DC Motor Control Using an H-Bridge

Overview

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

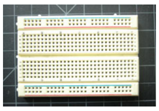
To reverse a DC motor, you need to be able to reverse the direction of the current in the motor. The easiest way to do this is using an H-bridge circuit. There are many different models and brands of H-Bridge. This tutorial uses one of the most basic, a Texas Instruments L293NE or a Texas Instruments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

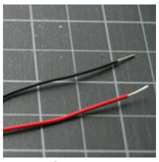
If you simply want to turn a motor on and off, and don't need to reverse it, for example if you're controlling a fan, try the tutorial on controlling high current loads with transistors.

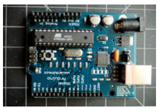
Table of Contents (hide)

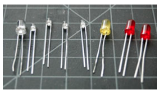
1. Parts

For this lab you'll need:


Solderless breadboard

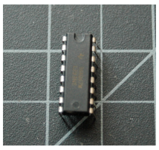

22-AWG hookup wire


Arduino Microcontroller module

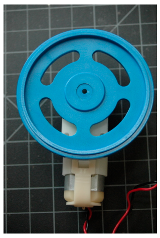

Light Emiting Diodes, LED


10Kohm resistors


Switch

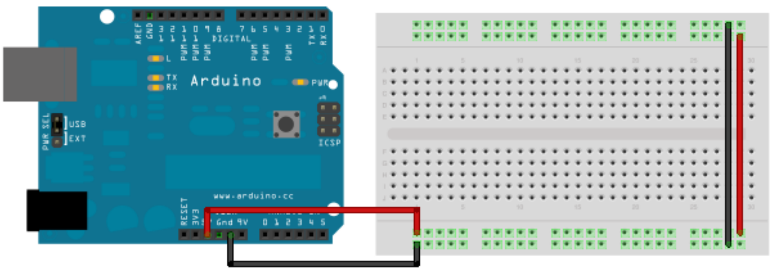

L293NE
or SN754410 H-bridge


12V DC power supply


DC Motor

2. Prepare the breadboard

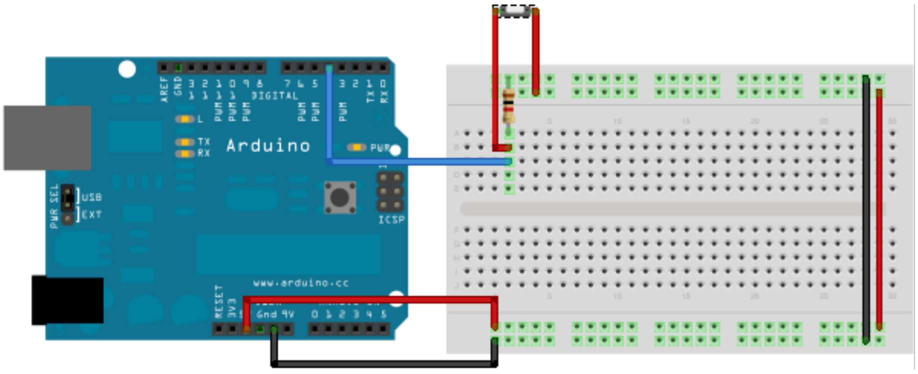
Connect power and ground on the breadboard to power and ground from the microcontroller. On the Arduino module, use the 5V and any of the ground connections:



(Diagram made with Fritzing - download)

3. Add a Digital Input (a switch)

Connect a switch to digital input 2 on the Arduino.



(Diagram made with Fritzing - download)

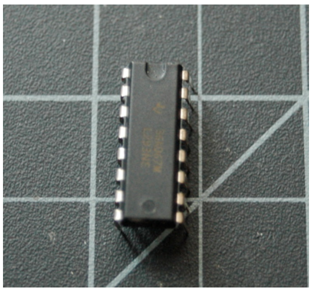
4. Find a motor

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 Computer Store on occasion has small a few, the junk shelf is almost always a goldmine for discarded motors and fans, or simply asking classmates and second years is a good approach to borrowing a motor.

Solder leads to the motor's terminals. With most motors, there is no polarity regarding the motor terminals so you may hook it up any way you'd like.

Now, consider testing your motor with a bench power supply from the ER. 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. 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.

5. Acquire an H-bridge



L293NE H-bridge

This example uses an H-bridge integrated circuit, the Texas Instruments L293NE or Texas Instruments SN754410. If an H-bridge is not available in your Physical Computing Kit, they should have them for purchase at the NYU Computer Store and online at many distributors such as: Digikey, SparkFun, Mouser and Jameco.

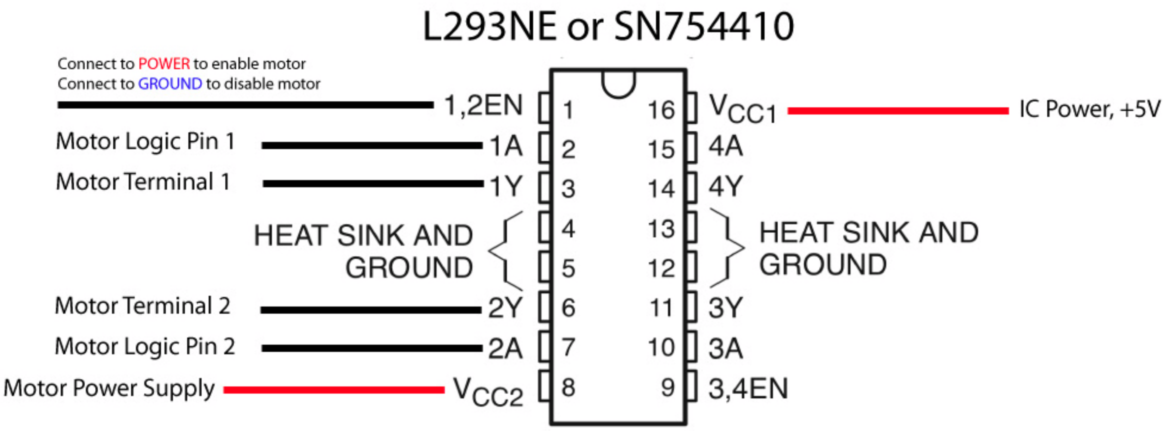
6. How your H-bridge works

The L293NE and SN754410 H-Bridge you are using has 4 half-H bridges, and can therefore 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 5V so this H-bridge will work just fine.

The L293NE and SN754410 is a very basic H-bridge. This one in particular has two bridges, one on the left side of the chip and one on the right. The H-bridge you are using 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 you are using 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).



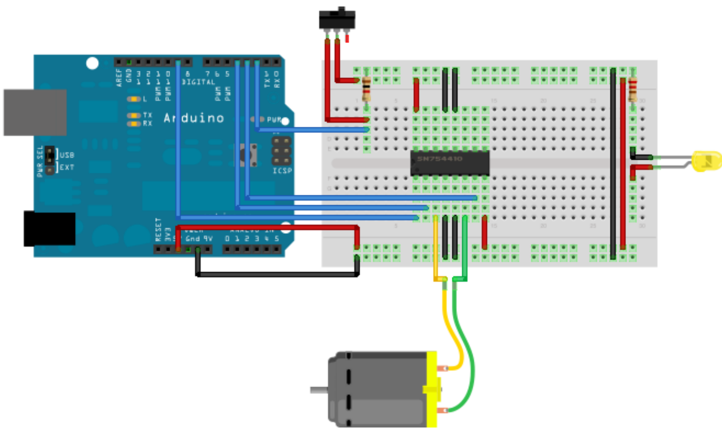
EN	1A	2A	FUNCTION
H	L	H	Turn right
H	H	L	Turn left
H	L	L	Fast motor stop
H	H	H	Fast motor stop
L	X	X	Fast motor stop

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 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. For this lab you will be using the 5V from our Arduino but with aid of our 12V power supply. Most motors require a higher voltage and higher current draw than a microcontroller so the use of an external power supply is often needed.

7. Connect the motor to the H-bridge

Connect the motor to the H-bridge as follows:

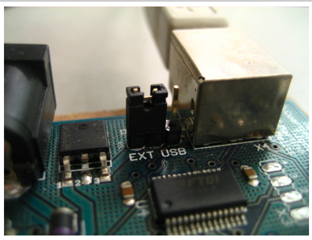


(Diagram made with Fritzing - download)

Most motors consume more current than a microprocessor, and need their own supply. So for this lab you will be using our 12V power supply to provide the current you need.

Change your Arduino's power jumper from USB to EXT so that your Arduino runs off 12V power supply. 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!). If you choose a motor that requires MORE than the 5V your Arduino consider using the 9V power pin or bypassing the Arduino for power all together.

Note the capacitor connecting the motor supply to ground. It smooths out the voltage spikes and dips that occur as the motor turns on and off.



Arduino Power Jumper

8. Program the Microcontroller

Program the microcontroller to run the motor through the H-bridge:

```
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
const int ledPin = 13;      // LED

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);

  // blink the LED 3 times. This should happen only once.
  // if you see the LED blink three times, it means that the module
  // reset itself,. probably because the motor caused a brownout
  // or a short.
  blink(ledPin, 3, 100);
}

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
  }
}

/*
  blinks an LED
 */
void blink(int whatPin, int howManyTimes, int milliSecs) {
  int i = 0;
  for ( i = 0; i < howManyTimes; i++) {
    digitalWrite(whatPin, HIGH);
    delay(milliSecs/2);
    digitalWrite(whatPin, LOW);
    delay(milliSecs/2);
  }
}
```

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 pin 9, the enable pin of the motor, and see what happens as you change the value of the analogWrite().

9. Get creative

This is a suggestion for a possible project. You can do any project you wish as long as it demonstrates your mastery of the lab exercises and good physical interaction.

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.