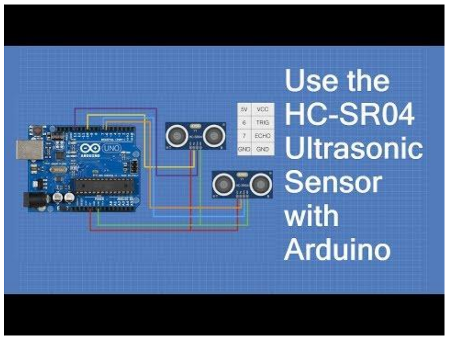
Using the HC-SR04 Ultrasonic Distance Sensor with Arduino

Table of Contents [hide]

One of the most useful sensors for robotics projects is a distance sensor. The HC-SR04 is an inexpensive Ultrasonic Distance Sensor that can assist your robot in navigating around a room. With a bit of care and an additional component it can also be used as a measurement device. In this article you'll learn everything you need to know to use this wonderful little device with an Arduino.



The HC-SR04 Ultrasonic Distance Sensor

The HC-SR04 Ultrasonic Distance Sensor is an inexpensive device that is very useful for robotics and test equipment projects. This tiny sensor is capable of measuring the distance between itself and the nearest solid object, which is really good information to have if you're trying to avoid driving into a wall!

The HC-SR04 can be hooked directly to an Arduino or other microcontroller and it operates on 5 volts. It can also be used with the Raspberry Pi, however since the HC-SR04 requires 5-volt logic you'll need a couple of resistors to interface it with the Pi's 3.3 volt GPIO port.

This ultrasonic distance sensor is capable of measuring distances between 2 cm to 400 cm (that’s about an inch to 13 feet for those of you who don’t “speak” Metric). It’s a low current device so it’s suitable for battery powered devices. And as a bonus it even looks cool, like a pair of Wall-E Robot eyes for your latest robotic invention!
So read on and I’ll show you how to hook up and use the HC-SR04 Ultrasonic Distance Sensor. We’ll also put it through some tests to see how accurate it is and we’ll look at how we can possibly improve upon that accuracy. And of course I’ll have some sample code and projects for you to try out.
Let’s get started!

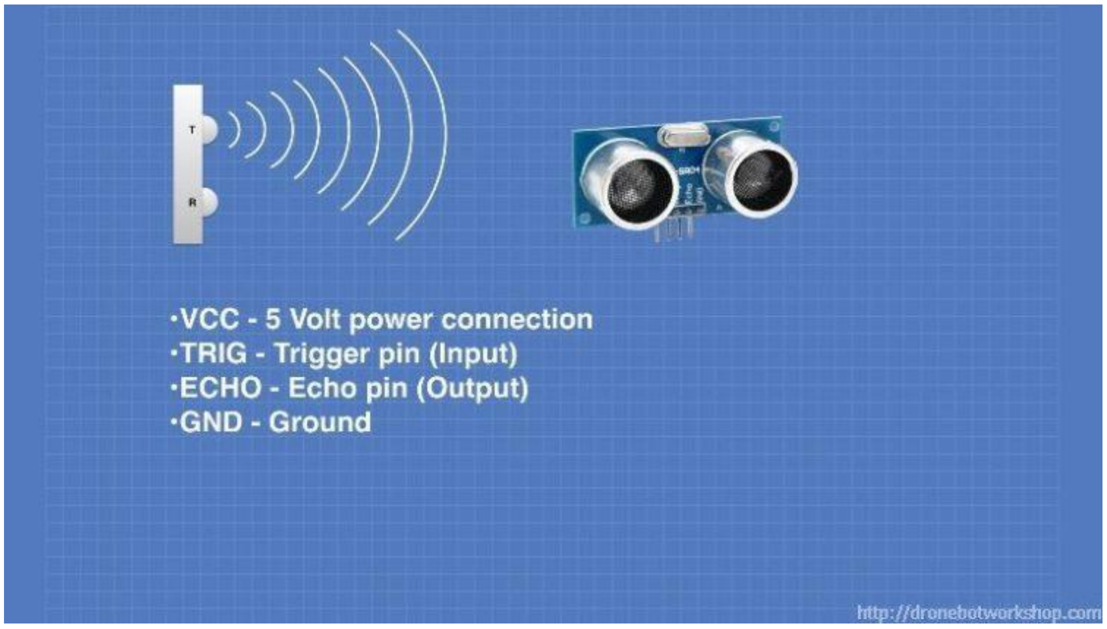
How the HC-SR04 Works

Ultrasonic distance sensors use pulses of ultrasonic sound (sound above the range of human hearing) to detect the distance between them and nearby solid objects. The sensors consist of two main components:

- **An Ultrasonic Transmitter** – This transmits the ultrasonic sound pulses, it operates at 40 KHz
- **An Ultrasonic Receiver** – The receiver listens for the transmitted pulses. If it receives them it produces an output pulse whose width can be used to determine the distance the pulse travelled.

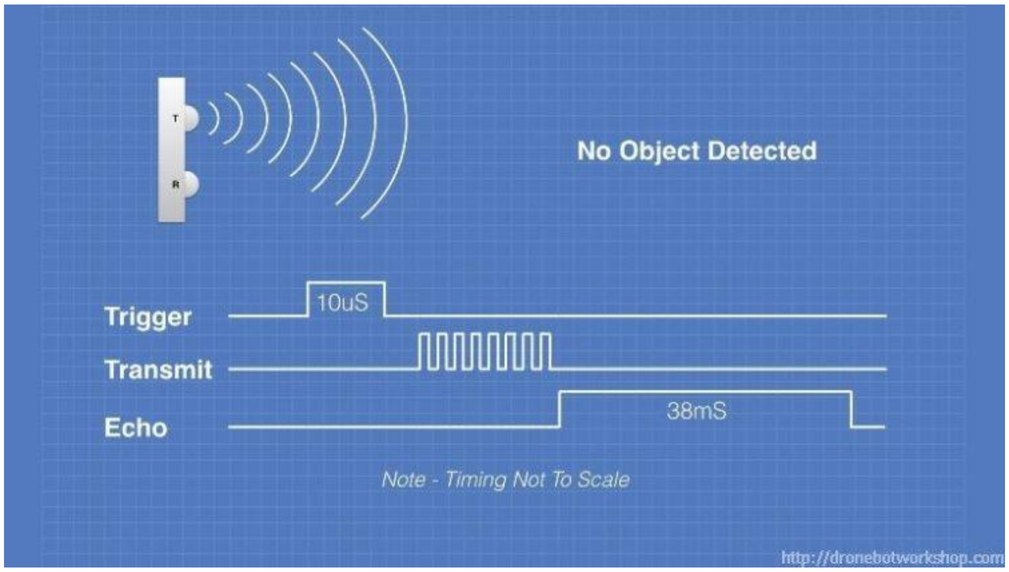
The HC-SR04 has the following four connections:

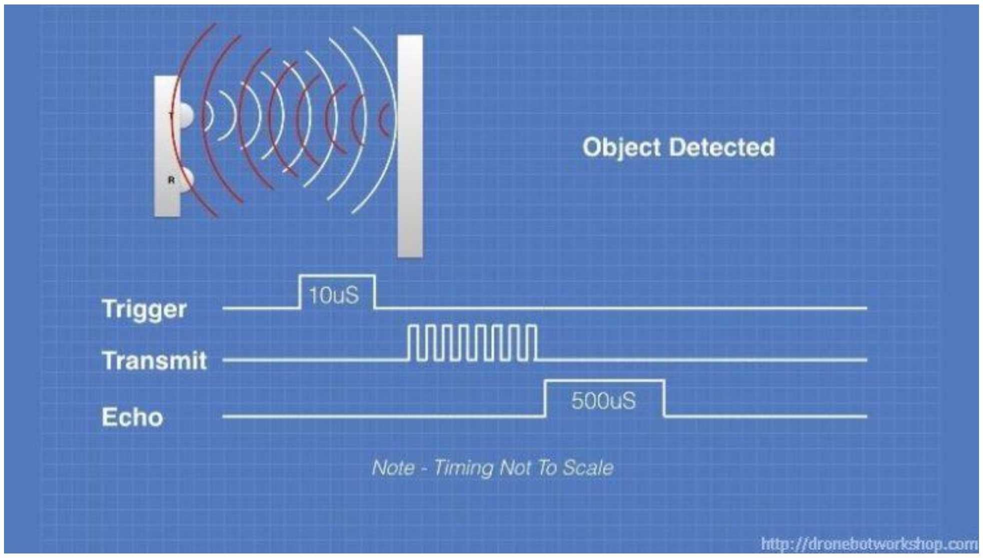
- **VCC** – This is the 5 Volt positive power supply.
- **Trig** – This is the “Trigger” pin, the one driven to send the ultrasonic pulses.
- **Echo** – This is the pin that produces a pulse when the reflected signal is received. The length of the pulse is proportional to the time it took for the transmitted signal to be detected.
- **GND** – This is the Ground pin.


T
R
•VCC - 5 Volt power connection
•TRIG - Trigger pin (Input)
•ECHO - Echo pin (Output)
•GND - Ground
http://dronebotworkshop.com

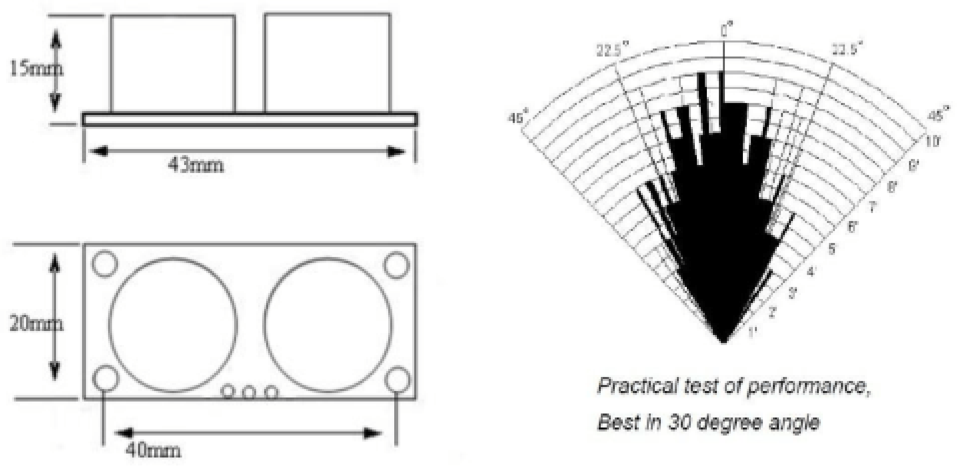
The device operates as follows:

1. A 5 volt pulse of at least 10 uS (10 microseconds) in duration is applied to the Trigger pin.
2. The HC-SR04 responds by transmitting a burst of eight pulses at 40 KHz. This 8-pulse pattern makes the “ultrasonic signature” from the device unique, allowing the receiver to discriminate between the transmitted pattern and the ultrasonic background noise.
3. The eight ultrasonic pulses travel through the air away from the transmitter. Meanwhile the Echo pin goes high to start forming the beginning of the echo-back signal.
4. If the pulse in NOT reflected back then the Echo signal will timeout after 38 mS (38 milliseconds) and return low. This produces a 38 mS pulse that indicates no obstruction within the range of the sensor.
5. If the pulse IS reflected back the Echo pin goes low when the signal is received. This produces a pulse whose width varies between 150 uS to 25 mS, depending upon the time it took for the signal to be received.
6. The width of the received pulse is used to calculate the distance to the reflected object. Remember that the pulse indicates the time it took for the signal to be sent out and reflected back so to get the distance you’ll need to divide your result in half.
7.



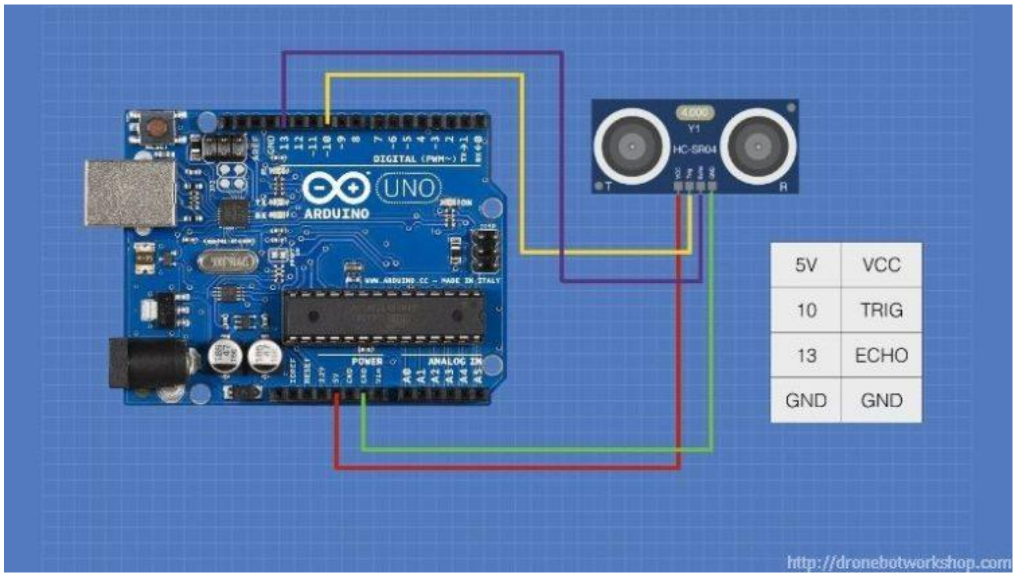


The illustration below shows the dimensions of the HC-SR04 Ultrasonic Distance Sensor as well as the effective angle of operation. As you can see the sensor is most accurate when the object to be detected is directly in front of it but you do get a response from objects within a 45 degree "window". The documentation recommends confining that window to 30 degrees (15 degrees on either side) for accurate readings.



Hooking Up the HC-SR04

Connecting the HC-SR04 to the Arduino is pretty easy. You'll need a couple of digital I/O ports and a connection to the Arduino's 5-Volt and Ground pins.



Actually if you're short of pins you can even connect both the Trigger and Echo pins of the HC-SR04 to just one digital I/O pin on the Arduino and use code to switch the pin between output (to send the 10 uS pulse) and input (to receive the Echo pulse). Some ultrasonic sensors actually have only one pin that does both Trigger and Echo. I'll discuss this and give an example further down, so keep reading.

Most of the examples I'll be showing you here use the more conventional two-pin method. Any Arduino and any digital I/O pins that are free can be used so if you wish to hook it up to a different set of I/O pins then simply change the sketches to reflect those changes. For our demo I'll use an Arduino Uno and pin 10 for the Trigger and pin 13 for the Echo.

The application notes for the HC-SR04 stress that you need to have the Ground pin connected before you hook up VCC (5-Volts), so if you're experimenting "live" on a solderless breadboard you might want to keep that in mind.

So now that we've hooked up our ultrasonic distance sensor it's time to write some code and test it out.

Basic Arduino Demonstration

In our first demonstration we will simply test the sensor to see if it is working. The sketch is pretty simple and it uses the Serial Monitor to display the distance it detects, measured in centimeters. Let's look it over in detail.

In order to test out the accuracy of the ultrasonic distance sensor I setup a test board with a sensor mounted on one end (I used Velcro to mount it). I put a 1 metre stick on the board so I could test the sensor in the 2 -100 cm range. You can watch the video associated with this article if you want to see the results for this and the other demos.

If you want to display your results in inches instead of centimeters there are two ways you can do this:

1. Use the Imperial value for the speed of sound instead of the Metric one. At sea level at 20 degrees Celsius (68 degrees Fahrenheit) sound travels at 343 metres per second, which is 1,125 feet per second or 13500 inches per second.
2. Keep the code as it is but convert to inches at the end. There are 2.54 centimetres in an inch. Personally this is how I would do this, as it would allow me to display the results in both Imperial and Metric values.

Otherwise just use the Metric system, it's used all over the world and (more importantly) it was used on Star Trek The Next Generation. If it's good enough for Captain Picard then it's good enough for me!

So on to the sketch. This is the basic HC-SR04 sketch, for a detailed breakdown on how it works just look at the video or simply read the comments in the code.

```
/*
  HC-SR04 Basic Demonstration
  HC-SR04-Basic-Demo.ino
  Demonstrates functions of HC-SR04 Ultrasonic Range Finder
  Displays results on Serial Monitor

  DroneBot Workshop 2017
  http://dronebotworkshop.com
*/

// This uses Serial Monitor to display Range Finder distance readings

// Hook up HC-SR04 with Trig to Arduino Pin 10, Echo to Arduino pin 13

#define trigPin 10
#define echoPin 13

float duration, distance;

void setup() {
  Serial.begin (9600);
  pinMode(trigPin, OUTPUT);
  pinMode(echoPin, INPUT);
}

void loop() {

  // Write a pulse to the HC-SR04 Trigger Pin

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

  // Measure the response from the HC-SR04 Echo Pin

  duration = pulseIn(echoPin, HIGH);

  // Determine distance from duration
  // Use 343 metres per second as speed of sound

  distance = (duration / 2) * 0.0343;

  // Send results to Serial Monitor

  Serial.print("Distance = ");
  if (distance >= 400 || distance <= 2) {
     Serial.println("Out of range");
  }
  else {
    Serial.print(distance);
    Serial.println(" cm");
    delay(500);
  }
  delay(500);
}
```

Arduino Code Libraries

In our first sketch we didn't use any code libraries, we simply used the Arduino's *delayMicrosecond* command to create our 10 uS pulse for triggering and the *pulseIn* command to measure the received signal pulse width. However there are other ways of doing this using special code libraries. There are quite a few of them available, the most versatile is one called "NewPing".

If you haven't had any experience using libraries in your Arduino sketches it's a skill that you really need to learn. Libraries provide code functions for specific tasks, and there are literally hundreds of libraries available for the Arduino for tasks and to support all kinds of external hardware.

The Arduino website has instructions for installing new libraries in your Arduino IDE.

The NewPing library was written by Tim Eckel and it replaces the older Ping library which was written by Caleb Zulawski and designed primarily for the Parallax Ping ultrasonic sensor (although it will work with the HC-SR04 if you use it in 3-pin mode).

The NewPing library is quite advanced and it improves upon the accuracy of our original sketch considerably. It also supports up to 15 ultrasonic sensors at once and it can directly output in centimetres, inches or time duration.

Here is our sketch rewritten to use the NewPing library:

```
/*
  HC-SR04 NewPing Library Demonstration
  HC-SR04-NewPing.ino
  Demonstrates functions of NewPing Library for HC-SR04 Ultrasonic Range Finder
  Displays results on Serial Monitor

  DroneBot Workshop 2017
  http://dronebotworkshop.com
*/

// This uses Serial Monitor to display Range Finder distance readings

// Include NewPing Library
#include "NewPing.h"

// Hook up HC-SR04 with Trig to Arduino Pin 10, Echo to Arduino pin 13
// Maximum Distance is 400 cm

#define TRIGGER_PIN  10
#define ECHO_PIN     13
#define MAX_DISTANCE 400

NewPing sonar(TRIGGER_PIN, ECHO_PIN, MAX_DISTANCE);

float distance;

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

void loop() {

  distance = sonar.ping_cm();

  // Send results to Serial Monitor
  Serial.print("Distance = ");
  if (distance >= 400 || distance <= 2) {
    Serial.println("Out of range");
  }
  else {
    Serial.print(distance);
    Serial.println(" cm");
    delay(500);
  }
  delay(500);
}
```

The above sketch is simple and works well but it only has a resolution down to one centimeter. If you want to bring back the decimal point values you can use NewPing in duration mode instead of in distance mode. We can then use the duration to calculate the distance as we did in the first sketch we looked at.

Here is our sketch NewPing rewritten to use duration instead of distance.

```
/*
  HC-SR04 NewPing Duration Demonstration
  HC-SR04-NewPing-Duration.ino
  Demonstrates using Duration function of NewPing Library for HC-SR04 Ultrasonic Range Finder
  Displays results on Serial Monitor

  DroneBot Workshop 2017
  http://dronebotworkshop.com
*/

// This uses Serial Monitor to display Range Finder distance readings

// Include NewPing Library
#include "NewPing.h"

// Hook up HC-SR04 with Trig to Arduino Pin 10, Echo to Arduino pin 13
// Maximum Distance is 400 cm
```

```

#define TRIGGER_PIN  10
#define ECHO_PIN     13
#define MAX_DISTANCE 400

NewPing sonar(TRIGGER_PIN, ECHO_PIN, MAX_DISTANCE);

float duration, distance;

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

void loop() {

  duration = sonar.ping();

  // Determine distance from duration
  // Use 343 metres per second as speed of sound

  distance = (duration / 2) * 0.0343;

  // Send results to Serial Monitor
  Serial.print("Distance = ");
  if (distance >= 400 || distance <= 2) {
    Serial.println("Out of range");
  }
  else {
    Serial.print(distance);
    Serial.println(" cm");
    delay(500);
  }
  delay(500);
}
```

Another function of NewPing is “iterations” To iterate means to go over something more than once, and that’s precisely what the iteration mode does. It takes many duration measurements instead of just one, throws away any invalid readings and then averages the remaining ones. By default it takes 5 readings but you can actually specify as many as you wish.
Here we go again with a NewPing sketch written to use iterations. As you can see it’s virtually identical to the previous sketch, all that has been added is a variable to specify the number of iterations.

```
/*
  HC-SR04 NewPing Iteration Demonstration
  HC-SR04-NewPing-Iteration.ino
  Demonstrates using Iteration function of NewPing Library for HC-SR04 Ultrasonic Range Finder
  Displays results on Serial Monitor

  DroneBot Workshop 2017
  http://dronebotworkshop.com
*/

// This uses Serial Monitor to display Range Finder distance readings

// Include NewPing Library
#include "NewPing.h"
```

```

// Hook up HC-SR04 with Trig to Arduino Pin 10, Echo to Arduino pin 13
// Maximum Distance is 400 cm

#define TRIGGER_PIN  10
#define ECHO_PIN     13
#define MAX_DISTANCE 400

NewPing sonar(TRIGGER_PIN, ECHO_PIN, MAX_DISTANCE);

float duration, distance;

int iterations = 5;

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

void loop() {

  duration = sonar.ping_median(iterations);

  // Determine distance from duration
  // Use 343 metres per second as speed of sound

  distance = (duration / 2) * 0.0343;

  // Send results to Serial Monitor
  Serial.print("Distance = ");
  if (distance >= 400 || distance <= 2) {
    Serial.println("Out of range");
  }
  else {
    Serial.print(distance);
    Serial.println(" cm");
    delay(500);
  }
  delay(500);
}
```

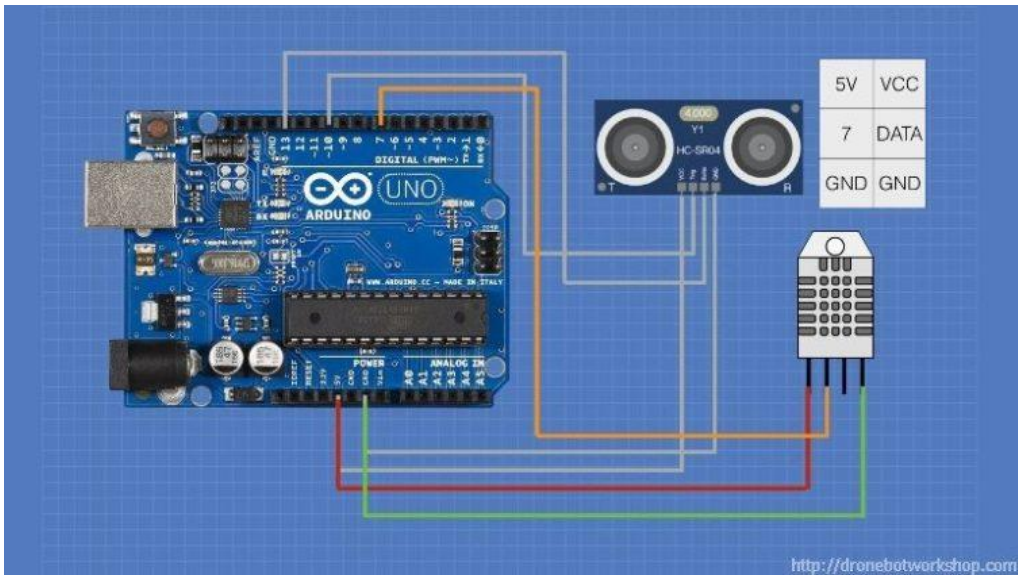
Getting Improved Accuracy

The HC-SR04 is reasonably accurate, in it’s basic form it is quite useable for robots, intruder detection or proximity alarms. But there are times you might want a bit more accuracy, for example you may be building a measuring tool or might be using your robot to map out the perimeter of a room. If so there are a few things you can do to improve upon the accuracy of the HC-SR04.

As I mentioned in the last section the NewPing library has implemented a number of internal techniques to improve the accuracy of the sensor. In most cases this is all you’ll need to improve your readings.

If you’re designing a device that is to be used outdoors or in an unusually hot or cold environment you might want to take into account the fact that the speed of sound in air varies with temperature, air pressure and humidity. Since the speed of sound factors into our HC-SR04 distance calculation this could affect our readings if the temperature was a lot hotter or colder than room temperature.

In order to factor in temperature and humidity I decided to use the DHT22 sensor, it's relatively inexpensive but very accurate. You could also use the less expensive DHT-11 to do this experiment with slightly less accuracy. Here is how I hooked it up:



The DHT22 requires a couple of code libraries to get it to function, Adafruit has two libraries that will work well with both the DHT22 and the DHT11. The Adafruit AM2315 library and the Adafruit Unified Sensor library can both be installed directly within the Arduino IDE using the Library Manager.

After I installed the Adafruit libraries I wrote a quick test sketch to be sure my DHT22 was working correctly.

```
/*
  DHT22 Basic Demonstration
  DHT22-Basic-Demo.ino
  Demonstrates functions of DHT22 Digital Temperature & Humidity Sensor
  Displays results on Serial Monitor

  DroneBot Workshop 2017
  http://dronebotworkshop.com
*/

// Include DHT Libraries from Adafruit
// Dependant upon Adafruit_Sensors Library

#include "DHT.h";

// Define Constants

#define DHTPIN 7     // DHT-22 Output Pin connection
```

```
#define DHTTYPE DHT22   // DHT Type is DHT 22 (AM2302)

// Initialize DHT sensor for normal 16mhz Arduino

DHT dht(DHTPIN, DHTTYPE);


// Define Variables

float hum;  //Stores humidity value
float temp; //Stores temperature value

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

void loop()
{
    delay(2000);  // Delay so sensor can stabalize

    hum = dht.readHumidity();  // Get Humidity value
    temp= dht.readTemperature();  // Get Temperature value

    // Print temperature and humidity values to serial monitor

    Serial.print("Humidity: ");
    Serial.print(hum);
    Serial.print(" %, Temp: ");
    Serial.print(temp);
    Serial.println(" Celsius");

}
```

And finally here is a sketch that factors in both temperature and humidity to improve accuracy using the DHT22.

```
/*
  HC-SR04 with Temp and Humidity Demonstration
  HC-SR04-Temp-Humid-Demo.ino
  Demonstrates enhancements of HC-SR04 Ultrasonic Range Finder
  With DHT22 Temperature and Humidity Sensor
  Displays results on Serial Monitor

  DroneBot Workshop 2017
  http://dronebotworkshop.com
*/

// Include DHT Libraries from Adafruit
// Dependant upon Adafruit_Sensors Library
#include "DHT.h";

// Include NewPing Library
#include "NewPing.h"

// Define Constants

#define DHTPIN 7       // DHT-22 Output Pin connection
#define DHTTYPE DHT22   // DHT Type is DHT 22 (AM2302)
#define TRIGGER_PIN  10
```

```
#define ECHO_PIN     13
#define MAX_DISTANCE 400

NewPing sonar(TRIGGER_PIN, ECHO_PIN, MAX_DISTANCE);

// Define Variables

float hum;    // Stores humidity value in percent
float temp;   // Stores temperature value in Celcius
float duration; // Stores HC-SR04 pulse duration value
float distance; // Stores calculated distance in cm
float soundsp;  // Stores calculated speed of sound in M/S
float soundcm;  // Stores calculated speed of sound in cm/ms
int iterations = 5;

// Initialize DHT sensor for normal 16mhz Arduino

DHT dht(DHTPIN, DHTTYPE);

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

void loop()
{

  delay(2000);  // Delay so DHT-22 sensor can stabalize

    hum = dht.readHumidity();  // Get Humidity value
    temp= dht.readTemperature();  // Get Temperature value

    // Calculate the Speed of Sound in M/S
    soundsp = 331.4 + (0.606 * temp) + (0.0124 * hum);

    // Convert to cm/ms

    soundcm = soundsp / 10000;

  duration = sonar.ping_median(iterations);

  // Calculate the distance
  distance = (duration / 2) * soundcm;

  // Send results to Serial Monitor

    Serial.print("Sound: ");
    Serial.print(soundsp);
    Serial.print(" m/s, ");
    Serial.print("Humid: ");
    Serial.print(hum);
    Serial.print(" %, Temp: ");
    Serial.print(temp);
    Serial.print(" C, ");
    Serial.print("Distance: ");

    if (distance >= 400 || distance <= 2) {
    Serial.print("Out of range");
    }
    else {
    Serial.print(distance);
    Serial.print(" cm");
    delay(500);
   }

  Serial.println(" ");
}
```

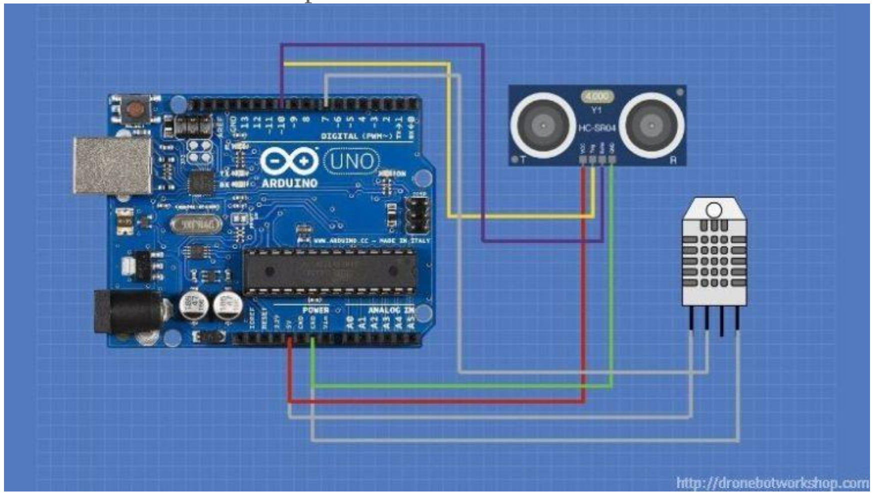
In my testing on the workbench I found that it did indeed improve the accuracy of the readings.

Using 3-Wire Mode

As I alluded to earlier it is possible to use the HC-SR04 in “3-Wire Mode”. In this mode you will only require one connection to a single Arduino digital I/O pin. There are other ultrasonic sensors that only operate in 3-Wire Mode.

In 3-Wire mode the single I/O pin is used as both an input and as an output. This is possible because there is never a time when both the input and output are being used. By eliminating one I/O pin requirement we can save a connection to our Arduino and use it for something else. It also is useful when using a chip like the ATtiny85 which has a limited number of I/O pins.

Here is how I hooked up the HC-SR04 to the Arduino.



As you can see all I did was to connect both the trigger and echo to Arduino pin 10.

And here is the sketch I wrote to use it. Note that the only difference between this sketch and the previous one is that I’ve specified pin 10 for both the Trigger and Echo pin values. The rest of the sketch is identical.

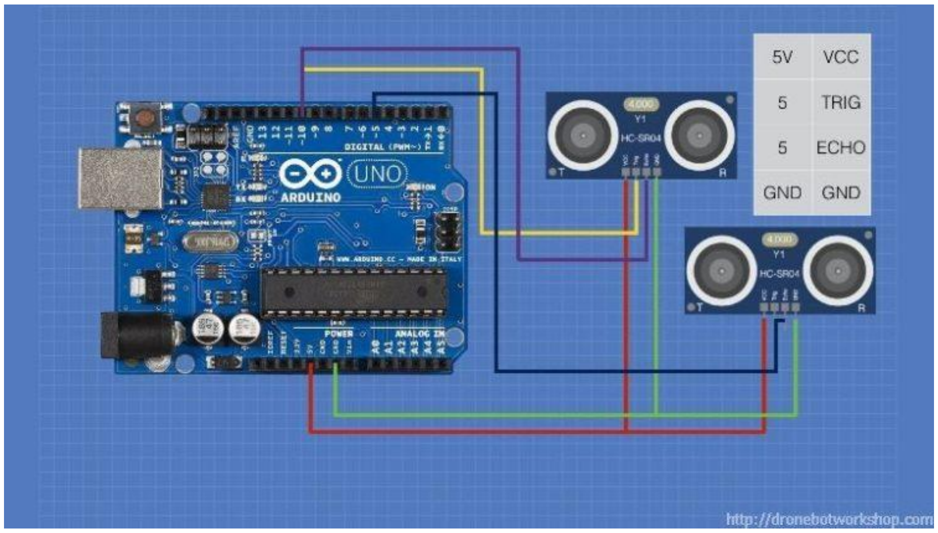
```
/*
  HC-SR04 in 3-Wire Mode with Temp and Humidity Demonstration
  HC-SR04-3Wire-Temp-Humid-Demo.ino
  Demonstrates enhancements of HC-SR04 Ultrasonic Range Finder
  With DHT22 Temperature and Humidity Sensor
  Displays results on Serial Monitor

  DroneBot Workshop 2017
  http://dronebotworkshop.com
*/

// Include DHT Libraries from Adafruit
// Dependant upon Adafruit_Sensors Library
#include "DHT.h";

// Include NewPing Library
#include "NewPing.h"

// Define Constants

#define DHTPIN 7       // DHT-22 Output Pin connection
#define DHTTYPE DHT22   // DHT Type is DHT 22 (AM2302)
#define TRIGGER_PIN  10  // Trigger and Echo both on pin 10
#define ECHO_PIN     10
#define MAX_DISTANCE 400

NewPing sonar(TRIGGER_PIN, ECHO_PIN, MAX_DISTANCE);

// Define Variables

float hum;    // Stores humidity value in percent
float temp;   // Stores temperature value in Celcius
float duration; // Stores HC-SR04 pulse duration value
float distance; // Stores calculated distance in cm
float soundsp;  // Stores calculated speed of sound in M/S
float soundcm;  // Stores calculated speed of sound in cm/ms
int iterations = 5;

// Initialize DHT sensor for normal 16mhz Arduino

DHT dht(DHTPIN, DHTTYPE);

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

void loop()
{

  delay(2000);  // Delay so DHT-22 sensor can stabalize

    hum = dht.readHumidity();  // Get Humidity value
    temp= dht.readTemperature();  // Get Temperature value

    // Calculate the Speed of Sound in M/S
    soundsp = 331.4 + (0.606 * temp) + (0.0124 * hum);

    // Convert to cm/ms

    soundcm = soundsp / 10000;

  duration = sonar.ping_median(iterations);

  // Calculate the distance
  distance = (duration / 2) * soundcm;

  // Send results to Serial Monitor

    Serial.print("Sound: ");
    Serial.print(soundsp);
    Serial.print(" m/s, ");
    Serial.print("Humid: ");
    Serial.print(hum);
    Serial.print(" %, Temp: ");
    Serial.print(temp);
    Serial.print(" C, ");
    Serial.print("Distance: ");

    if (distance >= 400 || distance <= 2) {
    Serial.print("Out of range");
    }
    else {
    Serial.print(distance);
    Serial.print(" cm");
    delay(500);
    }

  Serial.println(" ");
}
```

Using Multiple HC-SR04 Sensors

In many applications you'll want to use more than one HC-SR04 ultrasonic sensor in your design. This can happen when you want to monitor the distance to external objects from different sides of your robot or project. Two of them can be used for front and rear sensors, or hookup 6 of them and monitor each side of a cube – it's up to you!

When using multiple sensors one obvious consideration is that you need to keep the signal emitted by one sensor from being picked up and measured by another sensor. The easiest way to accomplish this is to simply pulse the trigger on one sensor and wait until you receive the echo back before proceeding to the next sensor. It might be wise to leave a small delay between readings just in case the previous sensors sound waves are still bouncing around the room!

Here is how I used two HC-SR04 ultrasonic sensor with the Arduino. Note that I wired these in 3-wire mode, if you wish you can also connect them in the conventional 4-wire fashion. If you do then just modify the sketch to specify different trigger and echo pins.



Here is a sketch that uses the NewPing library and two sensors. As with all of the other sketches it outputs the results to the serial monitor.

```
/*
  Dual HC-SR04 with Temp and Humidity Demonstration
  HC-SR04-Temp-Humid-Dual-Demo.ino
  Demonstrates enhancements of HC-SR04 Ultrasonic Range Finder
  With DHT22 Temperature and Humidity Sensor
  Displays results on Serial Monitor

  DroneBot Workshop 2017
  http://dronebotworkshop.com
*/

// Include DHT Libraries from Adafruit
// Dependant upon Adafruit_Sensors Library
#include "DHT.h";

// Include NewPing Library
#include "NewPing.h"

// Define Constants

#define DHTPIN 7       // DHT-22 Output Pin connection
#define DHTTYPE DHT22   // DHT Type is DHT 22 (AM2302)
#define TRIGGER_PIN_1  10
#define ECHO_PIN_1     10
#define TRIGGER_PIN_2  5
#define ECHO_PIN_2     5
#define MAX_DISTANCE 400

NewPing sonar1(TRIGGER_PIN_1, ECHO_PIN_1, MAX_DISTANCE);
NewPing sonar2(TRIGGER_PIN_2, ECHO_PIN_2, MAX_DISTANCE);

// Define Variables

float hum;    // Stores humidity value in percent
float temp;   // Stores temperature value in Celcius
float duration1; // Stores First HC-SR04 pulse duration value
float duration2; // Stores Second HC-SR04 pulse duration value
float distance1; // Stores calculated distance in cm for First Sensor
float distance2; // Stores calculated distance in cm for Second Sensor
float soundsp;  // Stores calculated speed of sound in M/S
float soundcm;  // Stores calculated speed of sound in cm/ms
int iterations = 5;

// Initialize DHT sensor for normal 16mhz Arduino

DHT dht(DHTPIN, DHTTYPE);

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

void loop()
{

  delay(1000);  // Delay so DHT-22 sensor can stabalize

    hum = dht.readHumidity();  // Get Humidity value
    temp= dht.readTemperature();  // Get Temperature value

    // Calculate the Speed of Sound in M/S
    soundsp = 331.4 + (0.606 * temp) + (0.0124 * hum);

    // Convert to cm/ms

                soundcm = soundsp / 10000;

                // Measure duration for first sensor

  duration1 = sonar1.ping_median(iterations);

  // Add a delay between sensor readings

  delay(1000);

  // Measure duration for first sensor

  duration2 = sonar2.ping_median(iterations);

  // Calculate the distances for both sensors

  distance1 = (duration1 / 2) * soundcm;
  distance2 = (duration2 / 2) * soundcm;

  // Send results to Serial Monitor

    Serial.print("Distance 1: ");

    if (distance1 >= 400 || distance1 <= 2) {
    Serial.print("Out of range");
    }
    else {
    Serial.print(distance1);
    Serial.print(" cm ");
    }

    Serial.print("Distance 2: ");

    if (distance2 >= 400 || distance2 <= 2) {
```

```
    Serial.print("Out of range");
    }
    else {
    Serial.print(distance2);
    Serial.print(" cm");
    }

  Serial.println(" ");
}
```

Be Kind to Animals!

One thing that bears consideration when designing a device around any ultrasonic emitter, like the HC-SR04, is that many animals can hear ultrasonic sound. According to WikiPedia this includes dogs, cats, gerbils and rabbits.

If you have furry 4-legged friends roaming your house you might want to refrain from subjecting them to the sound emitted by the sensor as it could be very annoying to them. Imagine hearing a high pitched series of "beeps" every second nonstop and you can empathize with what your pet might be going through.

There are other methods of measuring distance like IR light and LIDAR that don't involve ultrasonic sound, so you might want to look into that if animals are a consideration in your home. At the very least it's probably a good idea in general to keep Rover or Fifi out of the workshop!

Moving On

As you've see the HC-SR04 Ultrasonic Distance Sensor is an inexpensive yet useful device that can be used for a myriad of applications. If you are into building robots this is an essential component in your toolkit.

Hopefully you have found this article to be useful. If you have any questions about the HC-SR04 or the sketches I've presented here please write them in the comments section below. And if you come up with a project based around the HC-SR04 I'd love to hear about it.

So let's get sensing – I'll ping you later!

Parts List

Here are some components that you might need to complete the experiments in this article. Please note that some of these links may be affiliate links, and the DroneBot Workshop may receive a commission on your purchases. This does not increase the cost to you and is a method of supporting this ad-free website.

COMING SOON!

Resources

HC-SR04 Sketches   All of the sketches used in this article.

Arduino Site NewPing Article   The NewPing library as described on the official Arduino website.

NewPing Library on BitBucket   The NewPing Library repository on BitBucket. Go here for the latest version.

Adafruit DHT22 Library   The Adafruit Library for DHT22 on GitHub. You can also just install this from your Library Manager in the Arduino IDE.

Adafruit Unified Sensor Library   The Adafruit Unified Sensor Library on GitHub. Again you can just install this from your Library Manager.