

Lab: DC Motor Control Using an H-Bridge

Originally written on July 1, 2014 by Matt Richardson
Last modified on September 2, 2016 by Benedetta Piantella Simeonidis

Introduction

In this tutorial, you'll learn how to control a DC motor's direction using an H-bridge.

Contents [\[show\]](#)

To reverse a DC motor, you need to be able to reverse the direction of the current in the motor. The easiest way to do this is using an H-bridge circuit. There are many different models and brands of H-Bridge. This tutorial uses one of the most basic, a [Texas Instruments L293NE](#) or a [Texas Instruments SN754410](#).

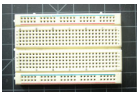
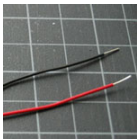




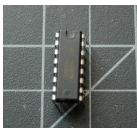

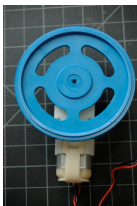
If you simply want to turn a motor on and off, and don't need to reverse it, for example if you're controlling a fan, try the tutorial on [controlling high current loads with transistors](#).

What You'll Need to Know

To get the most out of this lab, you should be familiar with the following concepts. You can check how to do so in the links below:

- [Electronic components](#)
- [Digital input with Arduino](#)
- [How to use transistors to control high current loads](#)
- [How to use transistors to control high current loads with Arduino](#)
- [Video: H-Bridges](#)

Things You'll Need

For this lab you'll need:				
				
Solderless Breadboard	22-AWG hookup wire	Arduino microcontroller	Light emitting diodes – LED	10Kohm resistors
				
Switch	L293NE or SN754410 H-bridge	12V DC power supply	DC motor	

Search

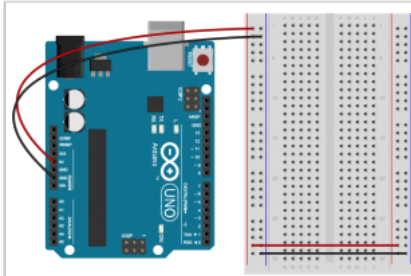
Find posts by category

- ☐ @ITP
- ☐ Analog
- ☐ Code
- ☐ Components
- ☐ Digital
- ☐ Electronics
- ☐ Lab
- ☐ Motors
- ☐ Mouse
- ☐ Multimeter
- ☐ Node.js
- ☐ P5.js
- ☐ Processing
- ☐ Programming
- ☐ PWM
- ☐ Sensors
- ☐ Serial Communication
- ☐ Sound
- ☐ Switches
- ☐ Transistor
- ☐ Uncategorized
- ☐ Video
- ☐ Wireless

Filter

Prepare the breadboard

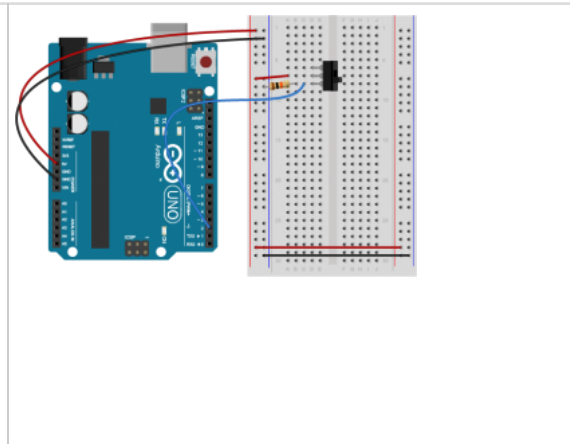
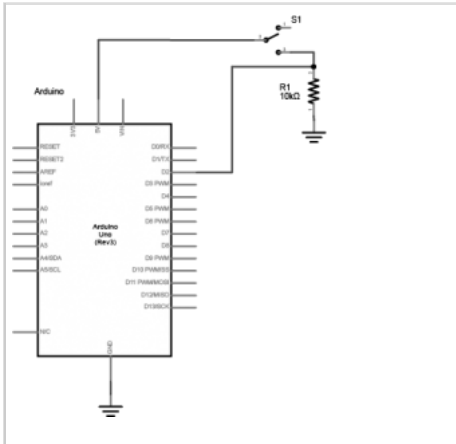
Connect power and ground on the breadboard to power and ground from the microcontroller. On the Arduino module, use the 5V and any of the ground connections:



Made with Fritzing

Add a Digital Input (a switch)

Connect a switch to digital input 2 on the Arduino.



Find a motor

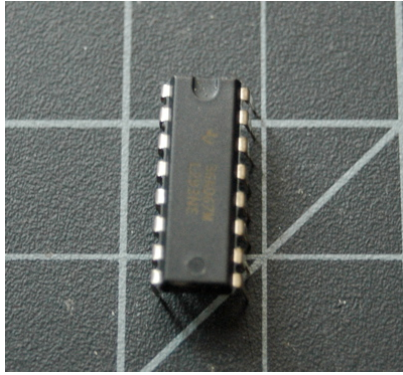
Find yourself a DC motor that runs on low DC voltage within the range of 5 – 15V. RadioShack often sells several small DC motors, the NYU Book Store on occasion has a few, the junk shelf is almost always a goldmine for discarded motors and fans. Asking classmates and second years is another good approach.

Solder leads to the motor's terminals. With DC motors, there is no polarity regarding the motor terminals so you can connect it any way you'd like.

Optional: Consider testing your motor with a bench power supply from the equipment room. Ask a teacher or resident if you need help setting one up. Begin by adjusting the voltage on the bench power supply and observe its effects. Take note of its speed at different voltages without dipping to low or too high.

Safety Warning: Running a motor at a voltage much lower or much higher than what it's rated for could potentially damage or permanently destroy your motor. When the motor doesn't spin, the voltage is too low. When the motor runs hot, or sounds like it's straining, the voltage is too high.

Set up the H-bridge



L293NE H-bridge

This example uses an H-bridge integrated circuit, the [Texas Instruments L293NE](#) or [Texas Instruments SN754410](#). There is one in your Physical Computing Kit, and the NYU Book Store and many distributors such as [Digikey](#), [SparkFun](#), [Mouser](#) and [Jameco](#) sell them as well.

How your H-bridge works

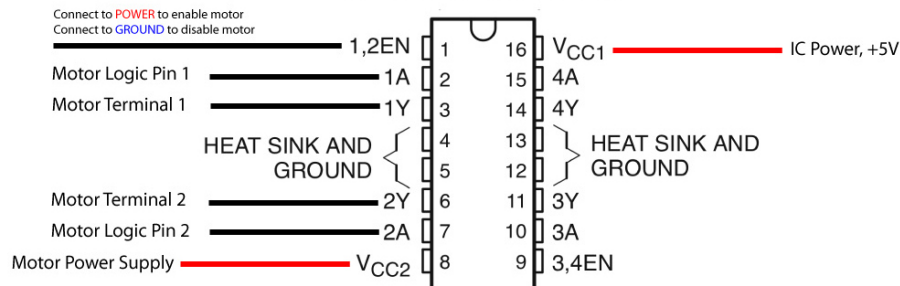
The L293NE/SN754410 is a very basic H-bridge. It has two bridges, one on the left side of the chip and one on the right, and can control 2 motors. It can drive up to 1 amp of current, and operate between 4.5V and 36V. The small DC motor you are using in this lab can run safely off a low voltage so this H-bridge will work just fine.

The H-bridge has the following pins and features:

- Pin 1 (1,2EN) enables and disables our motor whether it is give HIGH or LOW
- Pin 2 (1A) is a logic pin for our motor (input is either HIGH or LOW)
- Pin 3 (1Y) is for one of the motor terminals
- Pin 4-5 are for ground
- Pin 6 (2Y) is for the other motor terminal
- Pin 7 (2A) is a logic pin for our motor (input is either HIGH or LOW)
- Pin 8 (VCC2) is the power supply for our motor, this should be given the rated voltage of your motor
- Pin 9-11 are unconnected as you are only using one motor in this lab
- Pin 12-13 are for ground
- Pin 14-15 are unconnected
- Pin 16 (VCC1) is connected to 5V

Below is a diagram of the H-bridge and which pins do what in our example. Included with the diagram is a truth table indicating how the motor will function according to the state of the logic pins (which are set by our Arduino).

L293NE or SN754410



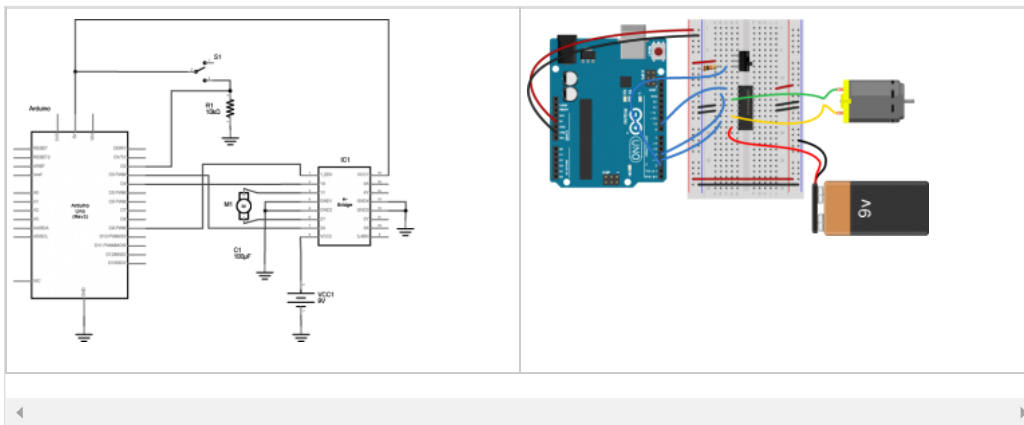
EN	1A	2A	FUNCTION
H	L	H	Turn right
H	H	L	Turn left
H	L	L	Fast motor stop
H	H	H	Fast motor stop
L	X	X	Fast motor stop

L = low, H = high, X = don't care

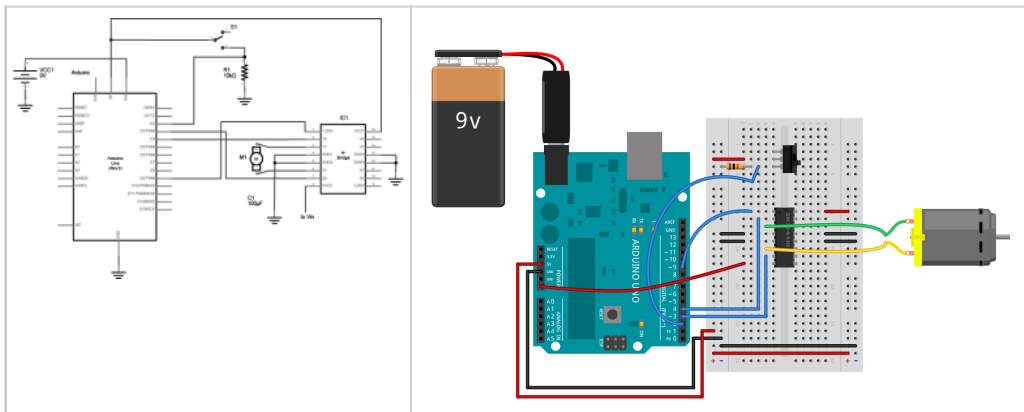
For this lab, the enable pin connects to a digital pin on your Arduino so you can send it either HIGH or LOW and turn the motor ON or OFF. The motor logic pins are also connected to designated digital pins on your Arduino so you can send it HIGH and LOW to have the motor turn in one direction, or LOW and HIGH to have it turn in the other direction. The motor supply voltage connects to the voltage source for the motor, which is usually an external power supply. If your motor can run on 5V and less than 500mA, you can use the Arduino's 5V output. **Most motors require a higher voltage and higher current draw than this, so you will need an external power supply.**

Connect the motor to the H-bridge

Connect the motor to the H-bridge as follows:



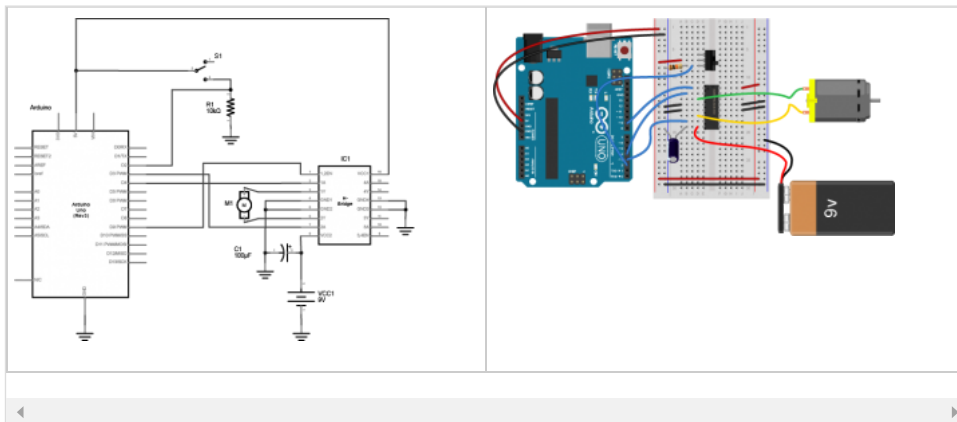
Or, if you are using an external power supply for the Arduino, you can use the Vin pin.



If you need an external power supply, you can use any DC power supply or battery from 9 – 15V as long as your motor can run at that voltage, and as long as the supply can supply as much current as your motor needs. Related video: [Connect 12V Power Supply, Wiring a power supply using a DC power jack](#)

Plug an external DC power source into the Arduino's external power input. You may still leave your USB cable plugged in for quick and easy reprogramming. Whichever motor you use, make sure the power source is compatible (i.e. don't use a 9V battery for a 3V motor!). The external voltage input is available at the Vin pin, so you can use it both to power the Arduino, and to power the motor.

Note on decoupling capacitors



If you find that your microcontroller is resetting whenever the motor turns on, add a capacitor across power and ground close to the motor. The capacitor will smooth out the voltage dips that occur when the motor turns on. This use of a capacitor is called a **decoupling capacitor**. Usually a 10 – 100uF capacitor will work. The larger the cap, the more charge it can hold, but the longer it will take to release its charge.

Program the microcontroller

Program the microcontroller to run the motor through the H-bridge. First set up constants for the switch pin, the two H-bridge pins, and the enable pin of the H-bridge. Use one of the analogWrite pins (3,5,6,9,10, or 11) for the enable pin.

```
const int switchPin = 2;    // switch input
const int motor1Pin = 3;    // H-bridge leg 1 (pin 2, 1A)
const int motor2Pin = 4;    // H-bridge leg 2 (pin 7, 2A)
const int enablePin = 9;    // H-bridge enable pin
```

In the setup(), set all the pins for the H-bridge as outputs, and the pin for the switch as an input. Then set the enable pin high so the H-bridge can turn the motor on.

```
void setup() {
  // set the switch as an input:
  pinMode(switchPin, INPUT);

  // set all the other pins you're using as outputs:
  pinMode(motor1Pin, OUTPUT);
  pinMode(motor2Pin, OUTPUT);
  pinMode(enablePin, OUTPUT);
  pinMode(ledPin, OUTPUT);

  // set enablePin high so that motor can turn on:
  digitalWrite(enablePin, HIGH);
}
```

In the main loop() read the switch. If it's high, turn the motor one way by taking one H-bridge pin high and the other low. If the switch is low, reverse the direction by reversing the states of the two H-bridge pins.

```
void loop() {
```

```
// if the switch is high, motor will turn on one direction:
if (digitalRead(switchPin) == HIGH) {
  digitalWrite(motor1Pin, LOW); // set leg 1 of the H-bridge low
  digitalWrite(motor2Pin, HIGH); // set leg 2 of the H-bridge high
}
// if the switch is low, motor will turn in the other direction:
else {
  digitalWrite(motor1Pin, HIGH); // set leg 1 of the H-bridge high
  digitalWrite(motor2Pin, LOW); // set leg 2 of the H-bridge low
}
```

Once you've seen this code working, try modifying the speed of the motor using the `analogWrite()` function, as explained in the [Analog Lab](#). Use `analogWrite()` on the enable pin of the motor, and see what happens as you change the value of the `analogWrite()`.

Get creative

This is a suggestion for a possible project. It's not a requirement for the class homework.

Use your motor to make something move, vibrate, rise, fall, roll, creep, or whatever you can think of, in response to user input from a digital input device (switch, floor sensor, tripwire, etc). Look inside moving toys, you'll find a number of excellent motors and gears you can re-purpose. See the innards of a cymbal monkey below as an example. Perhaps you can re-design the user interface to a toy, using the microcontroller to mediate between new sensors on the toy and the motors of the toy. Whatever you build, make sure it reacts in some way to human action.

