring diagram below shows you which connections you need to make. Note that I am using a DHT11 with breakout board (https://amzn.to/2RWFfsf), so I only need to wire up 3 pins. Be sure to check the label of the sensor, **the order of the pins can be different** depending on the manufacturer. The HC-SR04 sensor is connected in the same way as before.



HC-SR04 with Arduino and DHT11 wiring diagram.

DHT11 Connections

DHT11	Arduino
VCC (+)	5 V
Signal (s)	Pin 4
GND (-)	GND

The code below uses the **Adafruit DHT Humidity & Temperature Sensor** library which you can download here on GitHub (https://github.com/adafruit/DHT-sensor-library). This library only works if you also have the **Adafruit_Sensor** library installed, which is also available on GitHub (https://github.com/adafruit/Adafruit_Sensor). You can also download the two libraries by clicking on the buttons below:

DHT-sensor-library-master.zip
(https://www.makerguides.com/wp-content/uploads/2019/02/DHT-sensor-library-master.zip)

Adafruit_Sensor-master.zip **(https://www.makerguides.com/wp-content/uploads/2019/02/Adafruit_Sensor-master.zip)**

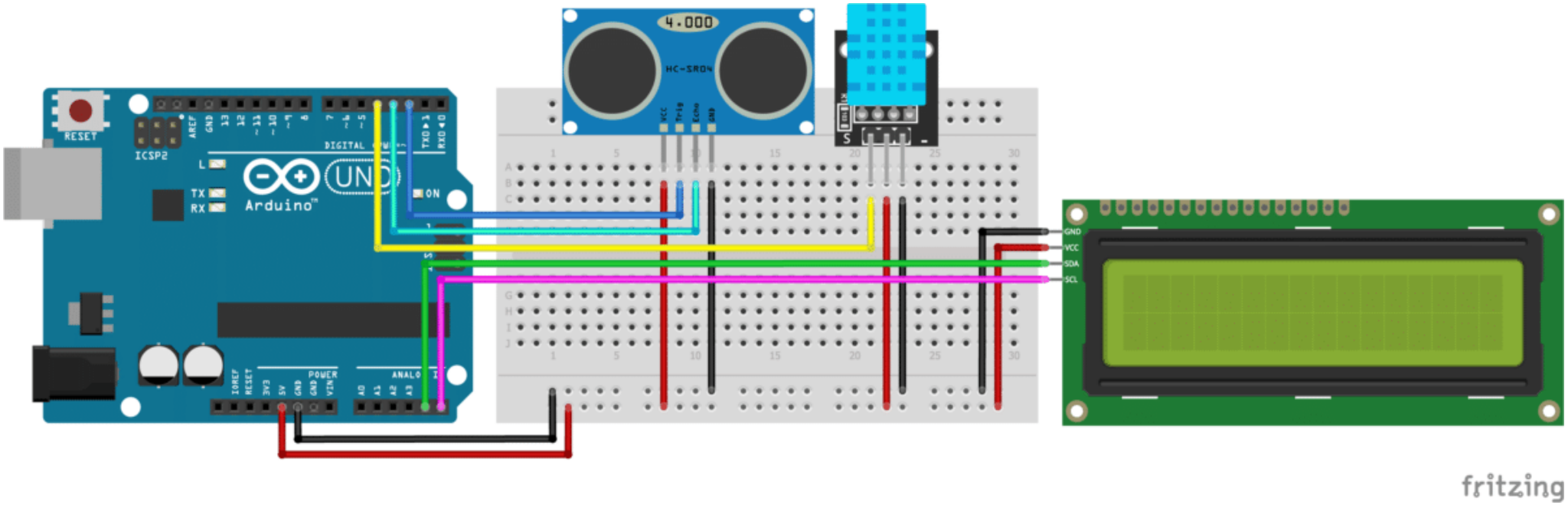
You can click the button in the top right corner of the code field to open the complete code in a new window.

```
/*
HC-SR04 ultrasonic distance sensor with DHT11 and Arduino example code.
More info: https://www.makerguides.com
*/

// Include Adafruit sensor library:
#include <Adafruit_Sensor.h> //https://github.com/adafruit/Adafruit_Sensor
// Include Adafruit DHT library:
#include <DHT.h> //https://github.com/adafruit/DHT-sensor-library

// Define Trig pin, Echo pin and DHTPin:
#define trigPin 2
#define echoPin 3
#define DHTPin 4

// Define DHT sensor type:
#define DHTType DHT11

// Define variables:
long duration;
int distance;
float speedofsound;

// Create a DHT sensor object:
DHT dht = DHT(DHTPin,DHTType);

void setup() {
  // Define inputs and outputs:
  pinMode(trigPin, OUTPUT);
  pinMode(echoPin, INPUT);
```

```
31.
32.    dht.begin();
33.
34.    // Begin Serial communication:
35.    Serial.begin(9600); // Starts the serial communication
36.  }
37.
38.  void loop() {
39.    //  Clear the trigPin by setting it LOW:
40.    digitalWrite(trigPin, LOW);
41.    delayMicroseconds(5);
42.
43.    // Trigger the sensor by setting the trigPin high for 10 microseconds:
44.    digitalWrite(trigPin, HIGH);
45.    delayMicroseconds(10);
46.    digitalWrite(trigPin, LOW);
47.
48.    // Read the echoPin. This returns the duration (length of the pulse) in microseconds:
49.    duration = pulseIn(echoPin, HIGH);
50.
51.    // Read the temperature:
52.    float temperature = dht.readTemperature();
53.
54.    // Calculate speed of sound in m/s:
55.    speedofsound = 331.3+(0.606*temperature);
56.
57.    // Calculate the distance in cm:
58.    distance = duration*(speedofsound/10000)/2;
59.
60.    // Print the distance and temperature on the Serial Monitor:
```

```
61.    Serial.print("Temperature = ");
62.    Serial.print(temperature);
63.    Serial.print(" Celsius");
64.    Serial.print(", Distance = ");
65.    Serial.print(distance);
66.    Serial.println("cm");
67.    delay(100);
68.  }
```

Example code HC-SR04 with DHT11 and I2C LCD



HC-SR04 with Arduino, DHT11 and I2C LCD wiring diagram.

The code below can be used to combine all 3 examples above. It displays both the temperature, the speed of sound and the measured distance on the LCD.

You can click the button in the top right corner of the code field to open the complete code in a new window.

```
/*
HC-SR04 ultrasonic distance sensor with DHT11, I2C LCD and Arduino example code.
More info: https://www.makerguides.com
*/

```

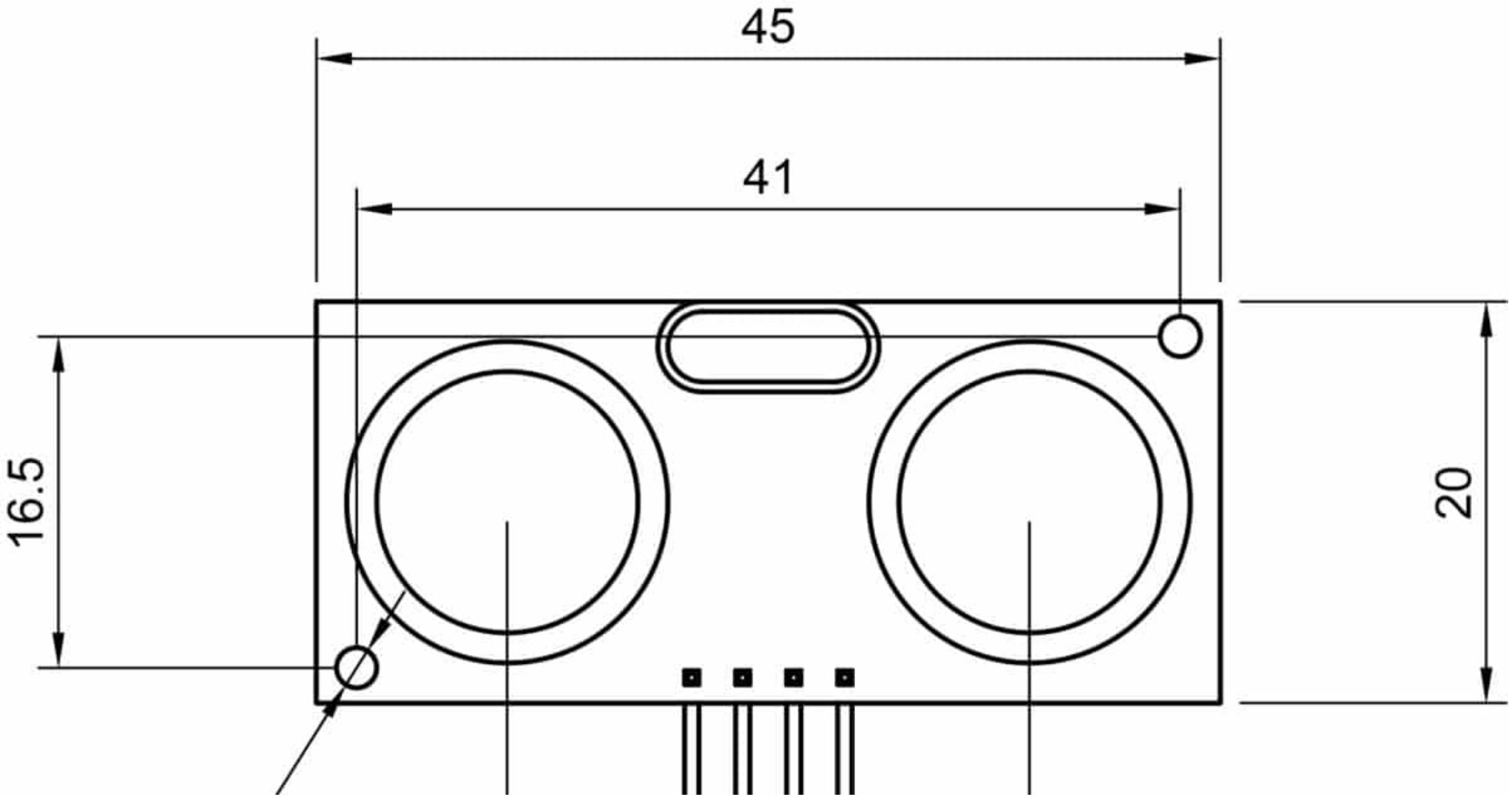
```
// Include Adafruit sensor library:
#include <Adafruit_Sensor.h> // https://github.com/adafruit/Adafruit_Sensor
// Include Adafruit DHT library:
#include <DHT.h> // https://github.com/adafruit/DHT-sensor-library
#include <Wire.h> // Library for I2C communication
#include <LiquidCrystal_I2C.h> // Library for LCD

// Define Trig pin, Echo pin and DHTPin:
#define trigPin 2
#define echoPin 3
#define DHTPin 4

// Define SDA and SCL pin from LCD:
#define SDAPin A4 // Data pin
#define SCLPin A5 // Clock pin

// Connect to LCD via i2c, default address 0x27 (A0-A2 not jumpered):
LiquidCrystal_I2C lcd = LiquidCrystal_I2C(0x27,20,4);

// Define DHT sensor type:
#define DHTType DHT11

// Define variables:
long duration;
int distance;
float speedofsound;

// Create a DHT sensor object:
DHT dht = DHT(DHTPin,DHTType);

```

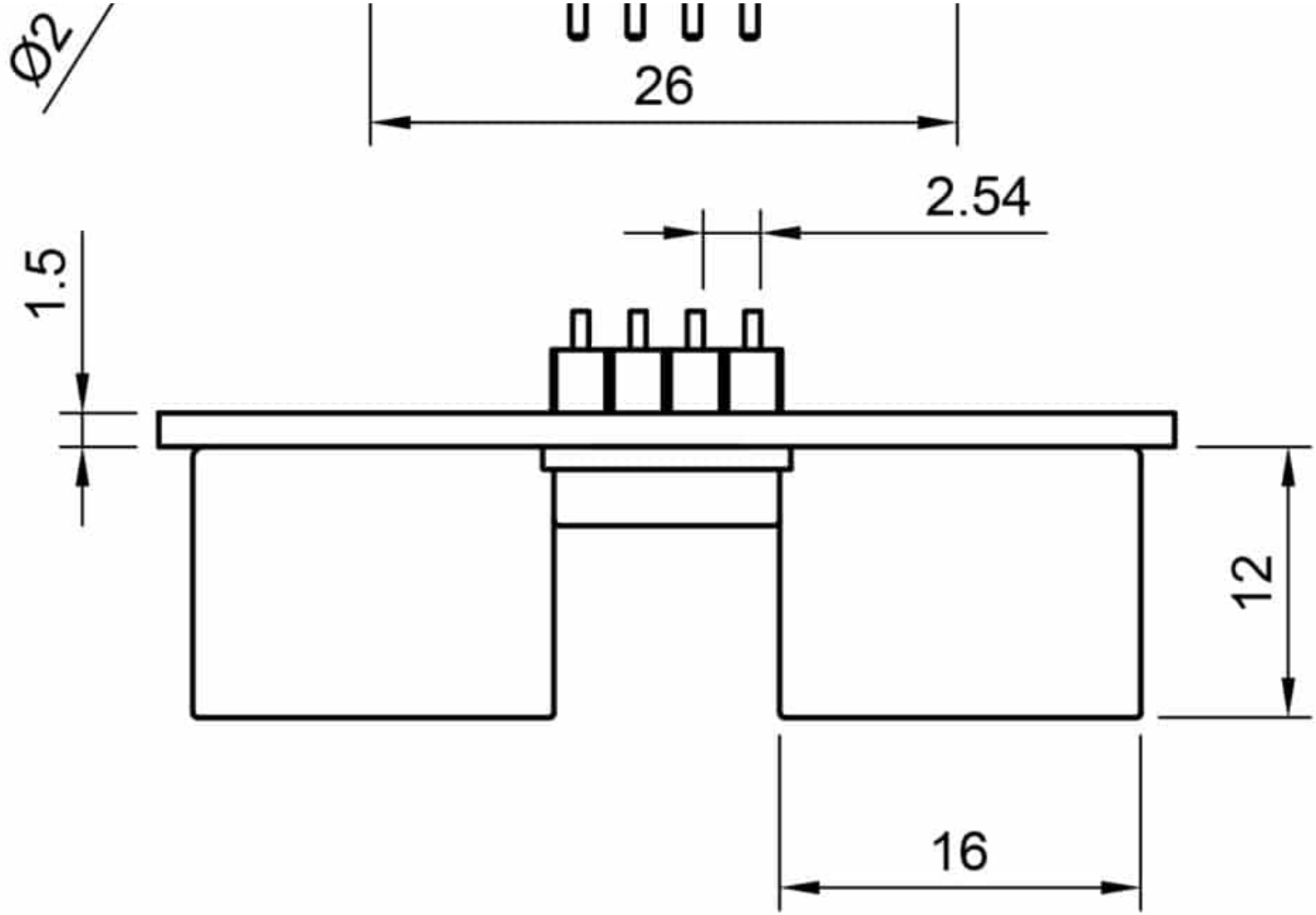
```
36. void setup() {
37.   // Define inputs and outputs:
38.   pinMode(trigPin, OUTPUT);
39.   pinMode(echoPin, INPUT);
40.
41.   dht.begin();
42.
43.   // Initiate the LCD:
44.   lcd.init();
45.   lcd.backlight();
46.
47.   // Begin Serial communication at a baudrate of 9600:
48.   Serial.begin(9600);
49. }
50.
51. void loop() {
52.   // Clear the trigPin by setting it LOW:
53.   digitalWrite(trigPin, LOW);
54.   delayMicroseconds(5);
55.
56.   // Trigger the sensor by setting the trigPin high for 10 microseconds:
57.   digitalWrite(trigPin, HIGH);
58.   delayMicroseconds(10);
59.   digitalWrite(trigPin, LOW);
60.
61.   // Read the echoPin. This returns the duration (length of the pulse) in microseconds:
62.   duration = pulseIn(echoPin, HIGH);
63.
64.   // Read the temperature:
65.   int temperature = dht.readTemperature();
```

```
66.
67.    // Calculate speed of sound in m/s:
68.    speedofsound = 331.3+(0.606*temperature);
69.
70.    // Calculate the distance in cm:
71.    distance = duration*(speedofsound/10000)/2;
72.
73.    // Print the distance and temperature on the Serial Monitor:
74.    lcd.setCursor(0,0);
75.    lcd.print("Temperature: ");
76.    lcd.print(temperature);
77.    lcd.print(" " "\xDF" "C");
78.    lcd.setCursor(0,1);
79.    lcd.print("Speed: ");
80.    lcd.print(speedofsound);
81.    lcd.print(" m/s ");
82.    lcd.setCursor(0,2);
83.    lcd.print("Distance: ");
84.    lcd.print(distance);
85.    lcd.print(" cm  ");
86.    delay(100);
87.  }
```

HC-SR04 Dimensions

Below you can find the dimensions of the HC-SR04 ultrasonic sensor. I have noticed that there are some small differences between manufacturers, so I recommend double checking against your own sensor.





HC-SR04 Dimensions

HC-SR04 CAD

I have created basic CAD drawings of the HC-SR04 ultrasonic sensor that you can download below.

HC-SR04 CAD.zip **(https://www.makerguides.com/wp-content/uploads/2019/02/HC-SR04-CAD.zip)**

Conclusion

In this article I have shown you how the HC-SR04 ultrasonic distance sensor works and how you can use it with Arduino. I hope you found it useful and informative. If you did, **please share it with a friend** that also likes electronics!

Personal project: A couple months ago I built an interactive wall installation with some friends. We used around 30 ultrasonic distance sensors to detect people walking in front of the wall. The wall included lights and sound effects that changed depending on how far away people were standing.

Photo: Guus Schoonewille

I would love to know what projects you plan on building (or have already built) with the HC-SR04 distance sensor. If you have any questions, suggestions or if you think that things are missing in this tutorial, **please leave a comment down below.**

Note that comments are held for moderation in order to prevent spam.

Beginner

Comments

Sanjeev says:
September 27, 2019 at 8:06 am (https://www.makerguides.com/hc-sr04-arduino-tutorial/#comment-662)

Hi,

I am building smart parking system for my office. I'd used the NodeMCU with Ultrasonic (HC-SR04). Now it's returing "0" value and tested with other senor that is returning "1067". I'd used the standard code (written above).
2 Queries
1- which sensor would be best (Ultrasonic HC-SR04 or MaxBotix).

2- Why it's returing incorrect values, earlier it's working fine.
Note- I've connected Ultrasonic (HC-SR04) off VCC pin with NodeMcu +3V. could it be a reason to return wrong value.
Could you please reply the above queries.

Reply

Benne de Bakker says:
September 27, 2019 at 8:26 am (https://www.makerguides.com/hc-sr04-arduino-tutorial/#comment-663)

Hi Sanjeev,

The HC-SR04 cannot be used at 3.3 V without modifying the output (I believe it outputs 5 V even when powered with 3.3 V, so you need a voltage divider). There are some tutorials online that explain how you can modify the sensor, but I would look at a MaxBotix sensor instead. I have written a tutorial for the MB1240, which works great at 3.3 V (https://amzn.to/2m3Nfd0). MaxBotix also sells some cheaper sensors like the LV-MaxSonar-EZ1 (https://amzn.to/2nbKh64).

Tutorial: makerguides.com/maxbotix-mb1240-arduino-tutorial (http://makerguides.com/maxbotix-mb1240-arduino-tutorial)

Benne

Reply

Eka says:
September 8, 2019 at 9:42 am (https://www.makerguides.com/hc-sr04-arduino-tutorial/#comment-512)

Can you help me please, how to change the read out from i2c to 8×32 led max72xxpanel.

Reply

Benne de Bakker says:
September 8, 2019 at 7:30 pm (https://www.makerguides.com/hc-sr04-arduino-tutorial/#comment-522)

I currently do not have a tutorial for this display, but I will write an article for the 8×32 MAX7219 panel soon!

Reply

David Svarrer says:

September 8, 2019 at 8:16 am (https://www.makerguides.com/hc-sr04-arduino-tutorial/#comment-509)

Hy dearest dearest authors of this good article,

One thing – on the echo – the waiting for an echo is, if I am not wrong – waiting for the pin to become HIGH, it is not measuring the duration of a HIGH.

This, if I got it right, is due to that it waits until the signal has come back, then it pulls the pin HIGH, which is what the echo pin detects.

Please correct me if I am wrong on that note.

Thanks for spending countless hours on writing this tutorial. It is really useful for all of us.

Reply

Benne de Bakker says:
September 8, 2019 at 9:38 am (https://www.makerguides.com/hc-sr04-arduino-tutorial/#comment-511)

Hi David,

Thank you for your comment! As far as I know, the echo Pin turns HIGH as soon as you send out the signal. It then stays HIGH until the signal comes back and is received by the sensor (then it turns LOW). With the pulseIn function, you measure the duration of the input pulse from the echo pin. The Arduino waits for the input to go from LOW to HIGH and then times how long it takes to go from HIGH to LOW again.
https://www.arduino.cc/reference/en/language/functions/advanced-io/pulsein/ (https://www.arduino.cc/reference/en/language/functions/advanced-io/pulsein/)

It would be cool to connect the sensor to an oscilloscope, to see the exact timing.

Benne

Reply

Trackbacks

MaxBotix MB7389 Weather Resistant Ultrasonic Sensor Tutorial (https://www.makerguides.com/maxbotix-mb7389-arduino-tutorial/) says:
October 6, 2019 at 11:25 am (https://www.makerguides.com/hc-sr04-arduino-tutorial/#comment-763)
[...] How to use a HC-SR04 Ultrasonic Distance Sensor [...]

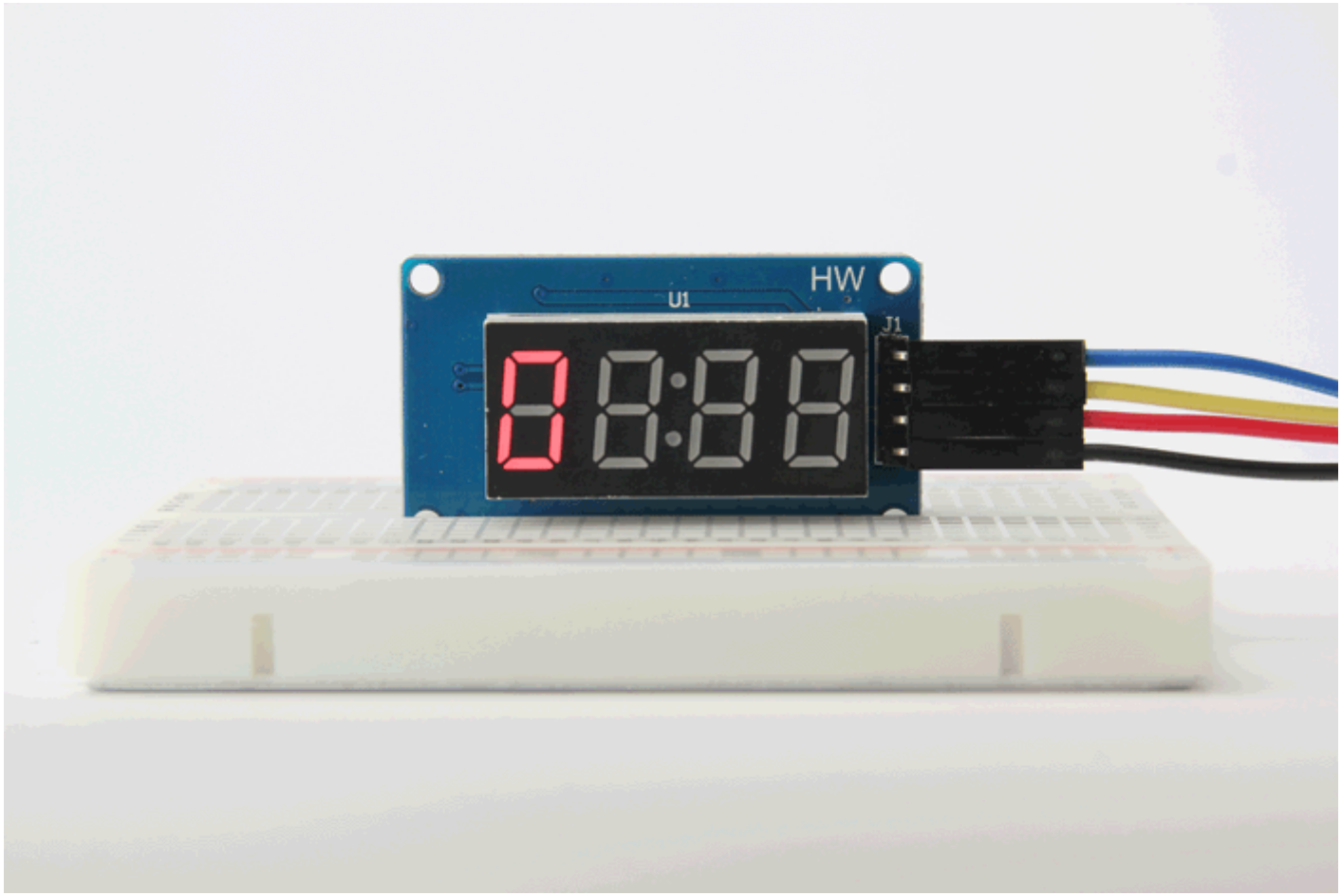
MaxBotix MB1240 Distance Sensor Arduino Tutorial (3 Examples) (https://www.makerguides.com/mb1240-arduino-tutorial/) says:
September 14, 2019 at 9:34 am (https://www.makerguides.com/hc-sr04-arduino-tutorial/#comment-555)
[...] Both the HC-SR04 and the MB1240 do not compensate for a change in air temperature during operation. The XL-MaxSonar and LV-MaxSonar sensors assume an air temperature of 22.5 degrees Celsius. The HR line of sensors features internal temperature calibration, so you don't need to add any sensors yourself. If you would like to see an example that includes a temperature sensor to calibrate the speed of sound in real-time, take a look at this article. [...]

16x2 Character LCD Arduino Tutorial (Wiring Diagram + Examples) (https://www.makerguides.com/character-lcd-arduino-tutorial/) says:
July 8, 2019 at 1:43 pm (https://www.makerguides.com/hc-sr04-arduino-tutorial/#comment-209)
[...] How to use a HC-SR04 Ultrasonic Distance Sensor with Arduino [...]

How to use SHARP IR Distance Sensor with Arduino (GP2Y0A21YK0F) (https://www.makerguides.com/sharp-gp2y0a21yk0f-ir-distance-sensor-arduino-tutorial/) says:
June 24, 2019 at 1:53 pm (https://www.makerguides.com/hc-sr04-arduino-tutorial/#comment-141)
[...] How to use a HC-SR04 Ultrasonic Distance Sensor with Arduino [...]

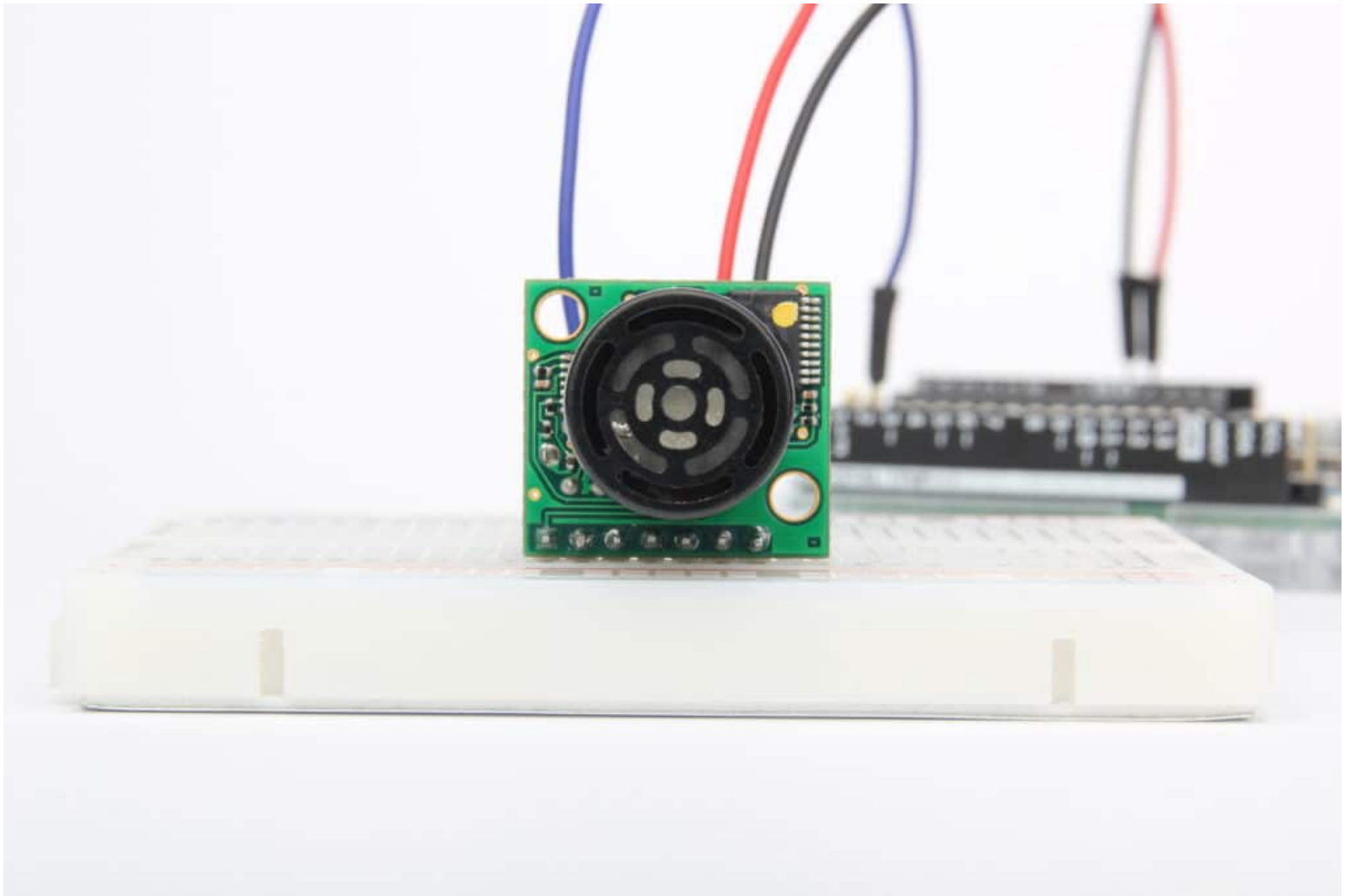
How to use SHARP IR Distance Sensor with Arduino (GP2Y0A710K0F) (https://www.makerguides.com/sharp-gp2y0a710k0f-ir-distance-sensor-arduino-tutorial/) says:
June 20, 2019 at 5:26 pm (https://www.makerguides.com/hc-sr04-arduino-tutorial/#comment-123)
[...] How to use a HC-SR04 Ultrasonic Distance Sensor with Arduino [...]

Waterproof JSN-SR04T Ultrasonic Sensor with Arduino (2 Examples) (https://www.makerguides.com/jsn-sr04t-arduino-tutorial/) says:
June 15, 2019 at 6:10 pm (https://www.makerguides.com/hc-sr04-arduino-tutorial/#comment-107)
[...] more info on how ultrasonic sensors work, you can check out my article on the HC-SR04. In this article the working principles of an ultrasonic distance sensor are explained in much [...]



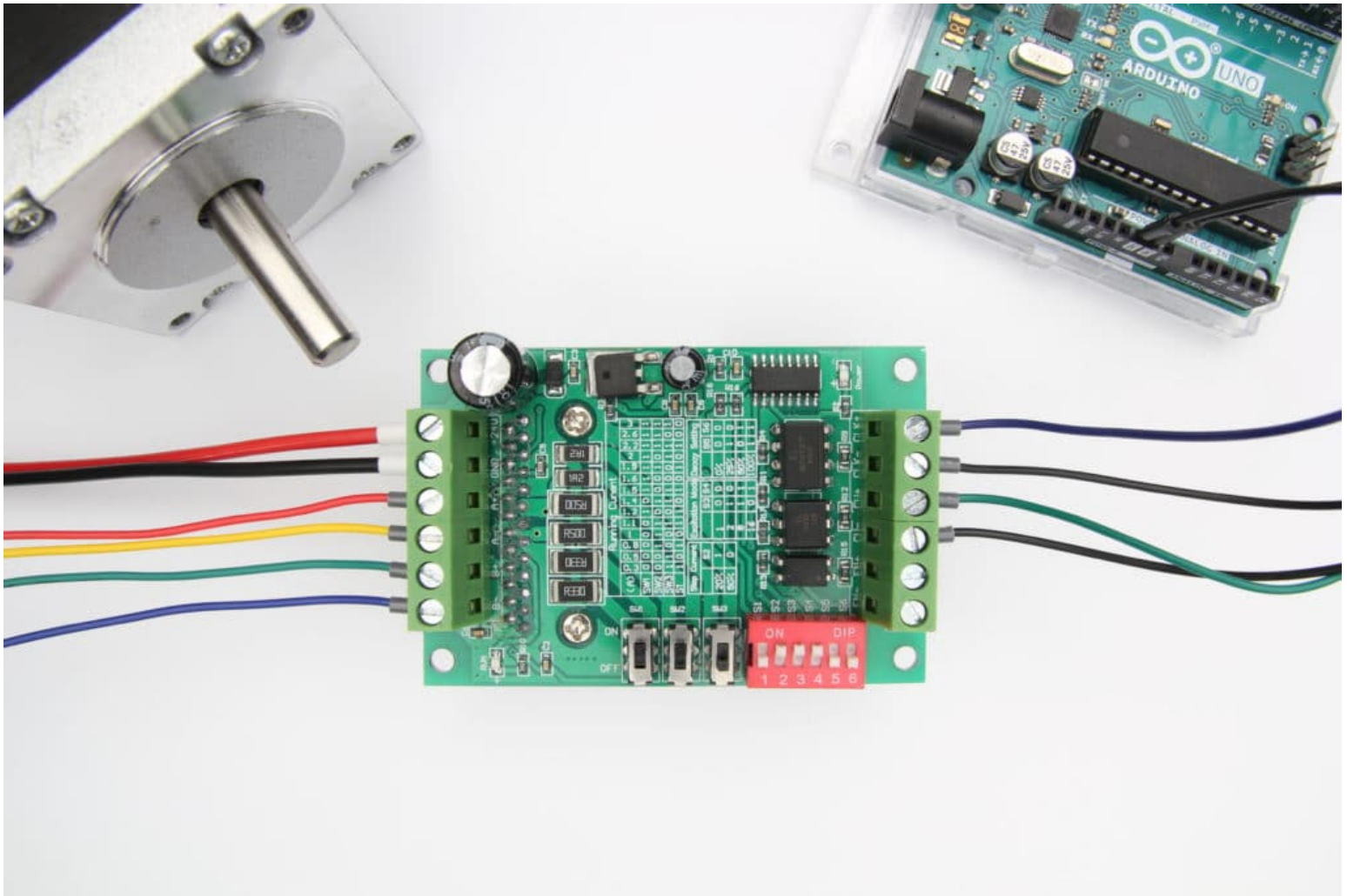
(https://www.makerguides.com/tm1637-arduino-tutorial/)

TM1637 4 Digit 7 Segment Display Arduino Tutorial (https://www.makerguides.com/tm1637-arduino-tutorial/)



(https://www.makerguides.com/maxbotix-mb1240-arduino-tutorial/)

MaxBotix MB1240 ultrasonic distance sensor Arduino tutorial (https://www.makerguides.com/maxbotix-mb1240-arduino-tutorial/)



(https://www.makerguides.com/tb6560-stepper-motor-driver-arduino-tutorial/)

TB6560 Stepper Motor Driver with Arduino Tutorial (https://www.makerguides.com/tb6560-stepper-motor-driver-arduino-tutorial/)