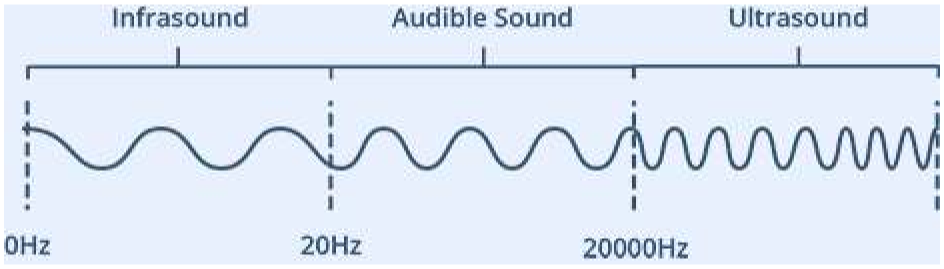
How HC-SR04 Ultrasonic Sensor Works & Interface It With Arduino

Give your next Arduino project bat-powers with the HC-SR04 Ultrasonic Distance Sensor that can report the range of objects up to 13 feet away. This is a good thing to know when you’re trying to save your robot from hitting a wall.

They are low power (suitable for battery operated devices), affordable, easy to interface and extremely popular with hobbyists.

Ultrasound is a high-pitched sound wave whose frequency exceeds the audible range of human hearing.



Humans can hear sound waves that vibrate in the range of about 20 times a second (a deep rumbling noise) to 20,000 times a second (a high-pitched whistle). However, ultrasound has a frequency of more than 20,000 Hz and is therefore inaudible to humans.

HC-SR04 Hardware Overview

An HC-SR04 ultrasonic distance sensor actually consists of two ultrasonic transducers.

One acts as a transmitter that converts the electrical signal into 40 KHz ultrasonic sound pulses. The other acts as a receiver and listens for the transmitted pulses.

When the receiver receives these pulses, it produces an output pulse whose width is proportional to the distance of the object in front.

This sensor provides excellent non-contact range detection between 2 cm to 400 cm (~13 feet) with an accuracy of 3 mm.

Since it operates on 5 volts, it can be connected directly to an Arduino or any other 5V logic microcontroller.

Technical Specifications

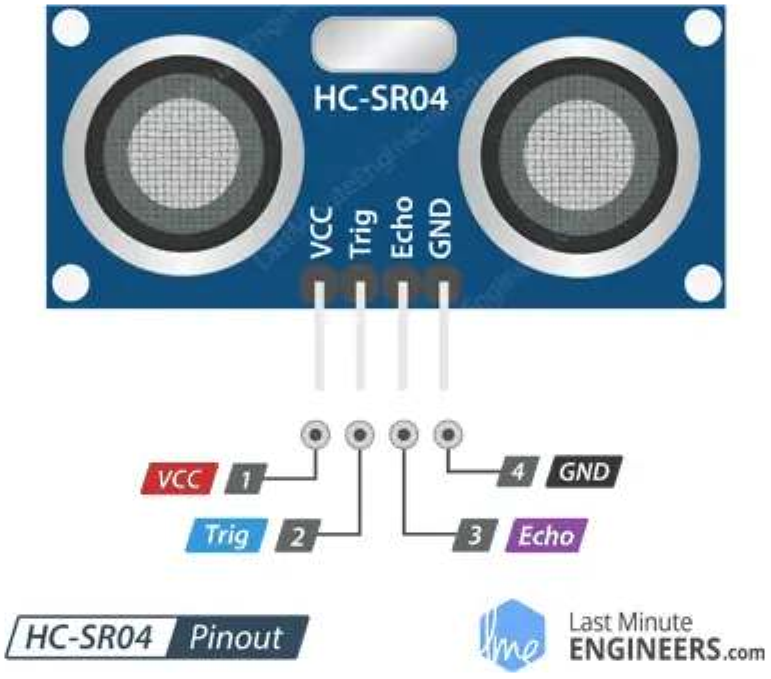
Here are the specifications:

Operating Voltage	DC 5V
Operating Current	15mA

Operating Frequency	40KHz
Max Range	4m
Min Range	2cm
Ranging Accuracy	3mm
Measuring Angle	15 degree
Trigger Input Signal	10µS TTL pulse
Dimension	45 x 20 x 15mm

HC-SR04 Ultrasonic Sensor Pinout

Let’s take a look at its pinout.



VCC supplies power to the HC-SR04 ultrasonic sensor. You can connect it to the 5V output from your Arduino.

Trig (Trigger) pin is used to trigger ultrasonic sound pulses. By setting this pin to HIGH for 10µs, the sensor initiates an ultrasonic burst.

Echo pin goes high when the ultrasonic burst is transmitted and remains high until the sensor receives an echo, after which it goes low. By measuring the time the Echo pin stays high, the distance can be calculated.

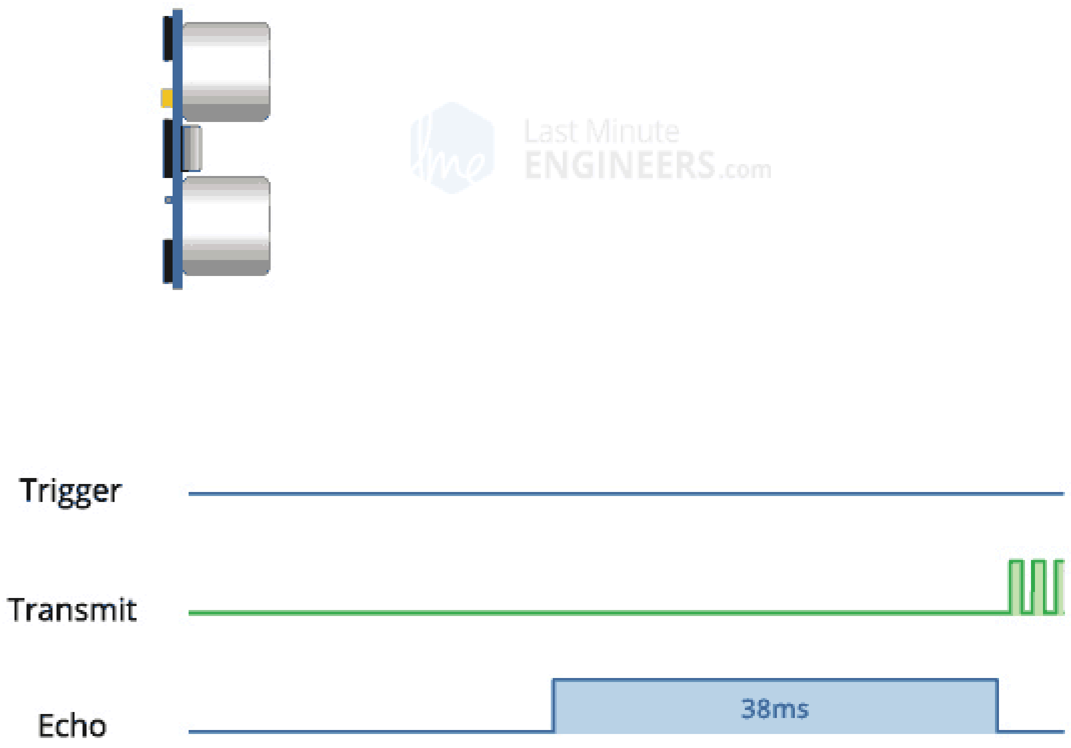
GND is the ground pin. Connect it to the ground of the Arduino.

How Does HC-SR04 Ultrasonic Distance Sensor Work?

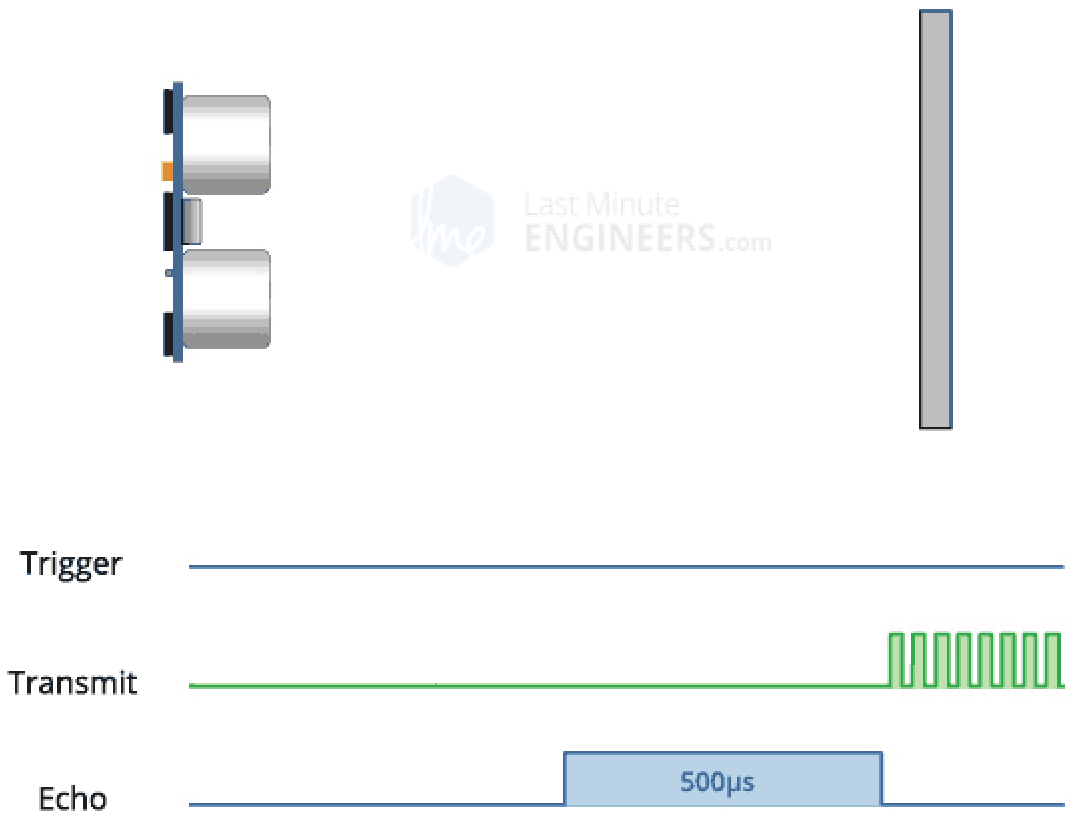
It all starts when the trigger pin is set HIGH for 10µs. In response, the sensor transmits an ultrasonic burst of eight pulses at 40 kHz. This 8-pulse pattern is specially designed so that the receiver can distinguish the transmitted pulses from ambient ultrasonic noise.

These eight ultrasonic pulses travel through the air away from the transmitter. Meanwhile the echo pin goes HIGH to initiate the echo-back signal.

If those pulses are not reflected back, the echo signal times out and goes low after 38ms (38 milliseconds). Thus a pulse of 38ms indicates no obstruction within the range of the sensor.

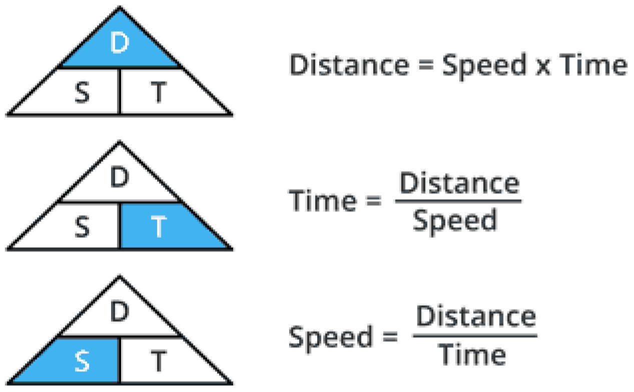


If those pulses are reflected back, the echo pin goes low as soon as the signal is received. This generates a pulse on the echo pin whose width varies from 150 μs to 25 ms depending on the time taken to receive the signal.



Calculating the Distance

The width of the received pulse is used to calculate the distance from the reflected object. This can be worked out using the simple distance-speed-time equation we learned in high school. An easy way to remember the equation is to put the letters in a triangle.



Let us take an example to make it more clear. Suppose we have an object in front of the sensor at an unknown distance and we receive a pulse of 500µs width on the echo pin. Now let’s calculate how far the object is from the sensor. For this we will use the below equation.

Distance = Speed x Time

Here we have the value of time i.e. 500 µs and we know the speed. Of course it's the speed of sound! It is 340 m/s. To calculate the distance we need to convert the speed of sound into cm/µs. It is 0.034 cm/µs. With that information we can now calculate the distance!

Distance = 0.034 cm/µs x 500 µs

But we're not done yet! Remember that the echo pulse indicates the time it takes for the signal to be sent and reflected back. So to get the distance, you have to divide your result by two.

Distance = (0.034 cm/µs x 500 µs) / 2

Distance = 8.5 cm

Now we know that the object is 8.5 cm away from the sensor.

Wiring an HC-SR04 Sensor to an Arduino

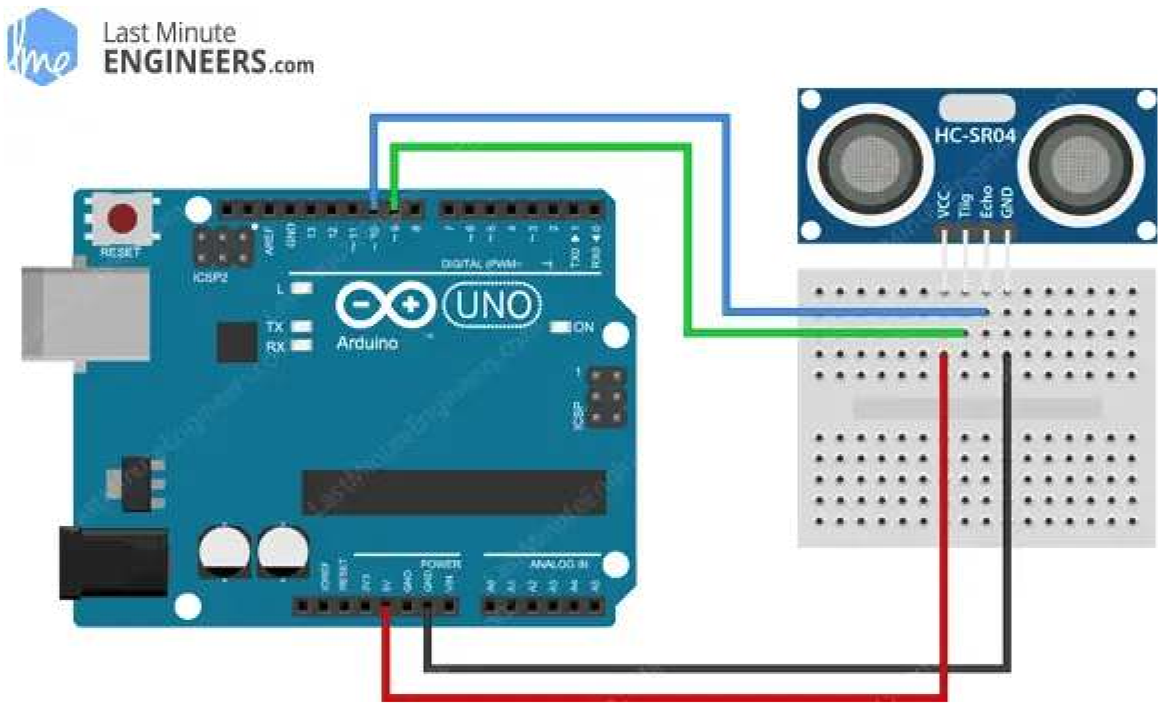
Now that we have a complete understanding of how the HC-SR04 ultrasonic sensor works we can start connecting it to our Arduino!

Connecting the HC-SR04 to Arduino is very easy. Start by placing the sensor on your breadboard. Connect the VCC pin to the 5V pin on the Arduino and the GND pin to the ground pin. Now connect the trig and echo pins to digital pins #9 and #10 respectively.

The following table lists the pin connections:

HC-SR04 Sensor	Arduino
VCC	5V
Trig	9
Echo	10
GND	GND

When you are done you should have something that looks similar to the image shown below.



Wiring HC-SR04 Ultrasonic Sensor to Arduino UNO – Normal Mode

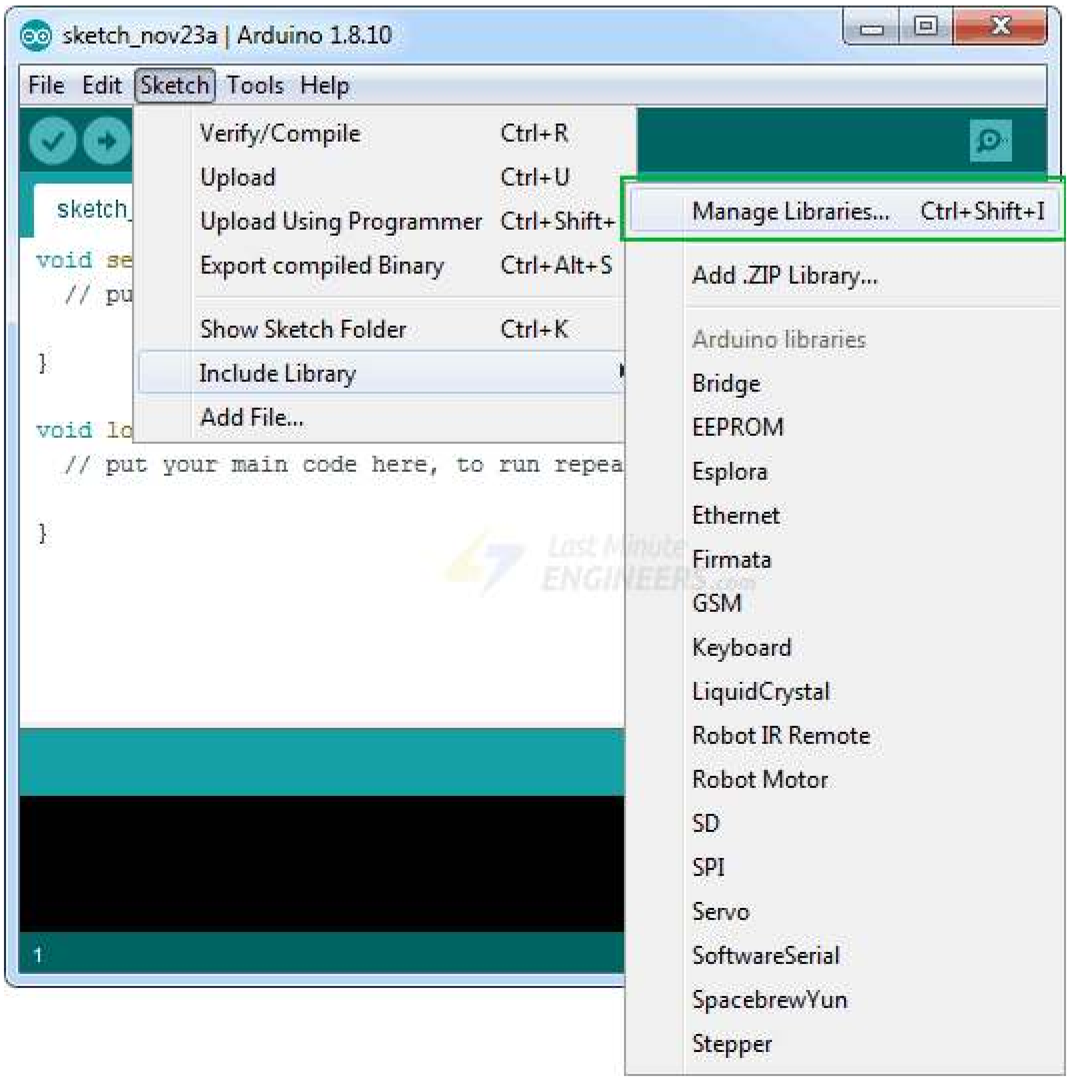
Library Installation

Triggering the ultrasonic sensor and measuring the received signal pulse width manually is a lot of work but luckily there are many libraries available to us. One of the popular libraries is the NewPing library. This is the library we will use in our examples.

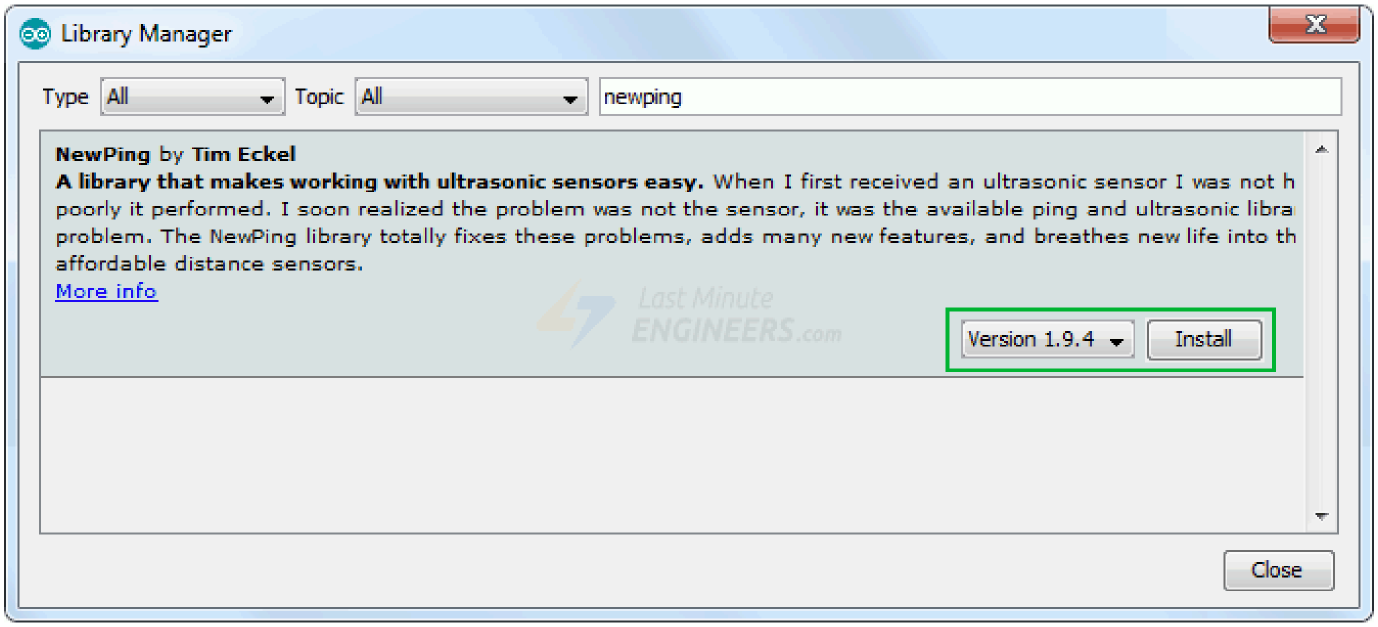
The NewPing library is quite advanced. It supports up to 15 ultrasonic sensors at once and can output directly in centimeters, inches, or time periods.

This library is not included in the Arduino IDE, so you will need to install it first.

To install the library navigate to Sketch > Include Libraries > Manage Libraries… Wait for Library Manager to download the library index and update the list of installed libraries.



Filter your search by typing ‘newping’. Click on the first entry and then select Install.



newping

NewPing by **Tim Eckel**
A library that makes working with ultrasonic sensors easy. When I first received an ultrasonic sensor I was not h
poorly it performed. I soon realized the problem was not the sensor, it was the available ping and ultrasonic libra
problem. The NewPing library totally fixes these problems, adds many new features, and breathes new life into th
affordable distance sensors.

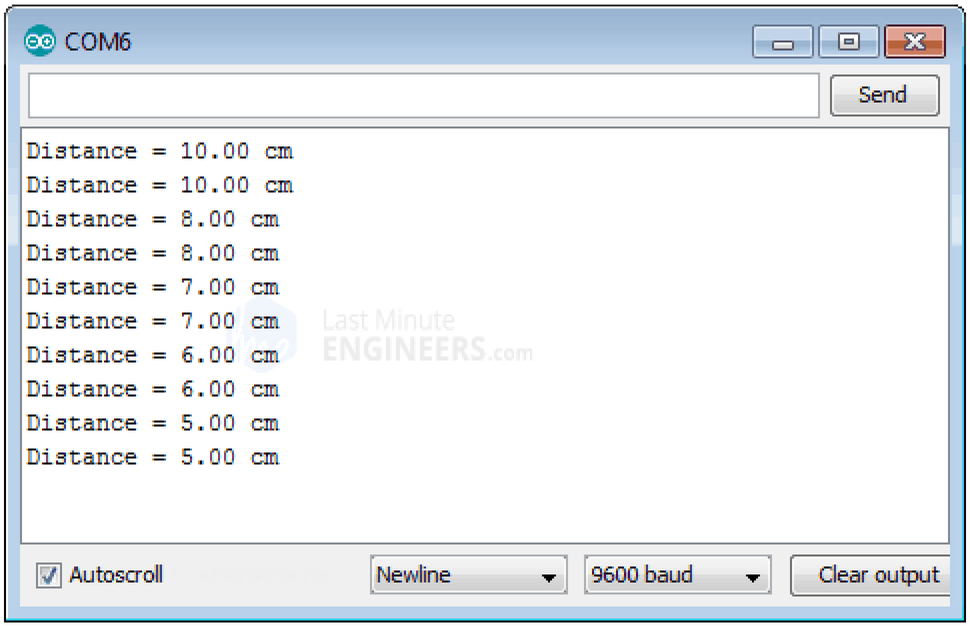
Version 1.9.4

Arduino Example Code

Here is a simple sketch that uses the serial monitor to display a distance measured in centimeters. Give this sketch a try before we start a detailed analysis of it.

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

// Hook up HC-SR04 with Trig to Arduino Pin 9, Echo to Arduino pin 10
#define TRIGGER_PIN 9
#define ECHO_PIN 10

// Maximum distance we want to ping for (in centimeters).
#define MAX_DISTANCE 400

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

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

void loop() {
    Serial.print("Distance = ");
    Serial.print(sonar.ping_cm());
    Serial.println(" cm");
    delay(500);
}
```

Once the sketch is uploaded, open your serial monitor, set the baud rate to 9600 bps. Try pointing the sensor at objects lying around you. You should see the measured distance begin to stream by.



```
Distance = 10.00 cm
Distance = 10.00 cm
Distance = 8.00 cm
Distance = 8.00 cm
Distance = 7.00 cm
Distance = 7.00 cm
Distance = 6.00 cm
Distance = 6.00 cm
Distance = 5.00 cm
Distance = 5.00 cm
```

Output on Serial Monitor

Code Explanation:

The sketch starts by including the newly installed NewPing library.

```
#include "NewPing.h"
```

First the Arduino pins are defined to which the Trig and Echo pins of the HC-SR04 are connected. We have also defined a constant called `MAX_DISTANCE`. It will set a maximum distance where pings beyond that distance are read as no ping “clear”. `MAX_DISTANCE` is currently set to 400 [default = 500cm].

```
#define TRIGGER_PIN 9
#define ECHO_PIN 10
#define MAX_DISTANCE 400
```

After this, an instance of NewPing library named `sonar` is created.

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

In the setup, we initialize the serial communication with PC.

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

In the loop, we simply call the `ping_cm()` function and print the result on the serial monitor. This function sends a ping and returns the distance in centimeters.

```
void loop() {
    Serial.print("Distance = ");
    Serial.print(sonar.ping_cm());
    Serial.println(" cm");
    delay(500);
}
```

Other useful functions in NewPing Library

There are a few useful functions you can use with NewPing object.

- Above sketch returns the distance in centimeters. If you want result to be in inches, use `sonar.ping_in()` function.

  ```
  Serial.print(sonar.ping_in());
  ```

- The sketch above only has a resolution of one centimeter. If you want to get the result in decimal form you can use NewPing in duration mode instead of distance mode. You need to change this line:

  ```
  Serial.print(sonar.ping_cm());
  ```

  with below line

  ```
  Serial.print((sonar.ping() / 2) * 0.0343);
  ```

- There is a method called `ping_median(iterations)` in the NewPing library to improve the accuracy of your HC-SR04. This method takes multiple measurements instead of just one, discards out-of-range readings, and then averages the remaining readings. By default it only takes 5 readings but you can specify as many as you want.

  ```
  int iterations = 5;
  Serial.print((sonar.ping_median(iterations) / 2) * 0.0343);
  ```

Arduino Project – Contactless Distance Finder

Let’s create a quick project to demonstrate how a simple ultrasonic sensor can be turned into a sophisticated contactless distance finder. In this project we will be using a 16×2 Character LCD which displays a horizontal bar to represent the distance from the object.

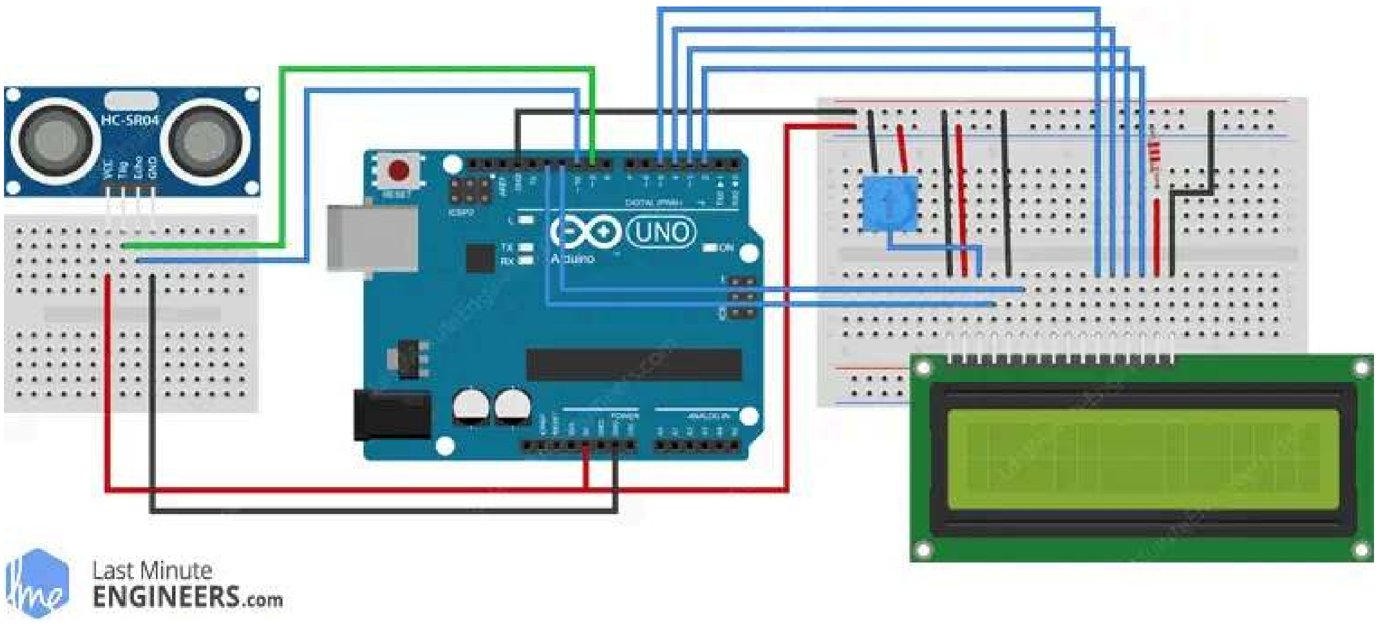
If you’re not familiar with 16×2 character LCDs, consider reading the tutorial below.

Interfacing 16×2 Character LCD Module with Arduino

The Serial Monitor is a handy tool for viewing data from your Arduino, but what if you want to make your project portable and view...

Wiring

Next we need to make the connection to the LCD as shown below.

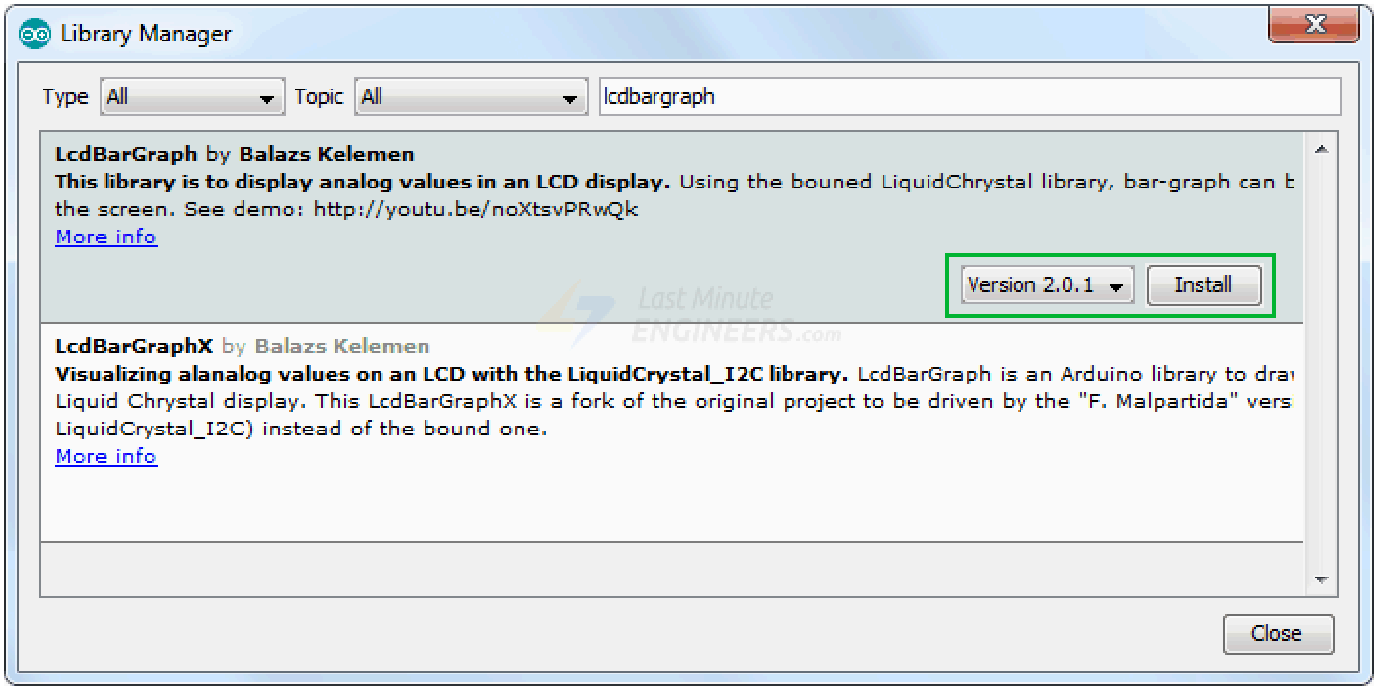


Wiring HC-SR04 Ultrasonic Sensor and 16×2 LCD to Arduino UNO

Library Installation

Before we upload the code and start playing with the sensor we need to install a library called LCDBarGraph. This library will help in drawing a horizontal bar on the LCD, where the length of the bar will represent the distance to the object.

To install the library navigate to Sketch > Include Libraries > Manage Libraries… Wait for Library Manager to download the library index and update the list of installed libraries. Filter your search by typing 'lcdbargraph'. Click on the first entry and then select Install.



LcdBarGraph by **Balazs Kelemen**
This library is to display analog values in an LCD display. Using the bouned LiquidChrystal library, bar-graph can b
the screen. See demo: http://youtu.be/noXtsvPRwQk
More info

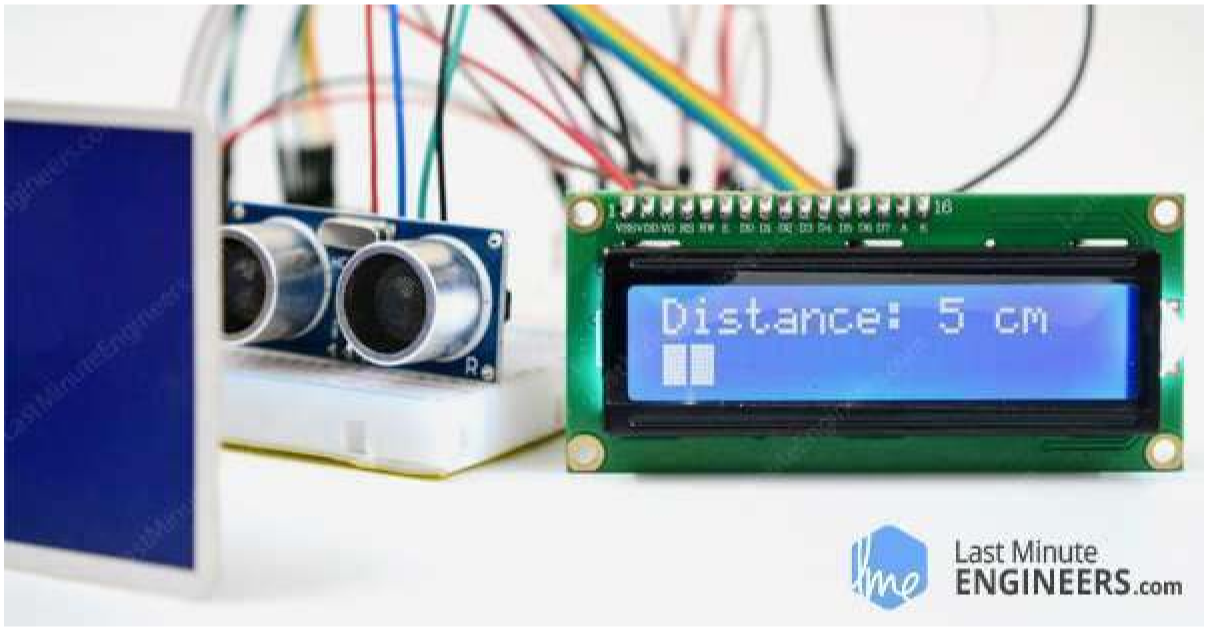
LcdBarGraphX by **Balazs Kelemen**
Visualizing alanalog values on an LCD with the LiquidCrystal_I2C library. LcdBarGraph is an Arduino library to dra
Liquid Chrystal display. This LcdBarGraphX is a fork of the original project to be driven by the "F. Malpartida" vers
LiquidCrystal_I2C) instead of the bound one.
More info

Arduino Code

Once you have installed the library, try the below sketch.

```
// includes the LiquidCrystal Library
#include <LiquidCrystal.h>

// includes the LcdBarGraph Library
#include <LcdBarGraph.h>

// Maximum distance we want to ping for (in centimeters).
#define max_distance 200

// Creates an LCD object. Parameters: (rs, enable, d4, d5, d6, d7)
LiquidCrystal lcd(12, 11, 5, 4, 3, 2);

LcdBarGraph lbg(&lcd, 16, 0, 1); // Creates an LCD Bargraph object.

const int trigPin = 9;
const int echoPin = 10;
long duration;
int distance;

void setup()
{
    lcd.begin(16,2); // Initializes the interface to the LCD screen

    pinMode(trigPin, OUTPUT);
    pinMode(echoPin, INPUT);
}
```

The output looks like this.



Bargraph Output on 16×2 Character LCD

Code Explanation:

First of all, you have to set up the Liquid Crystal Library as usual. Next, create an LcdBarGraph instance with the LiquidCrystal instance you just created. You should reference

LiquidCrystal to the constructor of LcdBarGraph.

The constructor of LcdBarGraph takes three more parameters. The second is the number of the Character column in the LCD (in our case it is 16). The last two parameters are optional and allow custom positioning of the bar.

```
// creating bargraph instance
LcdBarGraph lbg(&lcd, 16, 0, 1);
```

After calculating the distance from the sensor we use `drawValue(value, maxValue)` function to display the bargraph. It draws a bargraph with a value between `0` and `maxValue`.