Complete Guide for Ultrasonic Sensor HC-SR04 with Arduino

Description

The HC-SR04 ultrasonic sensor uses sonar to determine the distance to an object. This sensor reads from 2cm to 400cm (0.8inch to 157inch) with an accuracy of 0.3cm (0.1inches), which is good for most hobbyist projects. In addition, this particular module comes with ultrasonic transmitter and receiver modules.

The following picture shows the HC-SR04 ultrasonic sensor.
Features

Here’s a list of some of the HC-SR04 ultrasonic sensor features and specs—for more information, you should consult the sensor’s datasheet:

- Power Supply: +5V DC
- Quiescent Current: <2mA
- Working Current: 15mA
- Effectual Angle: <15°
- Ranging Distance: 2cm – 400 cm/1” – 13ft
- Resolution: 0.3 cm
- Measuring Angle: 30 degree
- Trigger Input Pulse width: 10uS TTL pulse
- Echo Output Signal: TTL pulse proportional to the distance range
- Dimension: 45mm x 20mm x 15mm

How Does it Work?

The ultrasonic sensor uses sonar to determine the distance to an object. Here’s what happens:

1. The ultrasound transmitter (trig pin) emits a high-frequency sound (40 kHz).
2. The sound travels through the air. If it finds an object, it bounces back to the module.
3. The ultrasound receiver (echo pin) receives the reflected sound (echo).
The time between the transmission and reception of the signal allows us to calculate the distance to an object. This is possible because we know the sound’s velocity in the air. Here’s the formula:

\[
\text{distance to an object} = \frac{(\text{speed of sound in the air} \times \text{time})}{2}
\]

- speed of sound in the air at 20ºC (68ºF) = \textbf{343 m/s}

**HC-SR04 Ultrasonic Sensor Pinout**

Here’s the pinout of the HC-SR04 Ultrasonic Sensor.

<table>
<thead>
<tr>
<th>VCC</th>
<th>Powers the sensor (5V)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Trig</td>
<td>Trigger Input Pin</td>
</tr>
<tr>
<td>Echo</td>
<td>Echo Output Pin</td>
</tr>
<tr>
<td>GND</td>
<td>Common GND</td>
</tr>
</tbody>
</table>
Arduino with HC-SR04 Sensor

This sensor is very popular among Arduino tinkerers. So, here we provide an example of how to use the HC-SR04 ultrasonic sensor with the Arduino. In this project, the ultrasonic sensor reads and writes the distance to an object in the serial monitor.

The goal of this project is to help you understand how this sensor works. Then, you should be able to use this example in your own projects.

Arduino with HC-SR04 Sensor Wiring

Follow the next schematic diagram to wire the HC-SR04 ultrasonic sensor to the Arduino.
The following table shows the connections you need to make:

<table>
<thead>
<tr>
<th>Ultrasonic Sensor HC-SR04</th>
<th>Arduino</th>
</tr>
</thead>
<tbody>
<tr>
<td>VCC</td>
<td>5V</td>
</tr>
<tr>
<td>Trig</td>
<td>Pin 11</td>
</tr>
<tr>
<td>Echo</td>
<td>Pin 12</td>
</tr>
<tr>
<td>GND</td>
<td>GND</td>
</tr>
</tbody>
</table>

**Code**

Upload the following code to your Arduino IDE.

```c
/*
* created by Rui Santos, https://randomnerdtutorials.com
*
* Complete Guide for Ultrasonic Sensor HC-SR04
*
* Ultrasonic sensor Pins:
*  VCC: +5VDC
*  Trig: Trigger (INPUT) - Pin11
*  Echo: Echo (OUTPUT) - Pin 12
*  GND: GND
*/

int trigPin = 11;  // Trigger
int echoPin = 12;  // Echo
long duration, cm, inches;

void setup() {
  //Serial Port begin
  Serial.begin (9600);
  //Define inputs and outputs
  pinMode(trigPin, OUTPUT);
  pinMode(echoPin, INPUT);
}

void loop() {
  // The sensor is triggered by a HIGH pulse of 10 or more microseconds.
```
How the Code Works

First, you create variables for the trigger and echo pin called `trigPin` and `echoPin`, respectively. The trigger pin is connected to digital Pin 11, and the echo pin is connected to Pin 12:

```cpp
int trigPin = 11;
int echoPin = 12;
```

You also create three variables of type long: `duration` and `inches`. The `duration` variable saves the time between the emission and reception of the signal. The `cm` variable will save the distance in centimeters, and the `inches` variable will save the distance in inches.

```cpp
long duration, cm, inches;
```

In the `setup()`, initialize the serial port at a baud rate of 9600, and set the trigger pin as an OUTPUT and the echo pin as an INPUT.

```cpp
//Serial Port begin
Serial.begin(9600);
//Define inputs and outputs
```
```cpp
pinMode(trigPin, OUTPUT);
pinMode(echoPin, INPUT);

In the `loop()`, trigger the sensor by sending a **HIGH** pulse of 10 microseconds. But, before that, give a short **LOW** pulse to ensure you’ll get a clean **HIGH** pulse:
```
digitalWrite(trigPin, LOW);
delayMicroseconds(5);
digitalWrite(trigPin, HIGH);
delayMicroseconds(10);
digitalWrite(trigPin, LOW);
```

We use the `pulseIn()` function to get the sound wave travel time:
```
duration = pulseIn(echoPin, HIGH);
```

The `pulseIn()` function reads a **HIGH** or a **LOW** pulse on a pin. It accepts as arguments the pin and the state of the pulse (either **HIGH** or **LOW**). It returns the length of the pulse in microseconds. The pulse length corresponds to the time it took to travel to the object plus the time traveled on the way back.

Then, we calculate the distance to an object, taking into account the sound speed.

```
distance = (traveltime/2) x speed of sound
```

The speed of sound is: 343m/s = 0.0343 cm/μs = 1/29.1 cm/μs

Or in inches: 13503.9in/s = 0.0135in/μs = 1/74in/μs

We need to divide the travel time by 2 because we have to consider that the wave was sent, hit the object, and then returned to the sensor.

```
cm = (duration/2) / 29.1;
```
```
inches = (duration/2) / 74;
```

Finally, we print the results in the Serial Monitor:
```
Serial.println();
```
```
Serial.println(inches);
Serial.println("in, ");
Serial.println(cm);
Serial.println("cm");
```

---

**Source code with NewPing Library**

You can also use the NewPing library. Download the library [here](https://randomnerdtutorials.com).

After installing the NewPing library, you can upload the code provided below.

```cpp
/*
 * Posted on https://randomnerdtutorials.com
```
```cpp
#include <NewPing.h>

#define TRIGGER_PIN 11
#define ECHO_PIN 12
#define MAX_DISTANCE 200

// NewPing setup of pins and maximum distance
NewPing sonar(TRIGGER_PIN, ECHO_PIN, MAX_DISTANCE);

void setup() {
  Serial.begin(9600);
}

void loop() {
  delay(50);
  unsigned int distance = sonar.ping_cm();
  Serial.print(distance);
  Serial.println("cm");
}
```

**How the Code Works**

Getting the distance to an object using the NewPing library is much simpler.

You start by including the NewPing library:

```cpp
#include <NewPing.h>
```

Then, define the trigger and echo pin. The trigger pin is connected to the Arduino digital Pin 11 and the echo to Pin 12. You also need to define the MAX_DISTANCE variable to be able to use the library.

```cpp
#define TRIGGER_PIN 11
#define ECHO_PIN 12
#define MAX_DISTANCE 200
```

Then, you create a `NewPing` instance called `sonar`:

```cpp
NewPing sonar(TRIGGER_PIN, ECHO_PIN, MAX_DISTANCE);
```

In the `setup()`, you initialize the Serial communication at a baud rate of 9600.

```cpp
Serial.begin(9600);
```
Finally, in the `loop()`, you just need to use the `ping_cm()` method on the `sonar` object to get the distance in centimeters.

```c
unsigned int distance = sonar.ping_cm();
```

If you want to get the distance in inches, you can use `sonar.ping_in()` instead.

**Demonstration**

Upload the code to your Arduino board. Then, open the Serial Monitor at a baud rate of 115200.

The distance to the nearest object is printed in the Serial Monitor window.