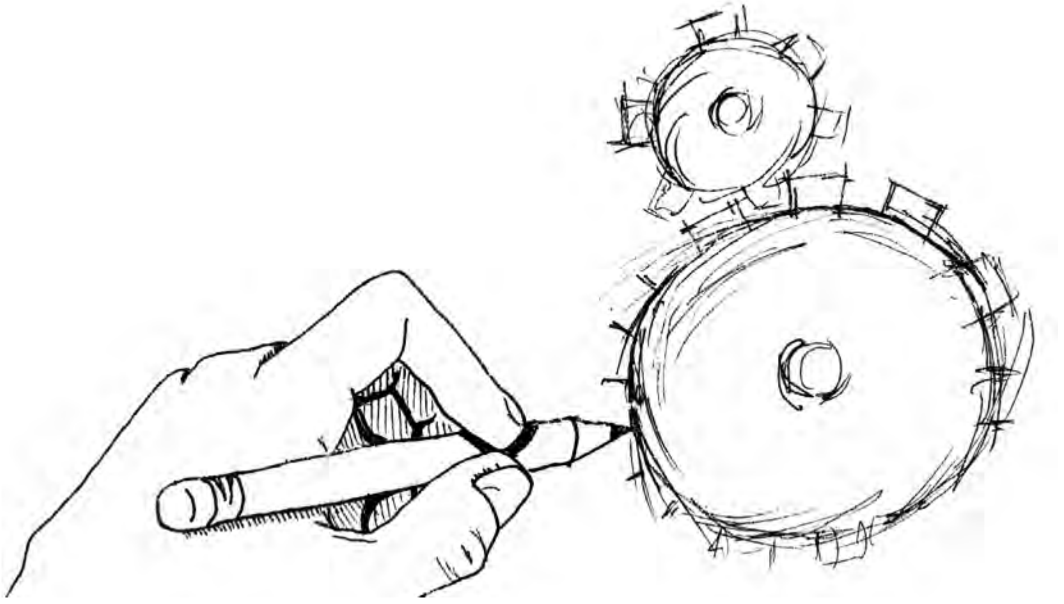
6

Eeny, Meeny, Miny, Motor: Options for Creating and Controlling Motion



The most important component of any moving system is an *actuator*, which is the thing that causes a mechanical system to move. Motors are the most common actuators, and as you'll learn in this chapter, there are many different kinds to choose from for your projects. We'll also cover a few other ways to create motion.

In previous chapters, you learned about force, torque, and power, so by now you have the tools to determine how strong your actuator must be for a specific task. We'll use that information, along with other project-specific requirements, to help us narrow down the available options. This chapter covers a lot of information, so take it slow and don't expect to understand everything on the first pass. Now that you've been warned, let's talk a bit about how motors work.

How Motors Work

Motors turn electrical energy into mechanical energy using coiled-up wires and magnets. When electricity flows through a wire, it creates a magnetic field around it. When you bring a permanent magnet close to that magnetic field, it will be repelled or attracted.

Motors take advantage of this magnetic field by mounting coils of wire on a shaft, so when the magnet repels the coils, the shaft begins to spin. In order to keep the shaft spinning, you need to keep flipping the magnetic field so the series of repel, attract, repel, and so on continues and the shaft keeps spinning. Different motors do this in different ways.

Project 6-1: DIY Motor with Magnet Wire

Let's make a simple motor to better understand how it generates mechanical energy.[1]

Shopping List:

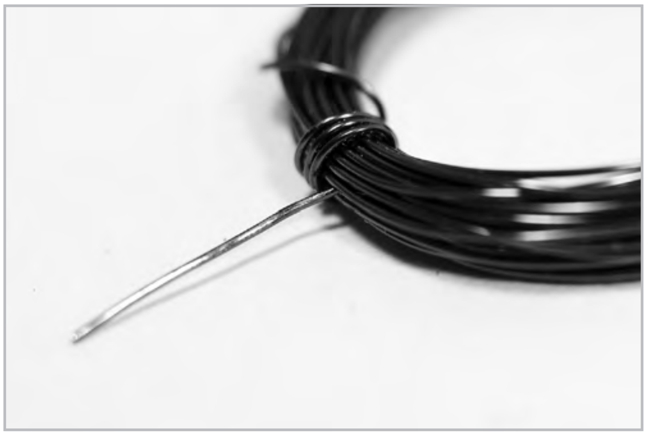
- 10 ft length of magnet wire (RadioShack 278-1345; use the green spool)
- Ceramic disk magnet or other strong magnet (McMaster 5857K15)
- Two big paperclips
- Large eraser, piece of clay, or block of stiff foam

- Two alligator clips (like RadioShack 270-1540)
- 9V battery clip (like RadioShack 270-324)
- 9V battery

Recipe:

1. Measure and cut 10 ft of the green wire.
2. Wrap the green wire tightly around the magnet a bunch of times to form a tight coil. Leave about 1.5 in unwrapped on each end.
3. Remove the coil from around the magnet. Loop the ends inside the coil, then back out, to secure it from unraveling. Make sure the finished coil looks symmetrical.
4. Using a knife, remove the coating from the wire on one side of each end at a 45° angle (see Figure 6-1). Scrape each side of the wire such that when the coil hangs at 45°, the scraped part faces down or up. You should see the shiny copper now.

FIGURE 6-1 Wire coiled with end scraped



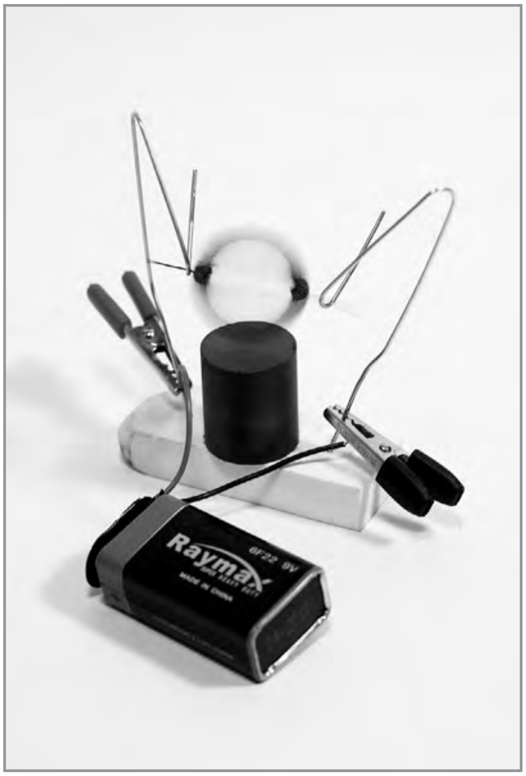
5. Using the paperclips and the eraser, make a cradle for the arms of the wire coil about 1.5 in apart (see Figure 6-2).

6. Place the wire coil in the paperclip cradle. Make sure it spins when you give it a nudge and doesn't get off center. If it does, adjust the wire coil until it looks symmetrical and it balances.

7. Place the magnet on top of the eraser, under the wire coil. Refine the spacing if necessary so the magnet doesn't touch the wire coil or the paperclips.

8. Attach one battery lead wire to the base of each paperclip with an alligator clip.

FIGURE 6-2 DIY motor setup

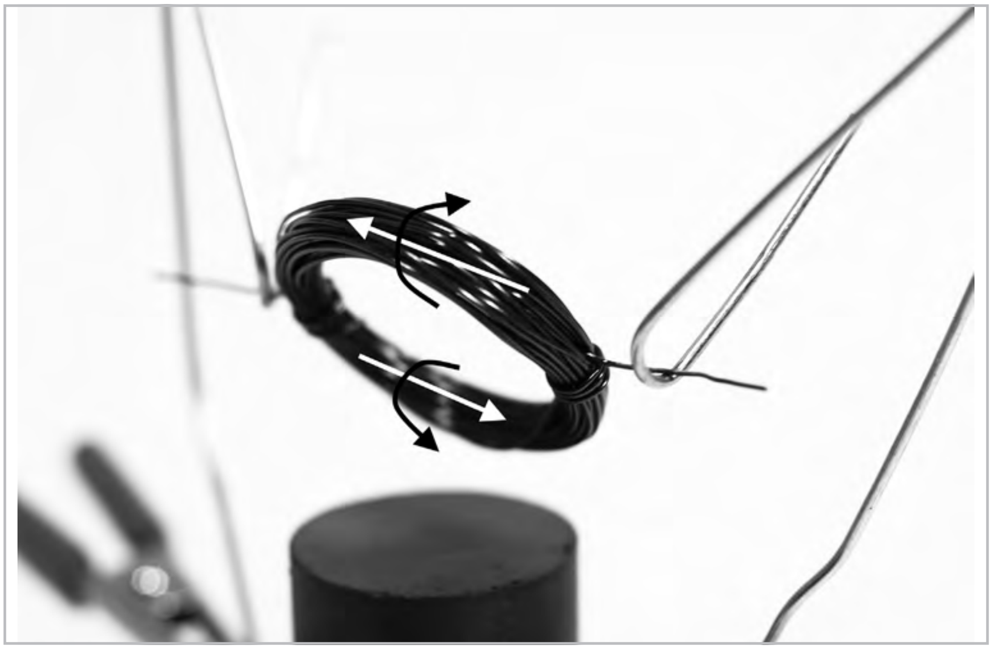


9. Your setup should look like Figure 6-2. Give the wire coil a little nudge, and the reaction between the current flowing through it and the magnet will keep the motor turning!

The wire coil sits on the paperclips, which are conductive. When the side of the wire coil with scraped-off coating makes contact with the paperclips, electricity flows from the battery to the paperclip, then across the wire coil to the other paperclip and back to the battery.

When electric current flows through a wire, it creates a magnetic field around the wire (see Figure 6-3). This magnetic field attracts the magnet sitting directly under the coil. Because the electricity is flowing in the opposite direction on the other side of the coil,

FIGURE 6-3 White arrows show direction of current flow and black arrows show the direction of the resulting magnetic field in the wire coil



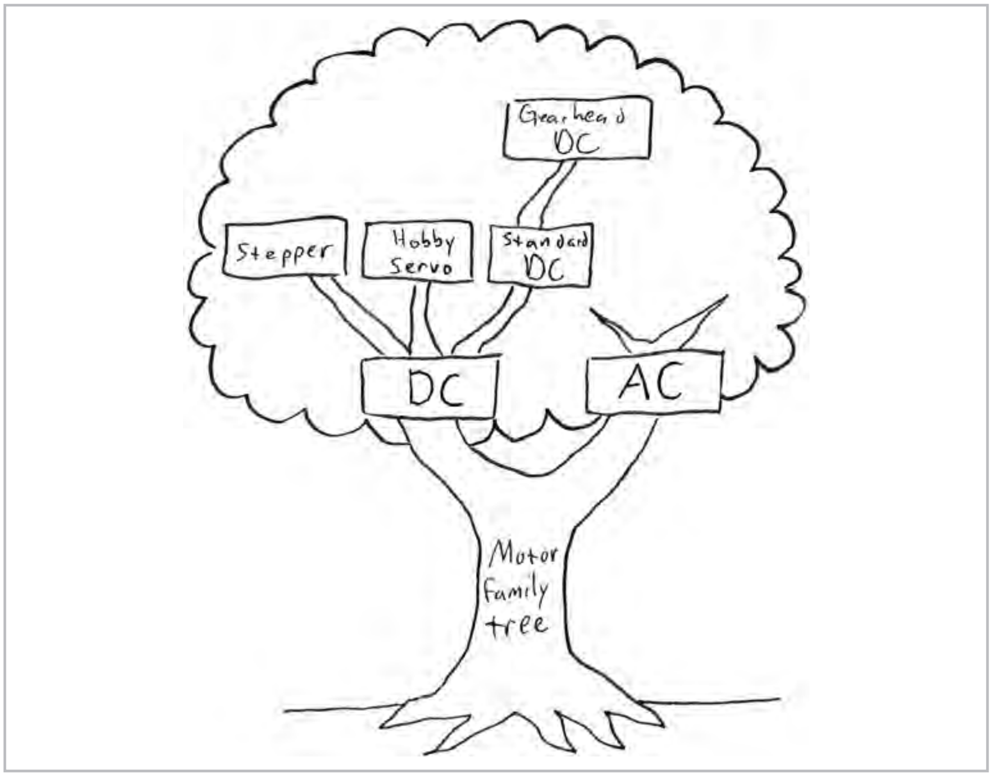
one side will repel the magnet, and the other side will attract it. In order to keep the wire spinning, we need to turn off this flow of current when one side of the coil is close to the magnet, or it will get stuck. So by scraping off the insulation on only one side of the wire, we are telling it to attract, turn off, attract, turn off, attract, and so on, and the momentum of the coil keeps it spinning!

Types of Rotary Actuators

All motors work under the same principles as our DIY motor, but different motors accomplish this in different ways. Each motor type in the motor family has pros and cons, is controlled in a different way, and is well suited to a different set of uses.

The most commonly used type of rotary actuator is the electric motor that spins and creates rotary, or circular, motion. Figure 6-4 shows the rotary motor family tree. There are some cousins I left off the tree, but these are all the motor types we're primarily concerned with in this book.

FIGURE 6-4 Rotary motor family tree



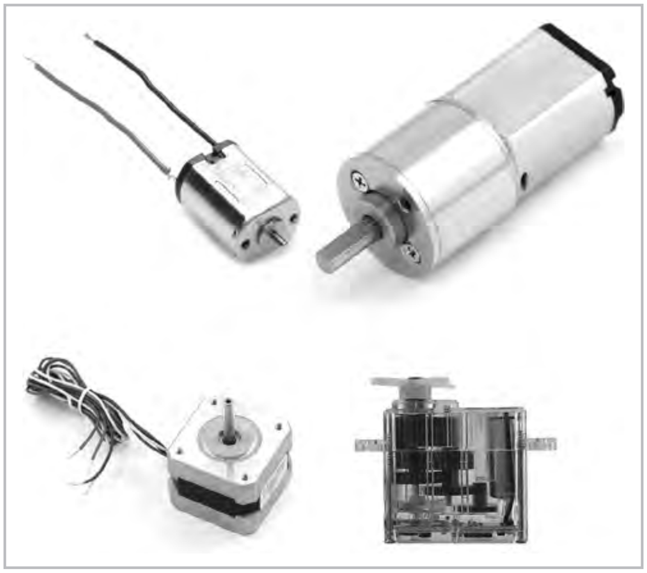
We'll explore each motor's personality by demystifying data sheets for each type.

DC Motors

As you learned in Chapter 5, DC is the kind of constant flow of electricity you get from batteries. Your cell phone also works on DC power, so your charging cable includes a bulky box that converts the AC power from a wall outlet into DC form the phone can use.

Figure 6-5 shows all the members of the DC branch of the motor family tree: DC motor, DC gearhead, hobby servo, and stepper motor.

FIGURE 6-5 DC motor family, clockwise from top left: DC motor, DC gearhead, hobby servo, and stepper motor (images used with permission from SparkFun Electronics and ServoCity)



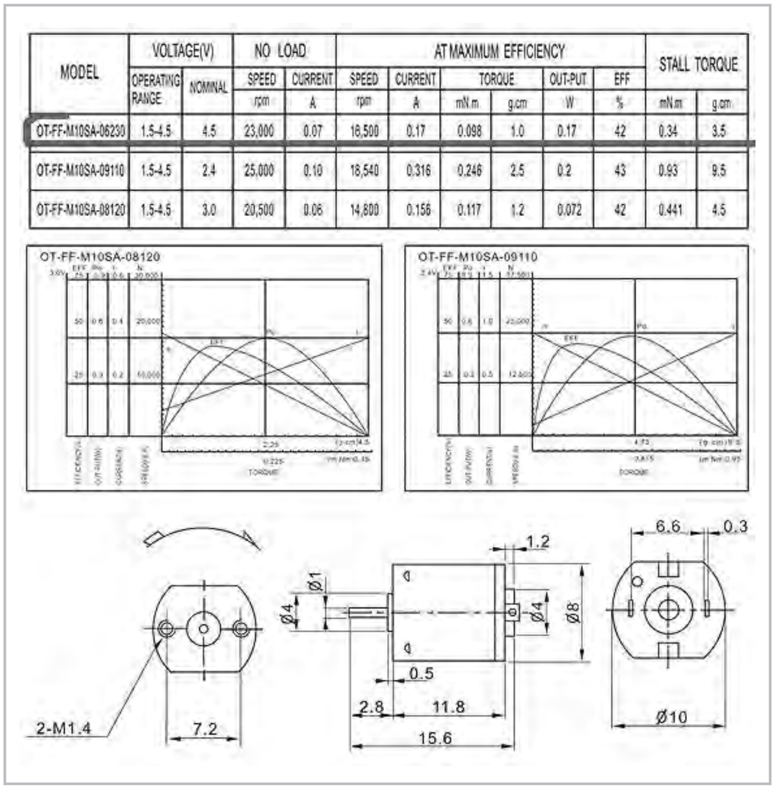
Standard DC Motors

The most basic motor you'll use is the standard DC motor, also called a DC toy motor. You'll find these in everything from toy cars to electric screwdrivers. The insides look like our DIY motor wrapped in a motor housing that resembles a can. Coils of wire are secured to the central shaft, and magnets are attached to the inside of the motor housing. There are also slightly more sophisticated versions of the DIY motor's scraped ends (a commutator) and paperclips (brushes) that enable the field to flip back and forth, as opposed to turning on and off. This makes even small motors more powerful than the DIY version in Project 6-1.

The motor has only two electrical connections, so all you need to do to make a 9V DC motor turn is hook it up to a 9V battery. To reverse the direction, reverse the connections to the battery. If you lower the voltage, it will still work over a certain range, but spin slower. If you raise the voltage, it will spin faster.[2]

DC toy motors usually need between 1.5V and 12V. They spin at speeds anywhere from 1,000 to 20,000 rpm or more. A good example is SparkFun's ROB-09608.

FIGURE 6-6 Data sheet for DC toy motor, ROB-09608 (image used with permission from SparkFun Electronics)



MODEL	VOLTAGE(V)		NO LOAD		AT MAXIMUM EFFICIENCY							STALL TORQUE	
	OPERATING RANGE	NOMINAL	SPEED	CURRENT	SPEED	CURRENT	TORQUE		OUT-PUT	EFF			
			rpm	A	rpm	A	mN.m	g.cm	W	%		mN.m	g.cm
OT-FF-M10SA-06230	1.5-4.5	4.5	23,000	0.07	18,500	0.17	0.098	1.0	0.17	42		0.34	3.5
OT-FF-M10SA-09110	1.5-4.5	2.4	25,000	0.10	18,540	0.316	0.246	2.5	0.2	43		0.93	9.5
OT-FF-M10SA-08120	1.5-4.5	3.0	20,500	0.06	14,800	0.156	0.117	1.2	0.072	42		0.441	4.5

The data sheet for this motor is shown in Figure 6-6 (www.sparkfun.com/datasheets/Robotics/ROB-09608.jpg).

Whoa, that's a lot of numbers! Let's step through this to make sense of what the data sheet is telling us and find the important parts.

- **VOLTAGE** The first column shows that the operating range is 1.5–4.5V, and nominal is 4.5V. This means that the motor will spin if you give it anywhere from 1.5V to 4.5V, but it really likes 4.5V the best. Your standard AA battery is 1.5V, so this motor will work with just one of those, but you could string three 1.5V AA batteries together in series to give the motor the 4.5V it prefers.

- **NO LOAD** The next column is split into speed and current. *No load* means this is what the motor is going to do when there is nothing attached to the shaft. Under the no load condition, this little motor is going to spin at 23,000 rpm! That's fast. And it's going to take only 0.07A to do it.

NOTE ***Sometimes you'll see motor speed in revolutions per second (rps) or radians/second (rad/s). There are 2π radians in one revolution, and 60 seconds in 1 minute. To convert from rad/s to rpm, multiply the rad/s by (60/2π) to get rpm. Or just go to www.onlineconversion.com/frequency so you don't need to remember the conversion.***

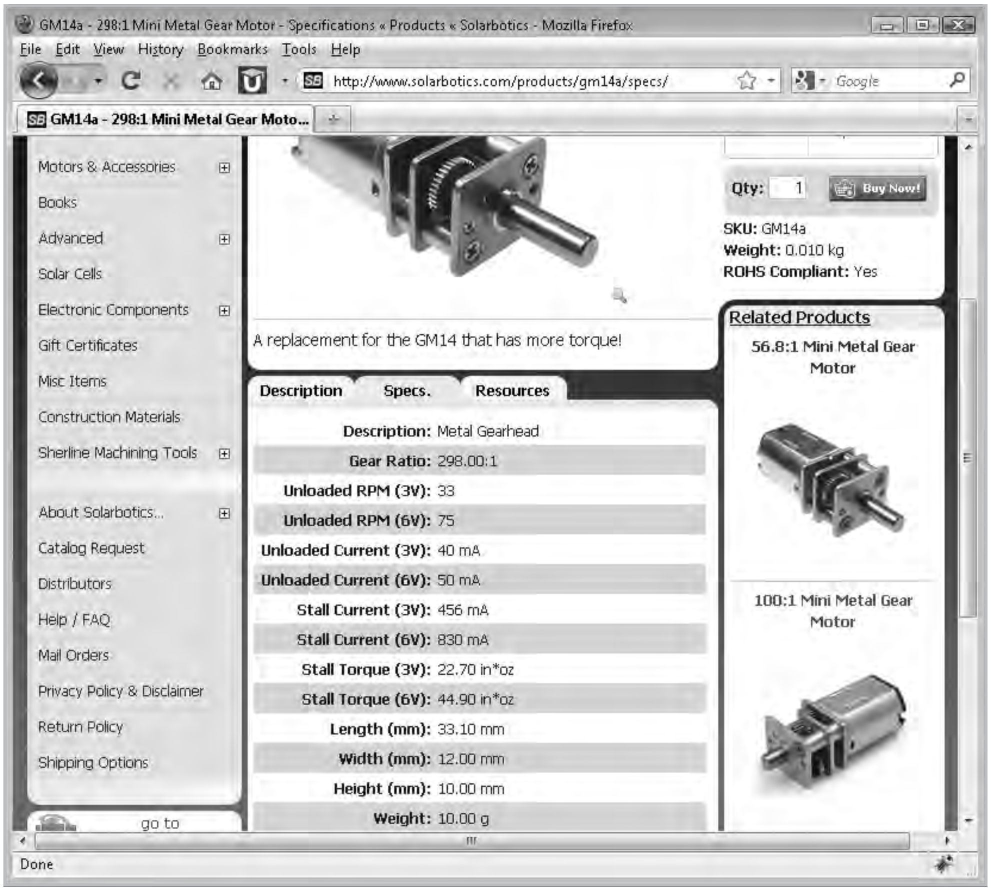
- **STALL TORQUE** Let's skip to the last column. This tells us that the motor will stall, or stop moving, when resisted with 0.34 millinewton-meters (mNm) of torque. Think of this as the maximum strength of the motor. This measurement of torque is in the familiar *force* × *distance* units, but if you can relate better to imperial units, go to www.onlinecoversion.com/torque to change it to something else. It turns out that 0.34 mNm equals about 0.05 oz-in. This is very weak, so this tiny motor could barely spin a 0.05 oz weight at the end of a 1 in stick glued to the motor shaft. You can feel how little torque this is by pinching the shaft with your fingers. It stops almost immediately. You can always stall DC toy motors with your fingers since the stall torque is so low.
- **AT MAXIMUM EFFICIENCY** This column contains a lot of numbers that are useful to review. Efficiency describes the relationship of mechanical power delivered to electrical power consumed. DC motors are most efficient at a fraction of the stall torque (in this particular case, maximum efficiency is around one-fourth of the stall torque). This torque corresponds with the EFF label on the bump on the graph of torque versus current in the middle of Figure 6-6. The motor uses power most efficiently at this torque. You can use the motor at a torque closer to its full stall torque, but it will be slower, and less of the electrical power will be converted to mechanical motion, which is particularly draining if you're running on batteries.

DC Gearhead Motors

The next step in motor complexity is the DC gearhead motor. This is just a standard DC motor with a *gearhead* on it. A gearhead is just a box of gears that takes the output shaft of the standard DC motor and "gears it up" to a second output shaft that has higher torque, but turns slower. How much slower depends on the gear ratio. This should sound familiar from earlier chapters that talked about mechanical advantage. Here, we're trading speed for torque: a 100:1 gearhead ratio will give us 100 times more torque than without the gears, but also will be 100 times slower. DC gearhead motors usually range from about 3V to 30V and run at speeds from less than 1 rpm to a few hundred rpm.

The GM14a from Solarbotics is an example of a tiny gearhead motor. You can even see the little gears. The data sheet, found under the Specs. tab on the Solarbotics website (www.solarbotics.com/products/gm14a/specs/), is shown in Figure 6-7. As shown here, on most DC gearhead motors, the gearhead is the end the shaft extends

FIGURE 6-7 Specs for DC gearhead motor GM14a from Solarbotics



SKU: GM14a
Weight: 0.010 kg
ROHS Compliant: Yes

A replacement for the GM14 that has more torque!

Description:	Metal Gearhead
Gear Ratio:	298.00:1
Unloaded RPM (3V):	33
Unloaded RPM (6V):	75
Unloaded Current (3V):	40 mA
Unloaded Current (6V):	50 mA
Stall Current (3V):	456 mA
Stall Current (6V):	830 mA
Stall Torque (3V):	22.70 in*oz
Stall Torque (6V):	44.90 in*oz
Length (mm):	33.10 mm
Width (mm):	12.00 mm
Height (mm):	10.00 mm
Weight:	10.00 g

from (usually centered, but not always), and the motor is the end where the power is connected.

All motor data sheets look different, and the terminology can vary, but don't let that scare you. Let's look down the list in Figure 6-7.

- **Gear Ratio** This doesn't tell us anything yet, because even though we know it has 298 times more torque than the tiny motor did without the gearhead, we don't know anything about the tiny motor.
- **Unloaded RPM** This is the same as the no load speed in Figure 6-6. The speed here at 3V is only 33 rpm—*much* slower than the 23,000 rpm of the DC toy motor! The next line shows the unloaded RPM at 6V. The two values indicate that the motor will run on anything between 3V to 6V just fine, so the specs give you the speed for each extreme.
- **Unloaded Current** This is the same as the no load current spec in Figure 6-6. Milliamps are used here instead of amps. A rating of 40mA is 0.040A, which is even less than the 0.070A required by the DC toy motor.
- **Stall Current** This is the current the motor needs at the stall torque.
- **Stall Torque** At 6V, the stall torque here is 44.90 in-oz, which is about *900 tim*es more torque than the DC toy motor! I told you DC gearheads are stronger.

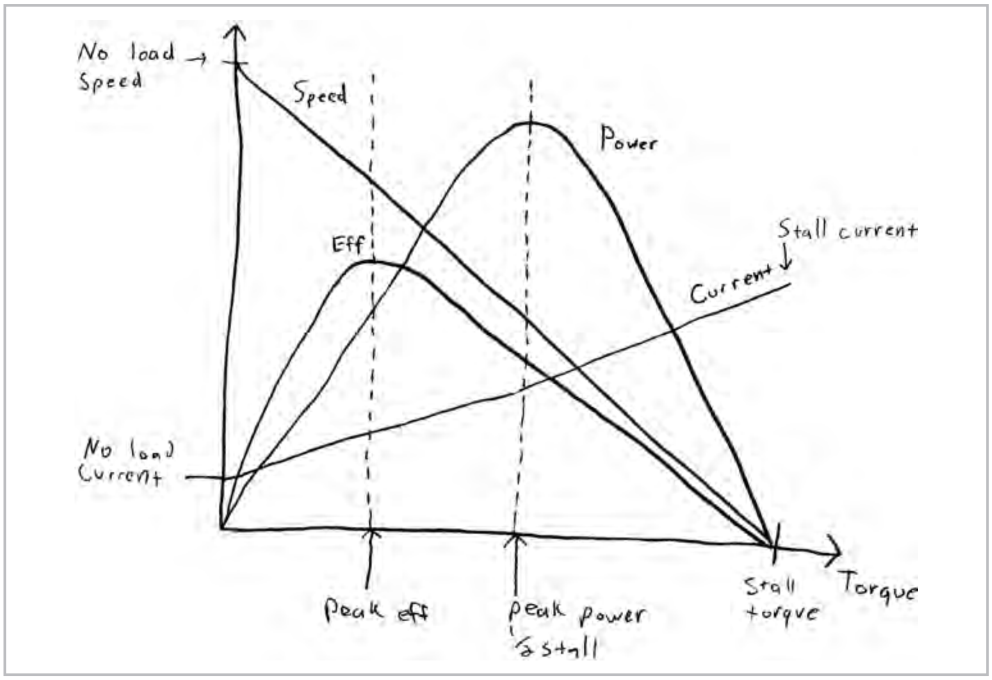
All DC motors have similar relationships among speed, power, efficiency, current, and torque. You've learned that maximum *efficiency* happens at about one-fourth of stall torque. As you can see in the data sheet for the DC toy motor and Figure 6-8, maximum *power* happens at one-half the stall torque.

Standard Hobby Servo Motors

There are two types of hobby servo motors: standard and continuous rotation. Standard servos are by far the more popular. They are usually found in radio-controlled models like planes and boats.

NOTE ***In industry terminology,* servo *refers to any motor with built-in feedback of some sort.* Feedback *just means there is some way to know where the output shaft is. I'll call the ones covered in this book* hobby servos *to distinguish them from industrial servo motors.***

FIGURE 6-8 Relationships among speed, power, efficiency, current, and torque in DC motors



Standard hobby servo motors are just little DC gearhead motors with some smarts in them, as shown in Figure 6-9. When you give the smarts a certain kind of *pulse*—basically just turning the power on and off in a specific pattern—you're actually telling the servo motor where to point the shaft.

Instead of having just two wires you attach to a power source like the DC motors described earlier, these motors have three wires and are controlled by pulses. Standard servos have ranges between 60° and 270° (typically 180°), so they are most useful for pointing and positioning tasks. They also typically use 4.8V to 6V.

FIGURE 6-9 Anatomy of a hobby servo motor (image used with permission from ServoCity)

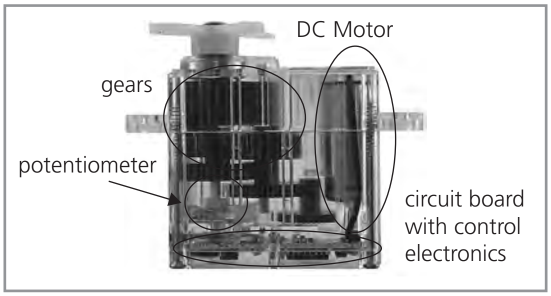
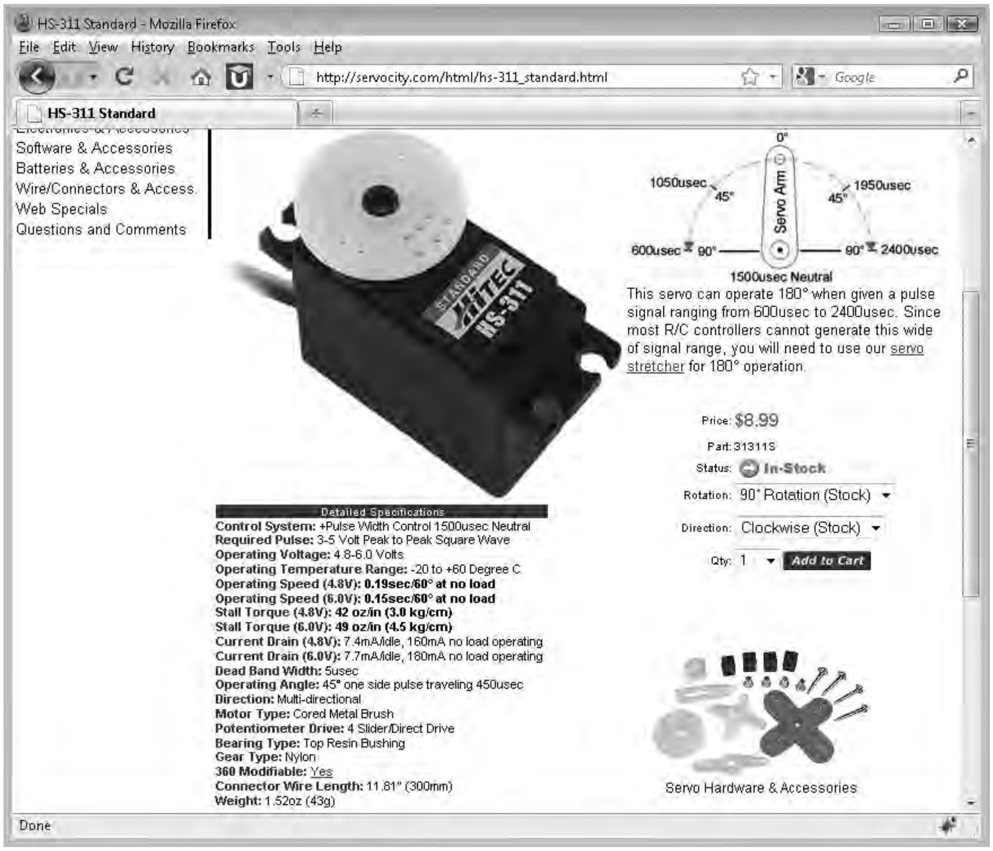


FIGURE 6-10 Detailed specifications for the Hitec HS-311 standard hobby servo motor



This servo can operate 180° when given a pulse signal ranging from 600usec to 2400usec. Since most R/C controllers cannot generate this wide of signal range, you will need to use our servo stretcher for 180° operation.

Detailed Specifications

Control System: +Pulse Width Control 1500usec Neutral
Required Pulse: 3-5 Volt Peak to Peak Square Wave
Operating Voltage: 4.8-6.0 Volts
Operating Temperature Range: -20 to +60 Degree C
Operating Speed (4.8V): 0.19sec/60° at no load
Operating Speed (6.0V): 0.15sec/60° at no load
Stall Torque (4.8V): 42 oz/in (3.0 kg/cm)
Stall Torque (6.0V): 49 oz/in (4.5 kg/cm)
Current Drain (4.8V): 7.4mA/idle, 160mA no load operating
Current Drain (6.0V): 7.7mA/idle, 180mA no load operating
Dead Band Width: 5usec
Operating Angle: 45° one side pulse traveling 450usec
Direction: Multi-directional
Motor Type: Cored Metal Brush
Potentiometer Drive: 4 Slider/Direct Drive
Bearing Type: Top Resin Bushing
Gear Type: Nylon
360 Modifiable: Yes
Connector Wire Length: 11.81" (300mm)
Weight: 1.52oz (43g)

Hitec and Futaba are two popular brands of hobby servo motors that you can find at ServoCity and other sources. Figure 6-10 shows the detailed specifications for a Hitec HS-311.

Let's go through the items of interest in the specifications list, skipping the Control System and Required Pulse lines for now (we'll return to them when we talk about motor control later in the chapter.)

- **Operating Voltage** This means the motor will work if you give it anywhere from 4.8V to 6V.
- **Operating Temperature Range** This indicates the environment in which you can safely use the motor. The limits here are set by the sensitive electronic components on the printed circuit board inside the motor housing.

- **Operating Speed** This is equivalent to the no load speed on the DC motors, but worded a little differently. Because servos don't rotate all the way around, you won't see rpm. These specs tell us that when given 6V, the motor will move 60° (or one-sixth of a full rotation) in 0.15 second with no load on the shaft.
- **Current Drain** This is similar to no load current on the DC motors. For this servo, we see two numbers. At 6V, the servo will draw 7.7mA of current just doing nothing, and 180mA when moving with no load on the shaft.
- **Stall Torque** This is the same as on the DC toy and gearhead motors. The stall torque is the highest torque the motor can give, and happens when you stop it or stall it when it's trying to move. This motor shows 49 oz-in of torque at 6V when stalled. This is equivalent to about 3 in-lbs.
- **Operating Angle and Direction** These tell us the servo can move 45° in either direction, for a total range of 90°. On this particular motor model, you can pay $10 more to get a 180° range.
- **Gear Type** This is the most relevant line for us in the rest of the lines in the specification, which describe the motor parts. Lower torque servos will have plastic gears usually made out of nylon. Stronger servos with higher torque use metal gears.

Continuous Rotation Hobby Servos

A continuous rotation servo is a modification of the standard servo motor. Instead of determining position, the pulses tell the motor how fast to go. You give up knowing the position of the servo arm here, but you gain speed control and 360° movement. A continuous rotation servo is a great option if you have something that needs to spin continuously but you want an easy way to control the speed, such as for an electronic toy mouse to chase your cat around.

You can either buy servos that are already modified for continuous rotation, like the Hitec HSR-1425CR, or get a standard hobby servo and perform some surgery to modify it yourself. If you're wondering which servos can be modified for continuous rotation, check ServoCity's Rotation Modification Difficulty List (www.servocity.com/html/rotation_modification_difficul.html).

Stepper Motors

The stepper motor combines the precise positioning of standard hobby servos and the continuous rotation of DC toy and gearhead motors. The central shaft of a stepper has a series of magnets on it in the shape of a gear, and there are several wire coils surrounding this gear magnet on the inside of the motor housing. It is a bit like an inside-out version of the previously described DC motors, which have the coils on the shaft and the magnets on the housing.

Steppers work by moving in a bunch of little increments, or steps. If you step them fast enough, it looks like continuous motion. Each time one of the coils is energized, it pulls one of the teeth on the shaft toward it to complete one step. For example, a 200-step motor moves in a full 360° circle at 1.8° per step.

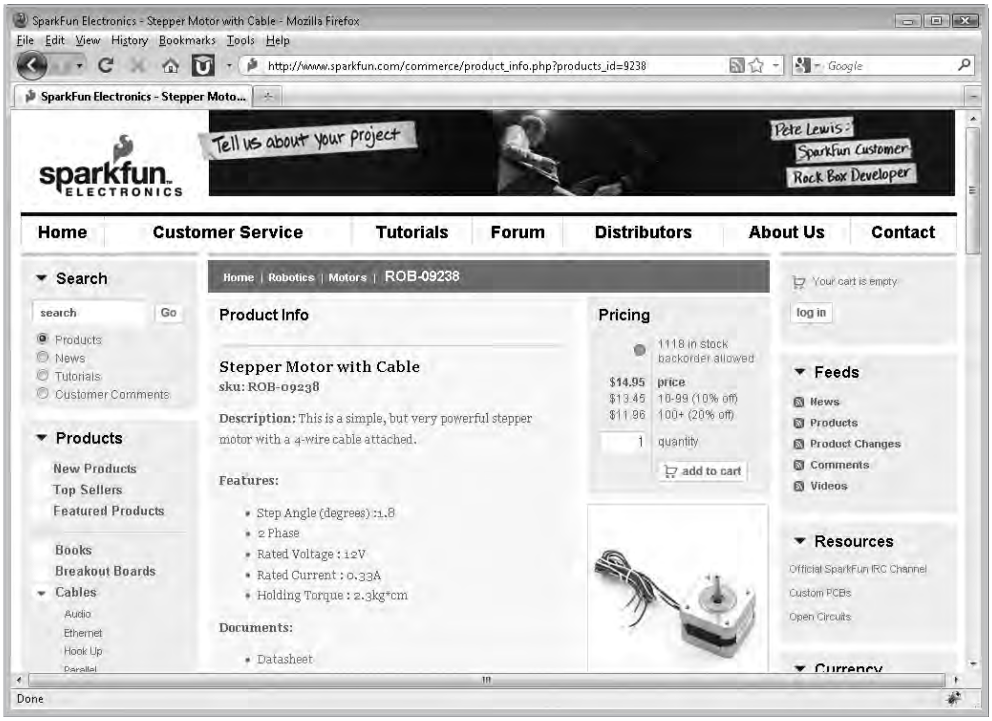
These motors have four to eight wires you need to use to control the pulses to make the shaft step continuously, so they're more complicated to control than the previously described motors. They are squatter looking than the rest of the DC motor family, and have less torque than you might expect for their size and weight. However, they're also the fastest way to integrate both speed and position control into a project. Printers and scanners use stepper motors to control the speed and location of the print head with the ink and rotate the paper through them. So if you see a discarded printer on the curb on garbage day, you just found yourself at least two free stepper motors.

A good example of a simple stepper motor is SparkFun's ROB-09238. Figure 6-11 shows the feature list from the website (www.sparkfun.com/commerce/product_info.php?products_id=9238).

Let's step through the list to see what we have here.

- **Step Angle** This is in degrees of 1.8. If you divide 360 by 1.8, you get 200 steps for one revolution. We'll talk about how to create these steps in the "Motor Control" section later in this chapter.
- **2 Phase** This stepper is bipolar (4 phase is unipolar). We'll also look at this characteristic in the "Motor Control" section.
- **Rated Voltage** This is 12V, which is just the voltage for which the stepper was designed. Give it more, and you're likely to burn out the motor. Give it less, and it might not turn at all.

FIGURE 6-11 Features of SparkFun's ROB-09238 stepper motor


SparkFun Electronics - Stepper Motor with Cable - Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://www.sparkfun.com/commerce/product_info.php?products_id=9238
Google
SparkFun Electronics - Stepper Moto...
sparkfun ELECTRONICS
Tell us about your project
Pete Lewis: SparkFun Customer Rock Box Developer
Home Customer Service Tutorials Forum Distributors About Us Contact
Search
search Go
Products News Tutorials Customer Comments
Products
New Products
Top Sellers
Featured Products
Books
Breakout Boards
Cables
Audio
Ethernet
Hook Up
Parallel
Home | Robotics | Motors | ROB-09238
Your cart is empty
log in
Feeds
News Products Product Changes Comments Videos
Resources
Official SparkFun IRC Channel
Custom PCBs
Open Circuits
Currency
add to cart
Done

Product Info

Stepper Motor with Cable
sku: ROB-09238

Description: This is a simple, but very powerful stepper motor with a 4-wire cable attached.

Features:

- Step Angle (degrees) :1.8
- 2 Phase
- Rated Voltage : 12V
- Rated Current : 0.33A
- Holding Torque : 2.3kg*cm

Documents:

- Datasheet

Pricing

1118 in stock
backorder allowed

$14.95	price
$13.45	10-99 (10% off)
$11.96	100+ (20% off)
1	quantity

- **Rated Current** Shown as 0.33A, this is the current the winding will draw at the rated torque. It's a good idea to size your power supply for a maximum current higher than this limit, so your motor will hit its own current limit before hitting the limit of its power supply.
- **Holding Torque** This is similar to the stall torque in the motors described previously. The difference is that stepper motors effectively stall every step because there is a series of wire coils around the motor that are activated in a sequence. This motor has 2.3 kg-cm (about 2 in-lbs) of holding torque, which is the torque of the motor when it's powered and *holding* its position at one of the steps. If you try to drive something that needs more torque than the motor is rated for, it will probably slip and you'll lose the benefit of precise positioning. Stepper motors don't inherently know anything about their position. They just know to rotate the number of steps you tell them to rotate. If you exceed the holding torque and the motor slips, all bets are off.

Some other features of stepper motors you might see are *detent torque* and *dynamic torque*. Detent torque is the torque of the motor when it's *moving* from step to step, which is lower than the holding torque, since the shaft is between two holding positions. Dynamic torque is kind of an average of detent and holding torque, and is approximately 65% of the holding torque.[3] As a rule of thumb, don't expect a stepper motor to give you more than 65% of the rated holding torque while it's pulling or pushing something.

The information included on this feature list is enough to choose the motor, but once you have it in your hands, you'll need to know some more details before you can use it—like how to mount it and which wire is which. Luckily, on SparkFun's web page for the motor, there is a link to the data sheet shown in Figure 6-12.

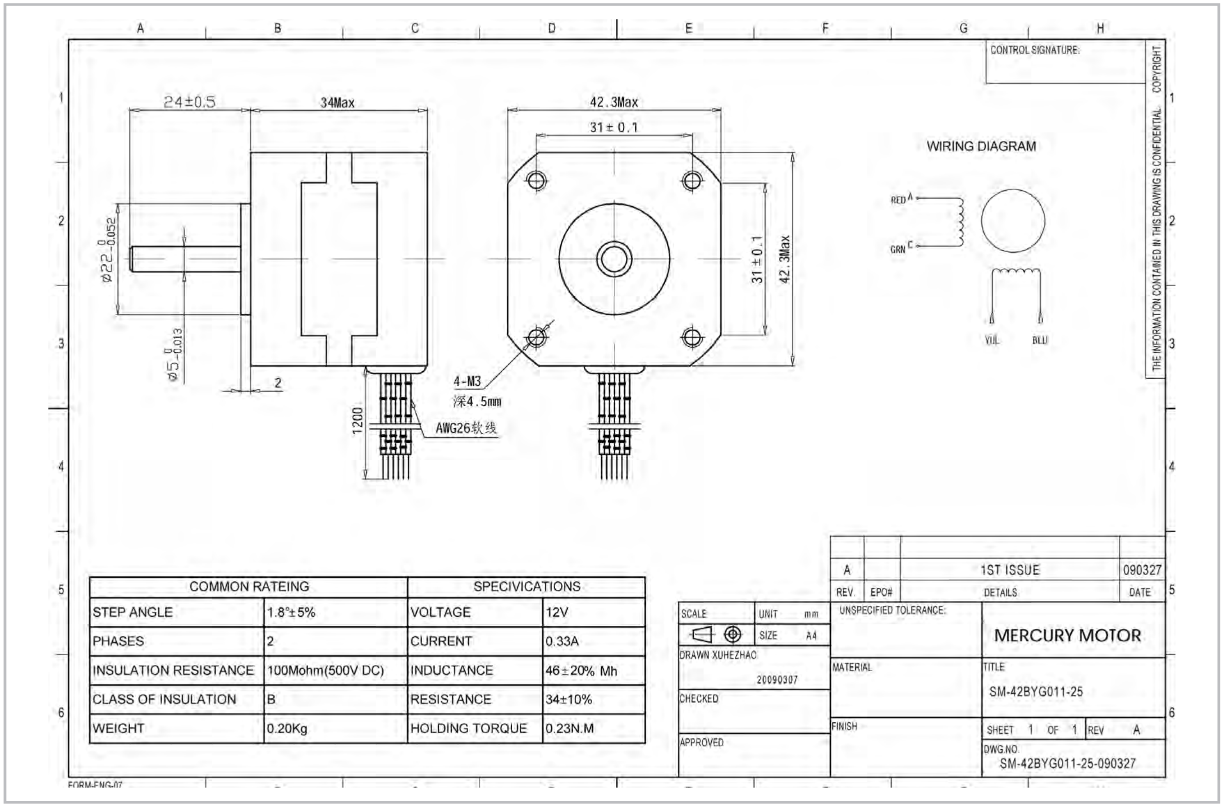
Moving from left to right on the diagram, one of the first numbers you see is the *diameter of the shaft*. Diameter is denoted with the ø symbol, and in this case is 5mm. You also see two small numbers to the right of this, which represent the tolerance of the shaft. They indicate the range of actual dimensions for the 5mm shaft. The small *0* on top indicates the largest shaft size is 5mm + 0 = 5mm, and the bottom number –.013 means that the shaft can be as small as 5 – .013 = 4.987mm. This will be important in the next chapter when we talk about attaching things to motor shafts.

Farther along to the right, you see there are four M3 tapped holes that go 4.5mm deep. M3 is a standard metric screw, and you'll need four of them to mount this motor. There must be no more than 4.5mm of the end of the screws sticking into the motor, or they won't fit.

The wiring diagram on the right shows that the red and green wires control one phase of the motor, and the yellow and blue ones control the other. This will be important when we get to the "Motor Control" section. You've already learned about the important columns in the table in Figure 6-12. The additional information is nice to have, but doesn't matter to us, so feel free to ignore those values, and definitely don't let them confuse you.

AC Motors

You'll find AC motors in many household appliances, like blenders and fans, because they are continuously rotating and use the AC from the wall to drive them. They can be useful if you have a stationary project and just need a plug-and-play motor that



COMMON RATEING		SPECIVICATIONS	
STEP ANGLE	1.8°±5%	VOLTAGE	12V
PHASES	2	CURRENT	0.33A
INSULATION RESISTANCE	100Mohm(500V DC)	INDUCTANCE	46±20% Mh
CLASS OF INSULATION	B	RESISTANCE	34±10%
WEIGHT	0.20Kg	HOLDING TORQUE	0.23N.M

REV.	EPO#	DETAILS	DATE
A		1ST ISSUE	090327

SCALE | UNIT mm | SIZE A4
DRAWN XUHEZHAO 20090307
CHECKED
APPROVED
UNSPECIFIED TOLERANCE:
MATERIAL
FINISH
MERCURY MOTOR
TITLE SM-42BYG011-25
SHEET 1 OF 1 REV A
DWG NO. SM-42BYG011-25-090327

FIGURE 6-12 Data sheet for SparkFun ROB-09238 stepper motor (image used with permission from SparkFun Electronics)

turns all the time and is pretty powerful. However, attempting to control AC motors can be dangerous. You're playing with 120V from the wall, which is much higher than the voltages needed by the DC motors.

An AC motor can draw as much current as it wants from the wall supply, up to about 15A before it trips a breaker in your house. The combination of high voltage and high current is enough to seriously hurt you if something goes wrong. In addition, AC motors near logic circuits are likely to drive those circuits crazy (see the "Helpful Tips and Tricks for Motor Control" section later in this chapter).

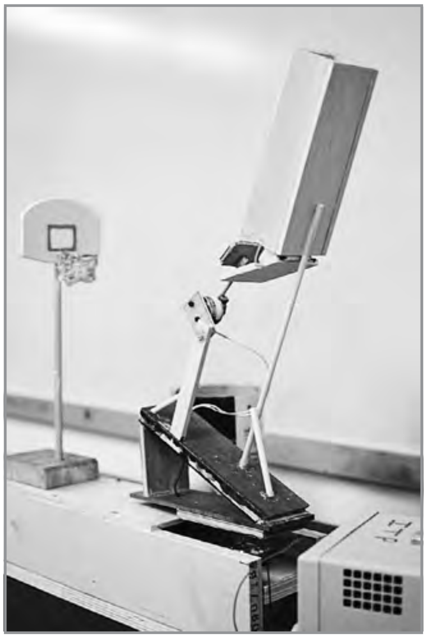
I don't recommend using AC motors for general mechanism projects. However, if you can use them without modification or control, they can be helpful. SparkFun carries a PowerSwitch Tail (COM-09842), which isolates the lethal AC power but still allows you to control whatever plugs into it. If you want to do more with AC motor control, and have the time to study up on AC motors, you are encouraged to seek out other sources of information so you can work safely and effectively.

Rotary Solenoids

Rotary solenoids are good for quick rotary movements through a short range of motion. They are really just modified linear solenoids (see the upcoming "Solenoids" section) that force the plunger into a guide that makes it rotate.

Rotary solenoids are pretty expensive for their limited application, but are ideal for throwing ping-pong balls at mini basketball hoops (see Figure 6-13). Ledex (www.ledex.com) manufactures a wide selection of these.

FIGURE 6-13 Using a rotary solenoid to launch ping-pong balls (CC-BY-NC-SA image used with permission from Greg Borenstein)



Types of Linear Actuators

Linear motors are far less common than rotary motors. There are quite a few other ways to create linear output from rotary input (see Chapter 7 for more on this). However, linear motors can be handy when you have a specific need. Figure 6-14 shows the two main types of linear actuators: linear motors and solenoids.

Linear Motors

Linear motors, like the ones from ServoCity shown in Figure 6-14, are DC gearhead motors that interact with an Acme or ball screw assembly to push a plunger in and out. We'll talk more about these kinds of screws in the next chapter.

Linear motors can do a great deal of work, but you will pay the price for the convenient packaging (they start at around $130). A former student of mine used them to create lifting shoe mechanisms strong enough to hold and lift her weight (see Figure 6-15).

On data sheets for these motors, you'll see a lot of terms that should look familiar by now: operating voltage, no load current, and so on. Since the action is linear, you'll see speed in inches per second instead of rpm. You should also see ratings for static load and dynamic thrust. *Static load* is the weight of something you can put on the

FIGURE 6-14 Linear motors (left) (image used with permission from ServoCity) and solenoid (right)

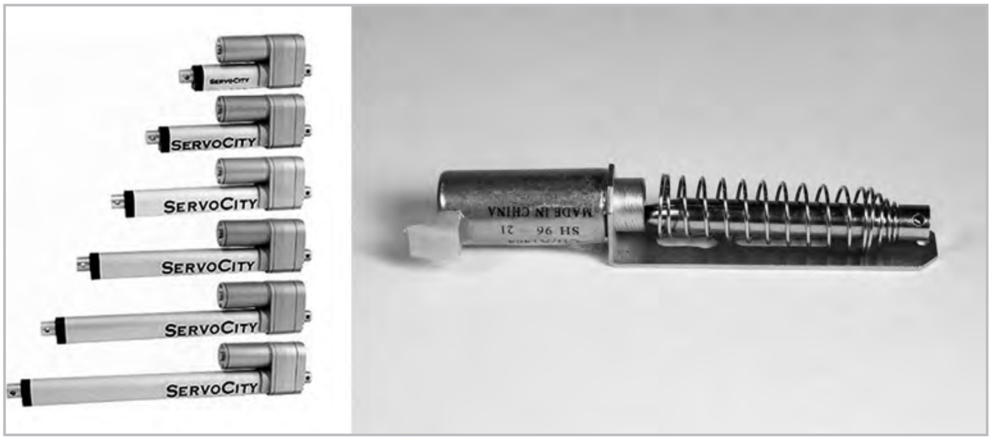
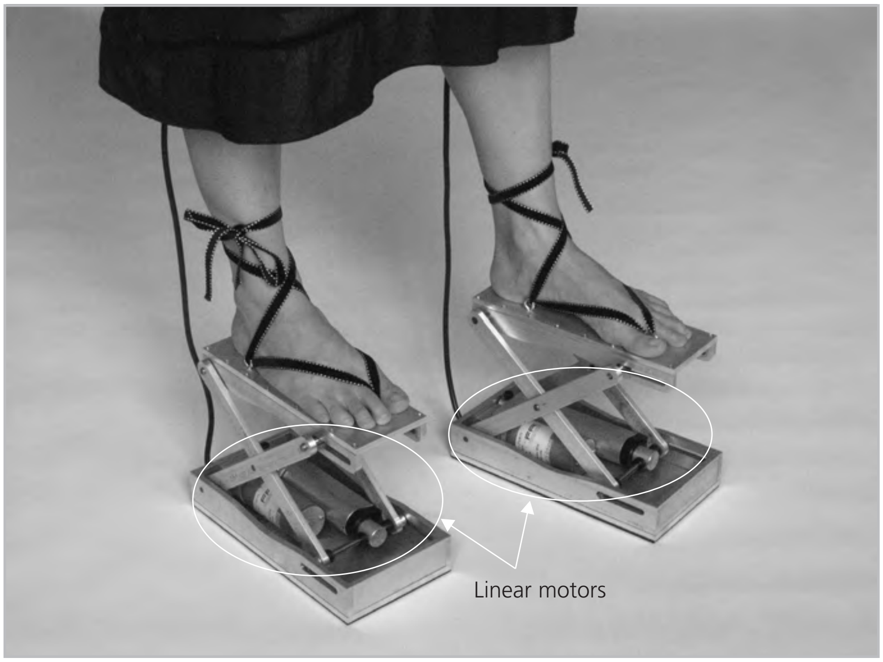


FIGURE 6-15 Linear motor controlling shoe lift (credit: Adi Marom)



plunger and expect it to hold. *Dynamic thrust* is the maximum weight of something you can expect the motor to move. For example, the 25 lb actuator from ServoCity in Figure 6-14 (the smallest one) will not lift you up if you weigh 150 lbs, but it will hold your weight if fixed in one place.

Solenoids

Solenoids work like a motor that translates (moves in or out) instead of spinning. A solenoid consists of a housing, a plunger, and usually a spring that returns the plunger to a resting state once the power is off. There's a coil of wire around the plunger, and when electricity flows through that coil, it either attracts or repels the plunger to give you a short, linear stroke—good for pushing buttons and making robotic instruments.

If you have a doorbell, it most likely has a solenoid in it. When you press the button, it closes a circuit that makes a solenoid turn on, which moves the plunger and hits a chime.

Motor Control

Many times in mechanism projects, you want to do more than just turn a motor on and off. You might need it to spin a certain number of times, point a camera at a certain angle repeatedly, or raise and lower a window shade. You can also make your motor react to certain sensors and switches, like using a photocell to help lower your window shade automatically when it gets too bright. In the following sections, we'll talk about how to go from just getting a motor to work to more advanced ways to control them. There are whole books written on motor control, so this section is not exhaustive, but it will get you (and your motor) moving. I'll point out additional sources as we go along.

In the spirit of rapid prototyping, we'll try to minimize soldering and maximize our use of breadboards and ready-made modules to talk with our motors. It can be time-consuming and takes special equipment, but it is a handy skill to have, so we'll go through a quick example.

Solderless breadboards are much easier to use for quick prototyping. A *breadboard* is a way of connecting wires and other components together to make circuits quickly. Once you go through the breadboard example, we'll kick it up a notch and use the Arduino prototyping platform—a kind of mini-computer—to give our motors more complicated instructions. Finally, we'll integrate an off-the-shelf module and an Arduino to control a stepper motor. If this all sounds like Greek to you, don't worry. We'll go through each project step by step.

Basic DC Motor Control

All you need to do to get a DC toy or DC gearhead motor to spin is hook it up to a power source within its desired voltage range.

NOTE ***The examples here use red, black, and yellow wires for power, ground, and signal. These appear as gray, black, and white in the images.***

Project 6-2: DC Motor Control 101—The Simplest Circuit

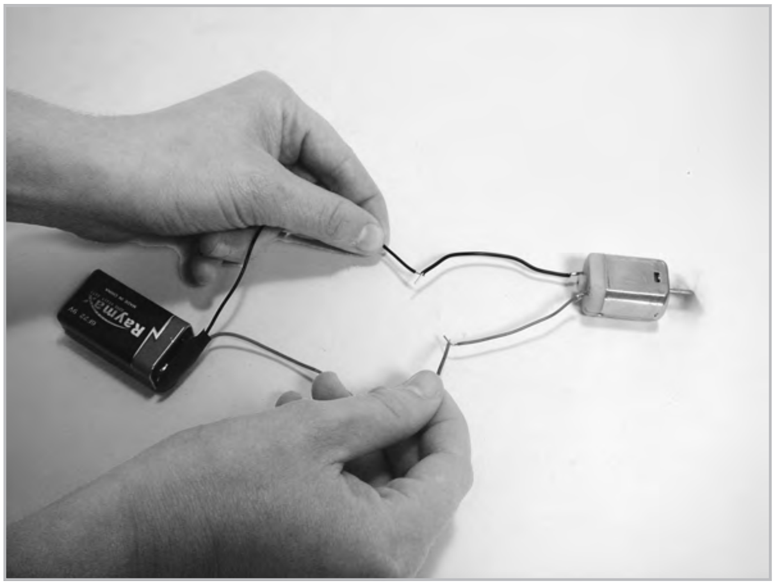
If you hook up a 9V battery to a 9V DC motor, it will spin. Reverse the battery connections, and it will spin the other way. Let's take two components—a battery and a motor—and join them in a simple circuit.

Shopping List:

- DC toy motor
- Corresponding battery (9V used here)
- Battery snap or holder with wire leads (like RadioShack 270-324)

For example, you could use a small DC motor (SparkFun's ROB-09608 already has the wire leads on it) and just one AA battery, because the motor will run off 1.5V. The DC toy motor in Figure 6-16 shows a 9V battery and snap connector, and a 6V motor that will run on 3V to 9V. All Electronics (www.allelectronics.com/) is a great source of battery holders for just about any size.

FIGURE 6-16 A simple circuit with a DC motor and battery



Recipe:

1. Touch the black wire from the motor to the black wire on the battery.
2. Touch the red wire from the motor to the red wire on the battery. Your motor should spin!
3. Reverse the black and red wires. The motor will spin the other way. The motor shown in Figure 6-16 has a small duct tape flag to make movement more obvious.

Project 6-3: Solder a Circuit

Solder is like conductive hot glue that lets you stick metal things together to conduct electricity. For this project, you'll need a soldering iron, which is like a pen with a hot tip that you plug in. A cheap one like RadioShack 64-2802 is fine for the amount of soldering we'll do in this book. This kit comes with a small stand and some solder, so you'll have three of the items you need for this project. If you plan to spend much time with electronics, you may want to spring for a nicer model with interchangeable tips and a temperature control dial like Jameco Electronics (www.jameco.com/) part 46595. You'll also need some solder. Lead-free solder is the standard in Europe. It's a little harder to use for beginners, but better for you and the planet.

You'll also need a single pole, single throw (SPST) on/off toggle switch. A SPST switch will have two legs and will toggle between on and off. When the switch is on, two metal pieces inside the switch touch, like when you touched the wires together in Project 6-2. When the switch is off, those metal pieces are pushed apart so no power can flow through the switch.

Shopping List:

- DC toy motor
- Corresponding battery (9V used here)
- Snap or holder with wire leads (like RadioShack 270-324)
- On/off toggle switch (like SparkFun COM-09276) or other SPST switch

- Soldering iron
- Solder (SparkFun TOL-09162)
- Stand, preferably with a sponge to wipe the tip on (like SparkFun TOL-09477)
- Helping hands (like SparkFun TOL-09317)
- Multitool with pliers or other pliers
- Rubbing alcohol (optional)
- Stiff brush (optional)

Recipe:

1. Plug your soldering iron in and set it on a stand.

CAUTION ***The soldering iron will get extremely hot!***

2. Clean the motor wires and switch terminals or wires with rubbing alcohol and a stiff brush. Although not strictly necessary, this step will make soldering a lot easier and make the connection better.
3. Loop the bare metal end of the red wire from your motor into one of the legs of the switch. Gently squish it with the pliers so it doesn't jiggle around too much. Turn the switch to the off position.
4. Position the switch in one of the clips of the helping hands so you don't need to hold it.
5. Unroll a little solder and touch it to the soldering iron tip. If the soldering iron is at the right temperature, the solder will melt instantly and stick to the tip. This is called *tinning* the tip, and makes your job easier.
6. Position your soldering iron on one side of the motor wire/switch connection. After a few seconds, the wire from the motor and the switch leg will heat up.

7. Touch the solder to the other side of the motor wire/switch connection, as shown in Figure 6-17. If you touch the solder directly to the soldering iron, it will melt quickly, but will not usually form a strong joint. The idea is to heat up the stuff you are joining, and let that stuff heat the solder. If you do it correctly, you'll see a shiny blob of solder melt into and around the motor wire/switch connection. Don't worry if it isn't pretty.

8. Do the same thing with the red wire from the battery clip or holder. The battery should *not* be attached yet. Use the pliers and helping hands as needed to hold the wires still while you work. Hold the solder in one hand and the soldering iron in the other hand, and hold anything else with the helping hands or whatever you can to secure the parts in place while you solder (duct tape, clamps, cable ties, and so on).

9. Solder the black wire from the battery pack to the black wire of your motor. You may want to twist the two bare ends together with the pliers first and use the helping hands so the wires stay put while you solder. Your circuit should look like Figure 6-18 (without the battery).

FIGURE 6-17 Soldering the switch

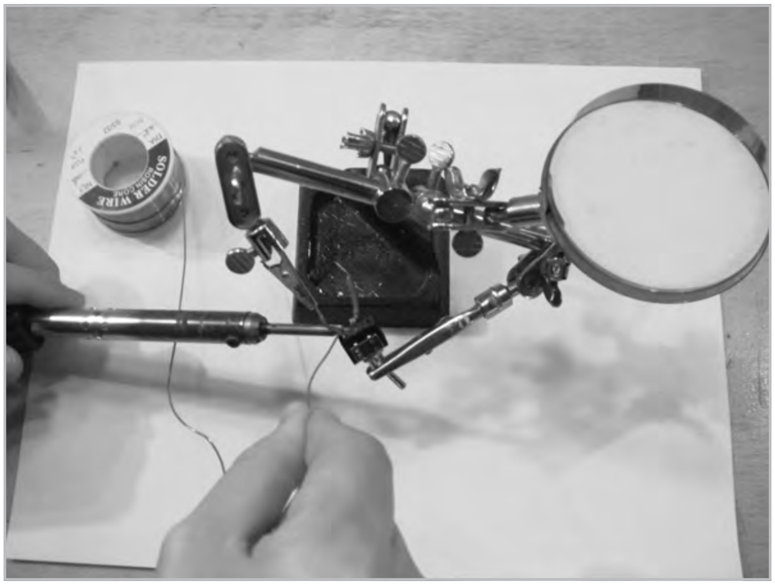
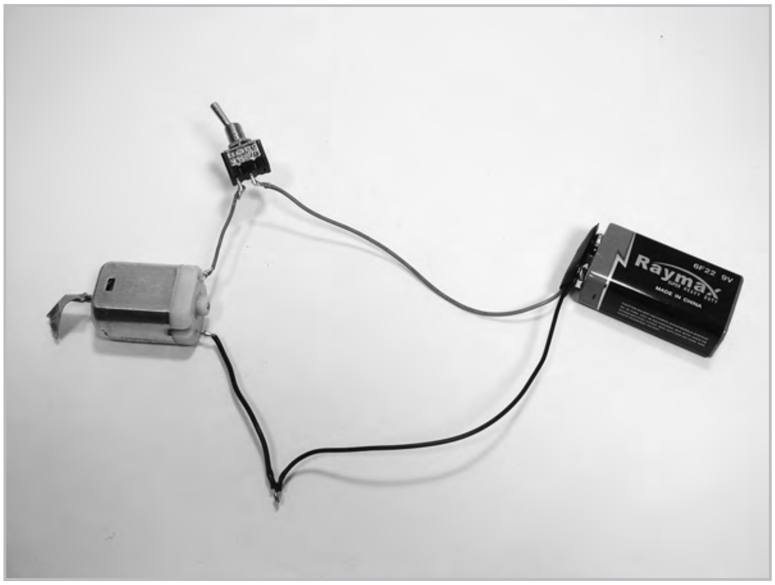


FIGURE 6-18 The complete soldered circuit



10. It's a good idea at this point to relieve any strain on your wires so the soldered joints don't break—a practice commonly called *strain relief*. Ideally, your soldered joints should not be used as mechanical joints. You also don't want to leave conductive parts exposed where you might short them against a wrench on your desk. Use heat-shrink tubing, hot glue, electrical tape, or cable ties wherever necessary.

11. Attach the battery, turn your switch on, and watch the motor move! Turning the switch on allows power from the batteries to flow through your soldered joints to the motor.

Project 6-4: Breadboard a Circuit

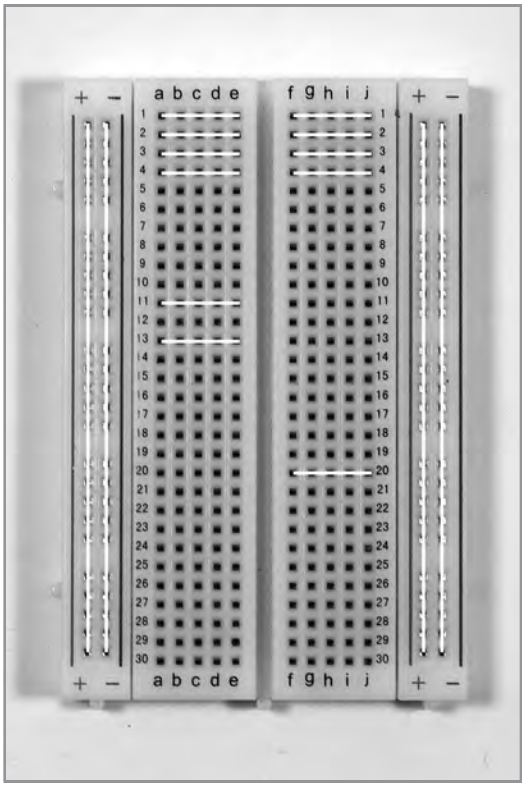
A breadboard is a tool you can use to connect wires together a lot faster than you can with soldering. Since the wires don't get stuck to each other, it's also easier to try different configurations quickly without undoing and redoing the soldered joints. Breadboards are made of plastic with metal links inside that connect the holes you see in the plastic cover. (See www.tigoe.net/pcomp/code/circuits/breadboards for a more complete description of breadboards.)

Figure 6-19 shows a breadboard and indicates which rows and columns are connected underneath the plastic cover. Instead of soldering two wires together to create a connection, you just stick each wire into holes in the same row, and the metal strips underneath that row automatically connect them.

In the following example, we'll create the same circuit as in Project 6-3, but use the breadboard to hold the wires instead of soldering them together. You'll need some jumper wires to create circuits on your breadboard. You can make your own jumper wires by cutting short lengths off red and black insulated solid wire spools (also called *hook-up wire*, like SparkFun PRT-08023 and PRT-08022). All you need is a pair of wire strippers (like SparkFun TOL-08696) to get started.

Wire comes in two flavors: solid and stranded. *Stranded wire* is made up of a bunch of tiny wires as thin as your hair that are twisted together and covered with plastic insulation.

FIGURE 6-19 Breadboard indicating some of the connected rows and columns



Solid wire is just that—a long, solid piece of wire that is covered with plastic insulation. Stranded wire is more flexible, but solid wire is stiffer, so it's easier to plug into breadboards. Wire size is measured in gauges. The wire spools specified above are 22 gauge, which works well in breadboards (the higher the gauge, the thinner the wire).

To strip the plastic insulation off the ends of a piece of wire from your spool, you need to find the groove at the end of your wire strippers that corresponds to the wire gauge number. Place the piece of wire in this groove with about 1/4 in of wire sticking out of one side. Then squeeze the wire strippers together on and off while you rotate the wire. This will cut the insulation but not the wire itself. Once you see a cut all the way around the wire, pull the insulation off with your fingers. Follow the same steps for the other end of your wire piece, and you have your own jumper wire!

Shopping List:

- DC toy motor
- Corresponding battery (9V used here) and snap or holder with wire leads (like RadioShack 270-324)
- On/off toggle switch (like SparkFun COM-09276) or other SPST switch
- Breadboard (like All Electronics PB-400)
- Jumper wires (like SparkFun PRT-00124) or hook-up wire to make your own (as just described)

Recipe:

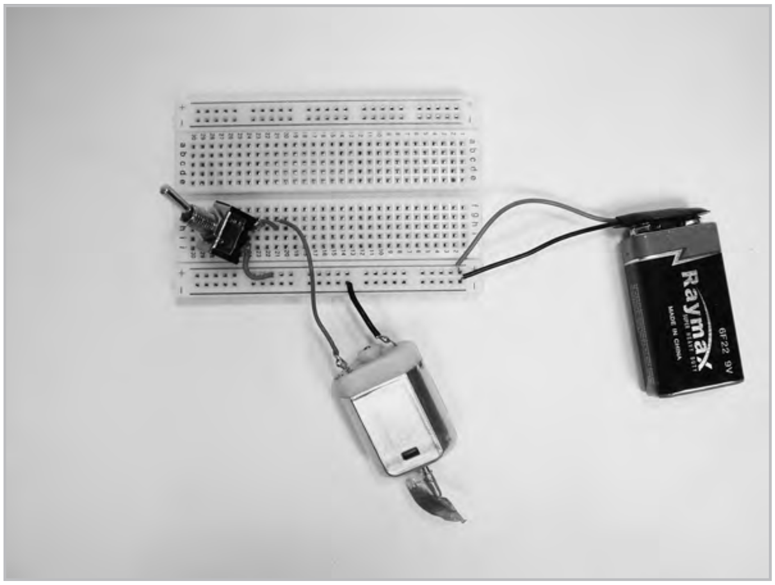
1. Solder jumper wires to the legs of the switch and the terminals of the motor (if they don't already have wire leads). On DC motors, it doesn't matter which terminal is which, so you can just pick one.
2. Plug one leg of the switch into one of the breadboard columns marked with a plus (+) sign and the other to a row of your choice on the breadboard. Turn the switch to the off position.
3. Plug the red wire of your motor into that same row, and the black wire to one of the breadboard columns marked with a minus (–) sign.

4. To get power to your breadboard, plug the red wire of the battery into the breadboard column marked with a + sign. Now plug the black one into the column with a – sign. This makes the whole + column power and the whole – column ground, so it completes your circuit. (Refer to the appendix for other ways to power a breadboard than directly from batteries.) Your circuit should look similar to Figure 6-20.

NOTE ***By convention, with breadboards (and electronics in general), red is power/on/+ and black (sometimes blue or green) is ground/off/–. If you use the marked side columns for power (+) and ground (–), it will be easier to follow your circuits. Red shows up as gray in the black-and-white photos here.***

5. Now flip the switch to on, and your motor should spin!

FIGURE 6-20 The same circuit as in Project 6-3 on a solderless breadboard



Project 6-5: Motor About-Face

In Project 6-2, we changed the direction of the motor by switching the red and black wires manually. It's great that all we need to do to switch the direction of a DC motor is change the direction of current flow, but how do we do that without disconnecting and reconnecting wires all the time?

The simplest way to switch direction is by using an integrated circuit (IC) chip called an H-bridge and a three-legged switch called an SPDT (for single pole, double throw) switch. Instead of just two positions (on and off), an SPDT switch has three positions (on, off, and on). When the switch is at either extreme on position, two metal pieces inside the switch touch. But unlike the SPST switch from the previous example, these metal pieces are independent, so this switch can control separate circuits. When the switch is in the middle (off) position, those metal pieces are pushed apart so no power can flow through the switch.

Inside the H-bridge chip is a series of electronic gates. In this example, when your switch is at one extreme, the gates in the chip will let current flow through the motor in only one direction. When you toggle the switch to the other extreme, the gates in the chip will reverse and make current flow through the motor in the opposite direction.[4]

Shopping List:

- DC toy motor with wire leads
- Corresponding battery (9V used here) and snap or holder with wire leads (like RadioShack 270-324)
- Breadboard (like All Electronics PB-400)
- Jumper wires (like SparkFun PRT-00124) or hook-up wire to make your own (see Project 6-4)
- On-off-on toggle switch (like SparkFun COM-09609) or other SPDT switch
- H-bridge motor driver chip (SparkFun COM-00315)
- Four AA batteries and holder (like SparkFun PRT-00552)

Recipe:

1. Place the H-bridge in the middle of the breadboard with the small notch facing up, as shown in the bottom image of Figure 6-21.

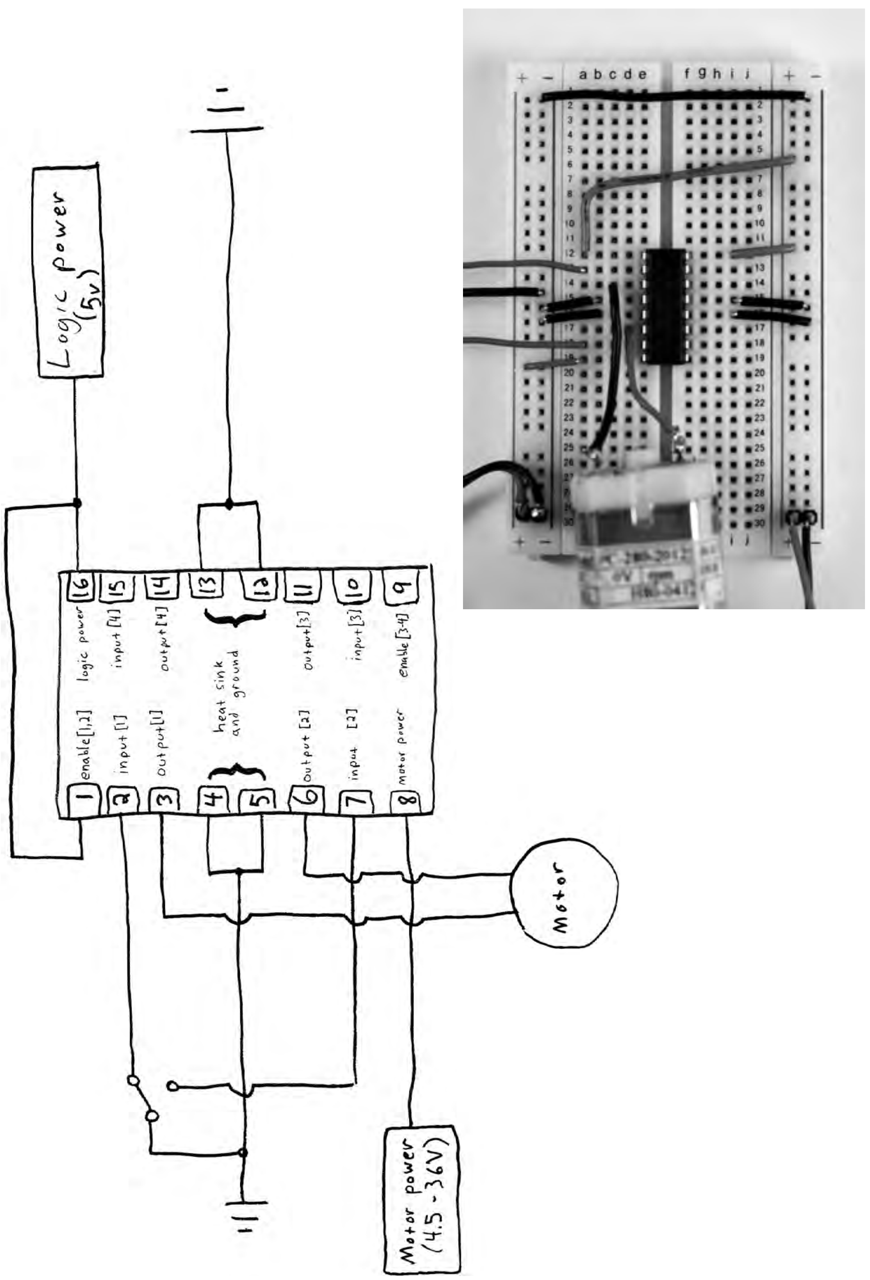


FIGURE 6-21 Identifying pins on the H-bridge (top) and close-up of breadboard with all the wires in place (bottom)

2. Plug the DC toy motor wire leads into the rows next to pins 3 and 6 on the H-bridge. Refer to the top image in Figure 6-21 to see which pin is which.

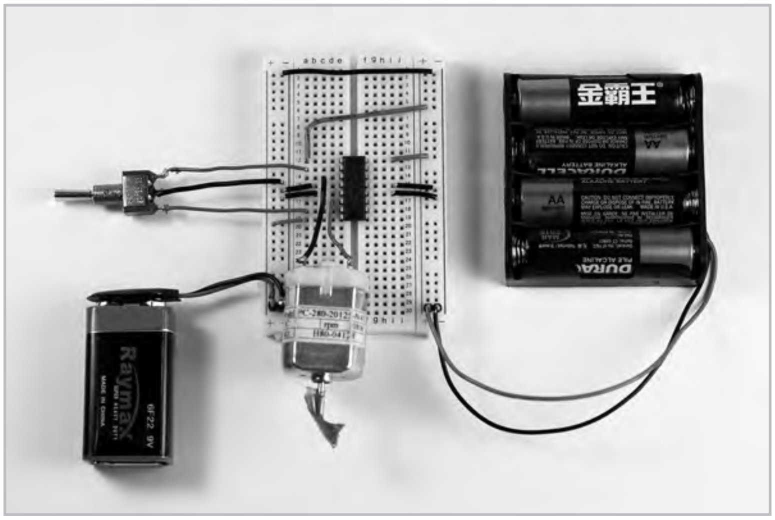
3. Solder jumper wires onto your SPDT switch. The example uses red for each side and black for the center (ground) leg.

4. Plug the two red wires of the SPDT switch into the rows next to pins 2 and 7 on the H-bridge. Connect the black wire from the middle leg to the ground column on the side of the breadboard marked with a – sign (ground). Refer to Figure 6-22 for the full setup. Make sure the switch is in the center (off) position.

5. Use jumper wires to connect pins 4, 5, 12, and 13 of the H-bridge to the ground columns.

6. The H-bridge chip needs about 5V to work. If you use four alkaline AA batteries as shown, that will add up to about 6V, which will work just fine (four rechargeable batteries will equal about 4.8V, which will also work). Plug the black wire from the battery holder into the ground column on the right side of the board and the red wire into the power column on the right side.

FIGURE 6-22 DC motor direction control with an SPDT switch and H-bridge chip



7. Connect pin 1 of the H-bridge to this 5V power column. This is the enable pin, which means it needs to be powered to tell the H-bridge chip you're ready to go. Connect pin 16 to this power column to give the circuit inside the chip power.

8. It's always a good idea to *use separate power supplies for the motor and the control logic parts of circuits*. Chips like the H-bridge we're using always want 5V, but motors usually want something different. Connect your motor battery (a 9V in this example) to the power and ground columns on the left side of the breadboard. This H-bridge chip will allow you to run motors that need up to 1A of current, which should be good for most of the motors we'll talk about in this book.

9. Even though motor power and control power come from different places, the ground columns should still be linked so they share a common zero point. You can do this on your breadboard by using a long jumper to link the two ground columns across the top of the board.

10. Connect pin 8 to the motor power column on the left side of the board.

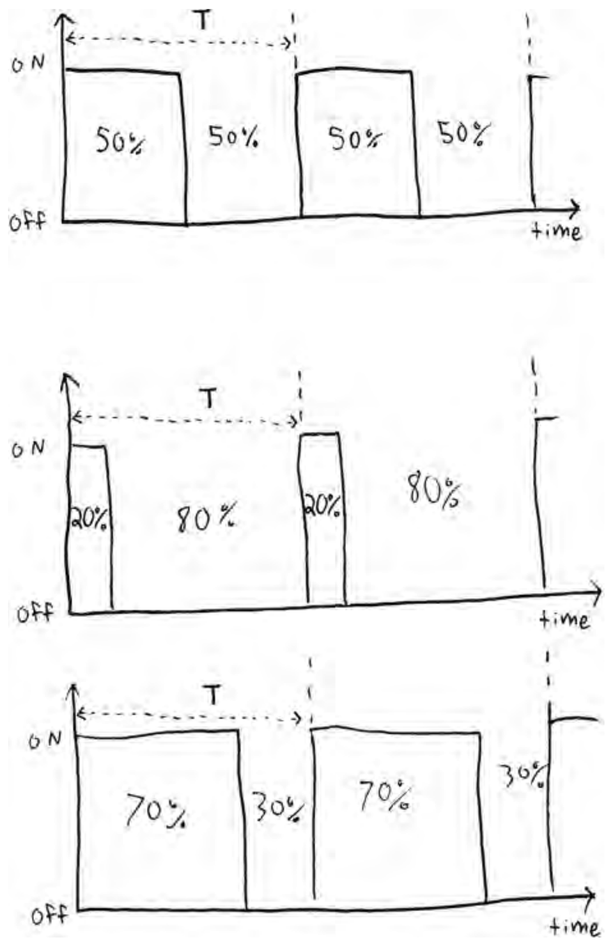
11. Try to flip the switch from on to off to on, and see how the motor spins. It should spin clockwise at one extreme, stop in the middle, and then spin counterclockwise at the other extreme.

At this point, you might be saying, "Whoa, that's a lot of wires. What's actually going on here?" For starters, most IC chips want power and ground like the H-bridge. An H-bridge will allow current to flow through the motor in one direction when given a digital on (high or 5V) signal. We're using a switch in this example to create the on signal. When you flip the switch to the other extreme, another on signal triggers the H-bridge to allow current to flow through the motor in the opposite direction.

Speed Control with Pulse-Width Modulation

By now you know how to turn a motor on and off with a switch, but what if you want to control the speed? Pulse-width modulation (PWM) lets you do this by creating a *duty cycle*—the percentage of on time versus off time—that is between 0% and 100% of a given time period (see Figure 6-23).

FIGURE 6-23 A PWM signal



Think of PWM as flicking a light switch on and off. If you flick the light on and off fast enough, the average of the dark and light makes it look like the light is on, but just dimmer than if you leave it on. The same goes for a PWM signal to control a motor. Instead of giving the motor its full-rated voltage, you flick the power on and off fast enough that the average voltage is below what your power source gives you. For example, if you have a 9V power source trying to drive a DC gearhead motor that wants 3V to 6V, you could give it a pulse width at 50% of the time interval to equal a voltage of 4.5V, and make the motor happy.

Project 6-6: Use Hardware PWM to Control Speed

You can create a PWM signal with hardware—that is, components you can hold—or in software. We'll start with the hardware version by building a circuit around a chip called a 555 timer to create the PWM signal.[5, 6]

You will need a potentiometer and a transistor to complete this circuit. A *potentiometer* is a variable resistor. The two outside legs act as a fixed resistor (like the ones we talked about in Chapter 5). The middle leg is a movable contact called a *wiper*, which moves across the resistor, producing a variable resistance between the center leg and either of the two sides.[7] So our 100KΩ potentiometer will act like two fixed resistors that add up to 100KΩ, and the knob allows us to choose the values of those resistors by moving the wiper. The potentiometer in Figure 6-24 has red, yellow, and black wires soldered to it (which appear gray, white, and black, respectively, in the figure).

We'll use a *transistor* as an electronic switch to connect parts of a circuit, just as the mechanical switches we've used do. As shown in Figure 6-25, the transistor has three legs: base (B), collector (C), and emitter (E). In an NPN type transistor (like this), applying a positive voltage to the base and a negative voltage to the emitter allows

FIGURE 6-24 A potentiometer

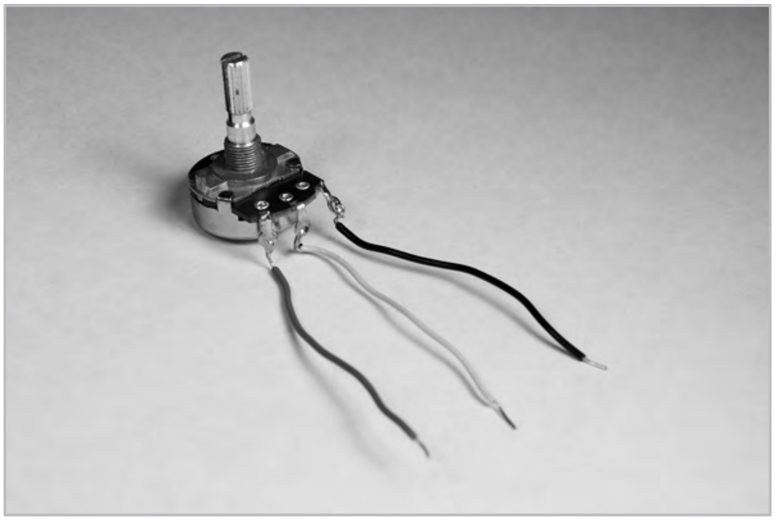
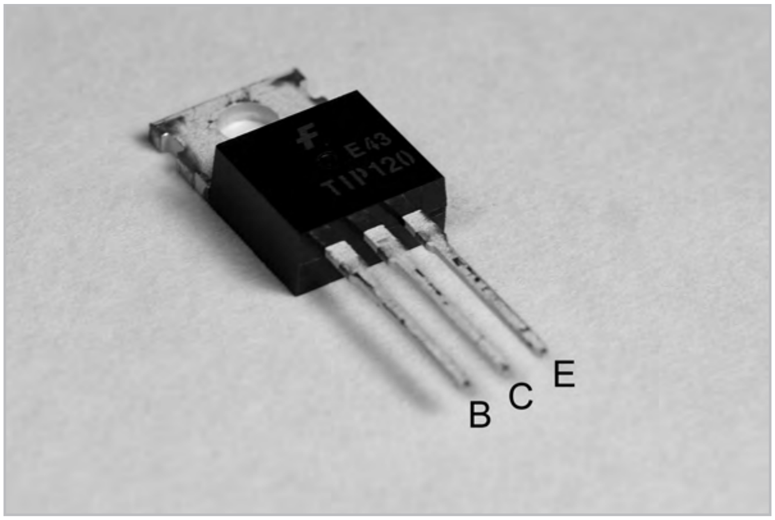


FIGURE 6-25 A transistor



current to flow from collector to emitter. We need a transistor here because even though we can send timing signals to the motor directly from the 555 timer chip, we can't actually send the motor power *through* it. That would fry the chip (the chip can handle only up to 200mA, and most motors use more than that). So we use the transistor like an electronic switch to allow power to flow to our motor only when the 555 timer says it's okay.

Shopping List:

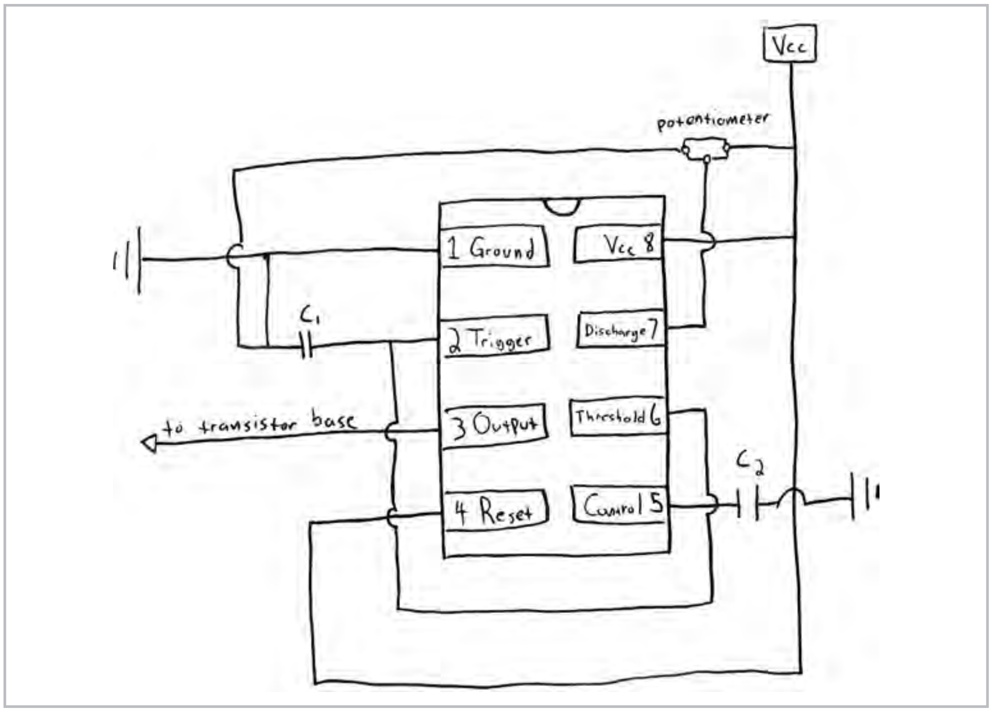
- DC toy motor with wire leads
- Corresponding battery (9V used here) and snap or holder with wire leads (like RadioShack 270-324)
- Breadboard (like All Electronics PB-400)
- Jumper wires (like SparkFun PRT-00124) or hook-up wire to make your own (see Project 6-4)
- On-off-on toggle switch (like SparkFun COM-09609) or other SPDT switch
- Four AA batteries and holder (like SparkFun PRT-00552)
- 555 timer chip (SparkFun COM-09273)
- TIP120 Darlington transistor (Digi-Key TIP120-ND or Jameco 32993)

- 100KΩ potentiometer (Jameco 29103)
- Two 0.1 μF capacitors (electrolytic shown here, but you can also use ceramic, like SparkFun COM-08375)

Recipe:

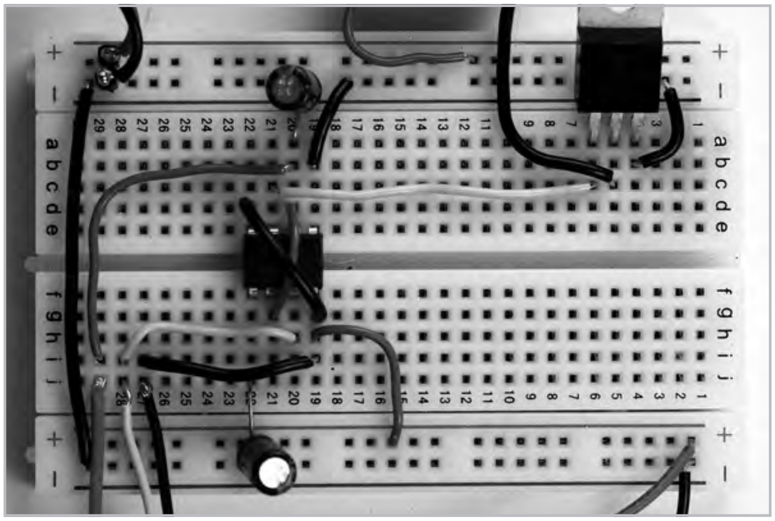
1. Connect the 5V power and ground (from the AA battery pack) to the power and ground columns on one side of the breadboard.
2. On the other side of the breadboard, connect the motor power (9V battery here) to power and ground. Use a long jumper to link both ground columns on the breadboard.
3. Plug the 555 chip into the breadboard with the small dimple on the top left (see Figure 6-26).

FIGURE 6-26 Layout of 555 timer chip circuit



4. Connect pin 1 to ground.
5. Connect pin 2 and pin 6 of the 555 timer with a short jumper.
6. Also connect pin 2 to one of the outside legs of a potentiometer.
7. Also connect pin 2 through a 0.1 μF capacitor to ground.
8. Connect pins 4 and 8 with a small jumper wire. Connect pin 8 to the 5V battery power column and to the other outside leg of the potentiometer. Refer to Figure 6-27 for a close-up of the completed breadboard circuit.
9. Connect pin 5 to ground through a 0.1 μF capacitor.
10. Connect pin 7 to the middle leg of the potentiometer.
11. Plug the transistor into the breadboard as shown so each of the three legs has its own row (see Figure 6-27).
12. Connect the output pin 3 on the 555 chip to the base of the transistor.

FIGURE 6-27 Using a 555 timer to PWM a motor, detailed view

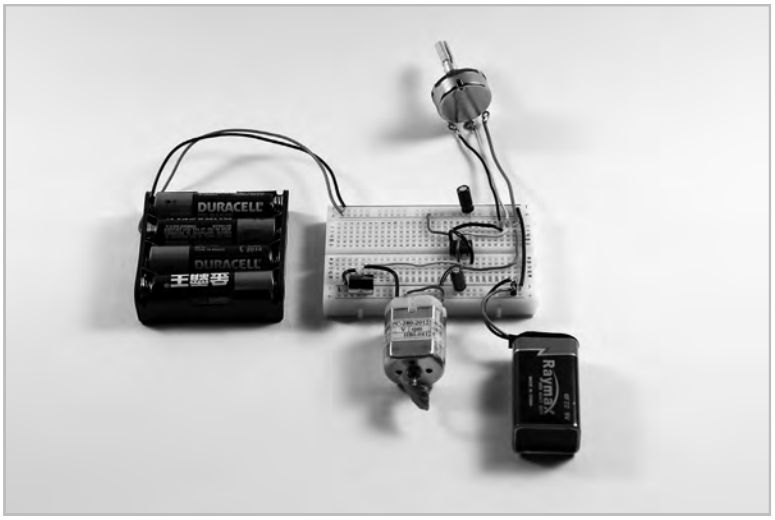


13. Connect the emitter leg of the transistor to ground.

14. Connect one leg of the motor to the collector (middle leg) of the transistor. Connect the other leg of the motor to the 9V battery power column. Your circuit should look something like Figure 6-28.

15. Now your motor should turn on. If it doesn't, turn the knob of the potentiometer clockwise or counterclockwise until you see the motor spin. Watch how the motor speeds up or slows down when the potentiometer is turned. In this circuit, we have the 555 timer wired as a pulse generator, where the length of the pulse depends on the ratio of resistor values from the potentiometer.

FIGURE 6-28 Using a 555 timer to PWM a motor, full-circuit view



Advanced Control of DC Motors

The next step up from using the breadboard, chips, and switches is using a microcontroller, such as the one on the Arduino prototyping platform, to talk to your motor. This is like giving your project a brain. The Arduino can do just about everything we've done with hardware in the preceding example with just a few lines of code.

Project 6-7: Use Software PWM to Control Speed

We will re-create the hardware PWM project in software so you can get a feel for what the Arduino can do. This example assumes you've downloaded the software, installed the drivers for your PC or Mac, and identified the port your Arduino is plugged into. Refer to the "Arduino Primer" section in the appendix for how to set up the Arduino to communicate with your computer.

Unfortunately, you usually can't plug motors directly into the Arduino. The Arduino can source up to only 40mA on each of the input/output pins, and up to 500mA through the power pins when connected through USB. A lot of motors you'll use need more current than this. We get around this issue by using the Arduino to give the motor instructions through a transistor, and giving the motor a separate power supply. This is similar to Project 6-6, except we'll replace the 555 timer with an Arduino.

In this project, we'll create a sketch that listens for input from the stuff you have plugged in (switches, sensors, and so on) and then talks to components you want to control (such as motors). We'll build the circuit first, and then go over how to turn a motor on and off through a transistor with code from the Arduino, and finally use PWM for speed control through the Arduino. (You can find plenty of well-documented example sketches of using the Arduino to talk to motors and other components. For example, see http://arduino.cc/en/Tutorial/HomePage for basic sketches. In most cases. you can just start with these examples and modify them as you see fit.)

Shopping List:

- Arduino with USB cable
- DC toy motor with wire leads
- Corresponding battery (9V used here) and snap or holder with wire leads (like RadioShack 270-324)
- Breadboard (like All Electronics PB-400)
- Jumper wires (like SparkFun PRT-00124) or hook-up wire to make your own (see Project 6-4)

- On-off toggle switch (like SparkFun COM-09276) or other SPST switch
- TIP120 Darlington transistor (Digi-Key TIP120-ND or Jameco 32993)
- 220KΩ resistor (Jameco 30470)
- Diode (SparkFun COM-08589)

Recipe:

1. Connect the 5V power and ground pins on the Arduino to power and ground on one side of the breadboard with jumper wires (see Figure 6-29). On the other side of the breadboard, connect a 9V battery to power and ground. Make sure that the ground is linked between the ground columns on the breadboard.
2. Plug the transistor into the breadboard as shown in Figure 6-30, so each of the three legs has its own row. Connect the emitter pin of the transistor to ground on the breadboard.
3. Connect pin 9 on the Arduino to the base pin of the transistor.

FIGURE 6-29 Powering a breadboard with the Arduino

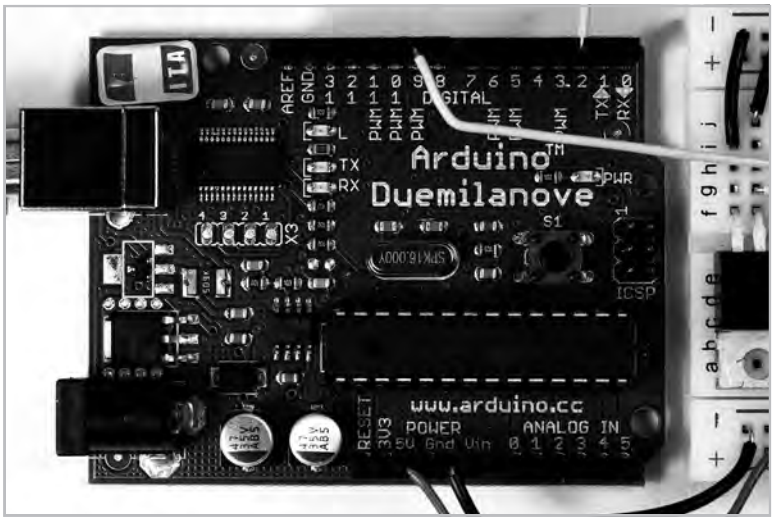
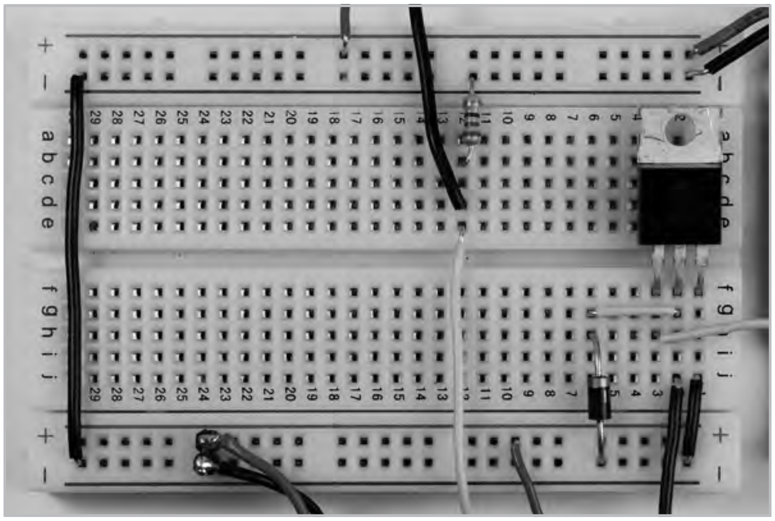


FIGURE 6-30 Close-up of the transistor wiring



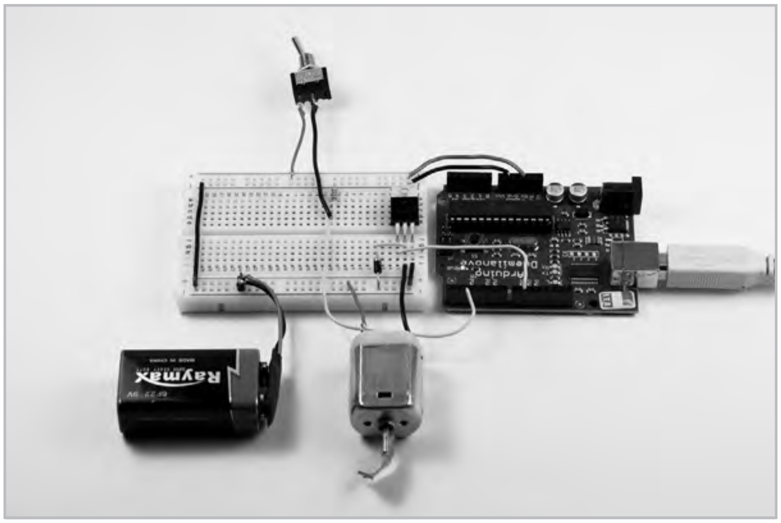
4. Connect the collector of the transistor to ground through the diode. Make sure it's pointing in the right direction, with the stripe mark closest to the middle of the board.

NOTE ***Diodes allow power to flow in one direction and block it in the other. Although not strictly necessary, this is good practice to make sure current is flowing only in the direction you want it to (in this case, into the collector from 9V power). The diode protects the TIP120 transistor from back voltage (power flowing the wrong direction through the motor) generated when the motor turns off.***

5. Connect one leg of the motor to the collector of the transistor on the breadboard. Connect the other leg to the column with the 9V battery power.

6. Place the toggle switch (in the off position) on the other side of the breadboard and connect one leg with a signal wire to pin 2 on the Arduino. Also connect this leg to ground through a 220KΩ resistor. Connect the other leg of the switch directly into the 5V power column on the breadboard fed from the Arduino. Your circuit should look like Figure 6-31.

FIGURE 6-31 The Arduino setup to drive a motor through a transistor



7. Open the Arduino application on your computer and start a new sketch. Type in the following code, verify it, and then upload it to the Arduino.

```
/*
Using Arduino to turn on a motor with input from a switch

Created June 2010
By Stina Marie Hasse Jorgensen, Sam Galison, and Dustyn Roberts
Adapted from code at
http://itp.nyu.edu/physcomp/Tutorials/HighCurrentLoads
*/

const int transistorPin = 9;    // connected to base of transistor
const int switchPin = 2;    // connected to switch

void setup()
{
pinMode(switchPin, INPUT); // set the switch pin as input:
pinMode(transistorPin, OUTPUT);  // set the transistor pin as output:
}
```

```
void loop()
{
if (digitalRead(switchPin) == HIGH) // if switch is on (HIGH)...
  {
  digitalWrite(transistorPin, HIGH);   // turn motor on (HIGH)
  }

else if (digitalRead(switchPin) == LOW)  // if switch is off (LOW)...
  {
  digitalWrite(transistorPin, LOW);   // turn motor off (LOW)
  }
}
```

8. Flip the switch from off to on and see how the motor turns on. When you flip the switch off, the motor should stop. The signal to turn on or off goes from the switch, to the Arduino, and then to the base of the transistor, and allows motor power to flow from 9V power through the transistor to the motor.

9. Now that your motor will turn on and off through a transistor, we'll introduce speed control. You may have noticed a few of the digital input pins on the Arduino board have "PWM" written next to them. These are specifically set to recognize PWM directions from the Arduino code language using the `analogWrite` command. To test this function, open a new sketch and type the following code, verify it, and then upload it to the Arduino.

```
/*
Using Arduino's built in PWM code (analogWrite) for motor speed control
to turn on a motor with input from a switch

for more on PWM with Arduino, see http://arduino.cc/en/Tutorial/PWM

Created June 2010
By Stina Marie Hasse Jorgensen, Sam Galison, and Dustyn Roberts
Adapted from code at
http://itp.nyu.edu/physcomp/Tutorials/HighCurrentLoads
*/

const int transistorPin = 9;    // connected to base of transistor
const int switchPin = 2;    // connected to switch

void setup()
{
pinMode(switchPin, INPUT); // set the switch pin as input:
pinMode(transistorPin, OUTPUT); // set the transistor pin as output:
}
```

```
void loop()
{
if (digitalRead(switchPin) == HIGH)  //if switch is on (HIGH)...
  {
  for (int i=0; i <= 255; i++)  //ramp up speed slowly
    {
    analogWrite(transistorPin, i);  //send value of i to transistorPin
    delay(10);
    }

  delay(500); //wait half a second

  for (int j = 255; j >= 1; j--)  //ramp down speed slowly
    {
    analogWrite(transistorPin, j);
    delay(10);
    }

  delay(500); //wait half a second
  }  //end if

else if (digitalRead(switchPin) == LOW)  // if switch is off (LOW)...
  {
  digitalWrite(transistorPin, LOW);   // turn motor off (LOW)
  }

}  //end loop
```

10. When the switch is turned on, the motor should start spinning slowly, speed up, and then slow back down. This cycle will repeat until you turn the switch off.

Arduino Extensions

If you want robust speed and/or direction control, you might want to check out ready-made modules that interface with your Arduino and do the hard work for you. These modules can make your life easy by incorporating many of the things in the "Helpful Tips and Tricks for Motor Control" section later in this chapter. You'll pay for this convenience, but sometimes it's worth it. For example, SparkFun's ROB-09670 is a motor driver that has an H-bridge already in it, along with other conveniences like direction-indicating LEDs. SparkFun also sells a Digital PWM Motor Speed Controller (ROB-09668), which can control the speed of your motor with PWM without sacrificing torque. Adafruit Industries (www.adafruit.com) sells a Motor/Stepper/Servo Shield for Arduino that can make things even easier. All you do is plug the shield in on top of your existing Arduino, attach the motor wires in the right spots, download the library, and copy a few lines of code. The kits come unassembled, and you need to do a fair amount of soldering to get started. However, there are excellent tutorials linked right from the site.

Hobby Servo Control

All hobby servo motors have circuits inside them that respond to pulses. Each servo has three wires: power, signal, and ground. You need to plug one wire into ground (usually the black one), one wire into a power source in the motor's working range (usually the red one), and one wire into something that can give it pulses (usually the yellow one). This signal is known as *pulse-proportional modulation* (PPM)[8] (also known as pulse-position modulation) and is similar to PWM. This is what the Control System and Required Pulse lines at the top of the hobby servo motor specifications shown earlier in Figure 6-10 tell us.

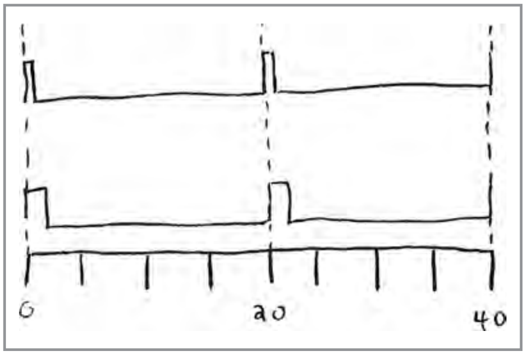
The smarts inside a servo motor expect a pulse every 20 milliseconds (ms), or 50 times a second. Different servos vary, but most servos use a pulse width between 0.5 ms and 2.5 ms out of this 20 ms to send a signal to the servo motor (see Figure 6-32). This signal, or pulse, is similar to repeatedly turning the light on for 0.5 to 2.5 ms, then turning it off until a total on/off time of 20 ms passes, and then repeating the cycle (see Figure 6-32). As with a PWM signal, you can create this pulse in hardware or with software.

NOTE ***There are 1,000 microseconds (μs) in 1 ms. Because the letter*** **u** ***looks like the Greek letter*** **μ** ***but is easier to type, you will often see servo data sheets that state the servo range as 500 to 2,500*** **usec.**

Standard Hobby Servo Control

Standard hobby servos are controlled by pulses that tell them which direction to point. The specs shown earlier in Figure 6-10 indicate that the servo's range is 600 to 2,400 μs, with 1,500 μs neutral. The circuit inside the servo knows that a 600 μs pulse width out of 20 ms means point to one extreme (0°), and a 2,400 μs pulse width means point to the other extreme (180°). Any pulse width between 600 and 2,400 μs moves the motor to a position proportionally between 0° and 180°. If you want the motor to stay put, you just keep sending the same pulse width.

FIGURE 6-32 A PPM signal



A potentiometer meshes with the gears in the servo to tell you exactly where the shaft is at all times. This is called closed-loop *feedback*. You can generate this pulse in hardware or software, or use a radio-controlled (RC) transmitter (like the ones found in model airplane kits) to send the signal to a receiver that talks to the motor.

Project 6-8: Control a Standard Hobby Servo

We'll use the Arduino in this example to generate the pulse, as we did in Project 6-7. However, instead of using the pulse to control the speed of a DC toy motor, it will control the pointing direction of a servo motor.[9] This time, we'll use a code library (which is just a bunch of code that's already written for you).

NOTE ***You can also take the long way and not use the servo code library. It's more involved but also gives you more control. For details, see Section 4.1, "Using the pulse method," at http://itp.nyu.edu/physcomp/Labs/Servo.***

Shopping List:

- Arduino Duemilanove with USB cable
- Servo motor (Hobbico CS-60 used here)
- Breadboard (like All Electronics PB-400)
- Jumper wires (like SparkFun PRT-00124) or hook-up wire to make your own (see Project 6-4)
- Male header pins (SparkFun PRT-00116)

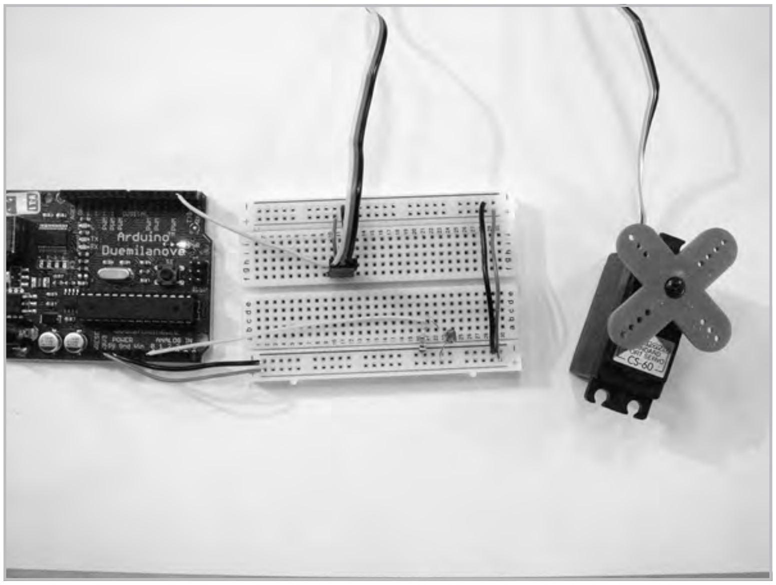
- Diagonal cutters (like SparkFun TOL-08794)
- Photocell (10KΩ – 100KΩ, Digi-Key PDV-P9007-ND used here) and resistor (10KΩ, like SparkFun COM-08374 used here)

NOTE ***You can also use a 1KΩ – 10KΩ photocell (SparkFun SEN-09088). In that case, you should use a 1KΩ resistor (SparkFun COM-08980) to get the best response.***

Recipe:

1. Connect 5V power and ground from the Arduino to the power and ground columns on one side of the breadboard. Use jumper wires to jump connect these to power and ground on the other side of the board.
2. Clip off a set of three header pins from one side of the long row with the diagonal cutters or just snap them off by hand. Choose three rows on the top of the breadboard to hold the pins, and then plug the servo motor connector onto the header (see Figure 6-33).

FIGURE 6-33 Standard hobby servo control circuit



3. Connect the red wire to 5V power, the black wire to ground, and the yellow wire to digital pin 2 on the Arduino. Sometimes this third wire is green or orange, but it will always be different from the red (power) and black (ground) wires.

4. Connect one leg of the photocell directly to power.

5. Connect the other leg to a row on the breadboard of your choice. Then connect this row to ground through the resistor. Also connect this row to analog pin 0 on the Arduino. Your circuit should now look like Figure 6-33.

NOTE ***We need to use the analog pins with a photocell because it's not just an on-off type of input like the switches we've used until now. The photocell will actually indicate a value between 0 and 1023 (as will any analog sensor), depending on how much light is hitting it.***

6. Open a new sketch in Arduino and type the following code. Then verify and upload the code to the Arduino.

```
/*
Servo control from an analog input using the Arduino Servo library

Created June 2010
By Stina Marie Hasse Jorgensen and Dustyn Roberts
Adapted from code at http://itp.nyu.edu/physcomp/Labs/Servo
*/

#include <Servo.h>       // include the servo library

Servo servoMotor;     // creates an instance of the servo object
                      // to control a servo
int analogPin = 0;       // the analog pin that the sensor is on
int servoPin = 2;        // the digital pin for the yellow servo
                         // motor wire
int analogValue = 0;     // the value returned from the photocell

void setup()
{
servoMotor.attach(servoPin);  // attaches the servo on Arduino pin
                              // 2 to the servo object
}
```

```
void loop()
{
// read the analog input from the photocell (value between 0 and
// 1023)
analogValue = analogRead(analogPin);

// map the analog value from the photocell (0 - 1023) to the angle
// of the servo (0 - 179)
analogValue = map(analogValue, 0, 1023, 0, 179);

// write the new mapped analog value to set the position of the
// servo
// servoMotor.write(analogValue);

delay(15);  // waits for the servo to get there before getting
            // another photocell reading
}
```

7. Try to make the servo motor move by using your finger to block the photocell, and then moving it away and letting light hit it. It should move back and forth, but depending on the light in the room, you probably won't get the full range of the servo by doing this.

Continuous Rotation Servo Control

If you have a continuous rotation servo (that you bought or modified from a standard servo), you no longer have control over position. Instead, the signal you give the servo controls the speed.

The maximum speed you can expect with no load is already stated for you in the data sheet. For example, for the Hitec HS-311 servo motor, that speed is 60° in 0.15 seconds. Since there are 360° in one revolution, that means a modified HS-311 servo could finish one full revolution in 0.15 × 6 = 0.9 seconds. Because there are 60 seconds in a minute, dividing 60 seconds by 0.9 seconds gives a speed of 67 rpm.

Stepper Motor Control

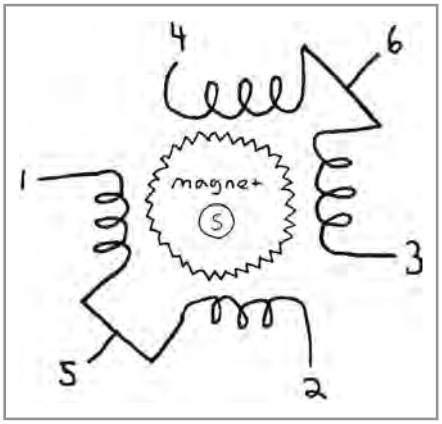
There are two main types of stepper motors: unipolar and bipolar. They have between four and eight wires coming from the housing, and there are no standards as to what the wire colors mean. You use all these wires to give power to different parts of the motor in a specific sequence. The specifics of the sequence determine the stepping behavior (forward, backward, one-half step at a time, and so on). Both unipolar and bipolar steppers can be controlled by the same stepping sequence, but are wired differently.

Unipolar Stepper Motors

Unipolar, or four-phase, steppers have five, six, or sometimes (but rarely) eight wires. They have four sets of wire coils alternating around the outside of the motor housing (hence the term *four-phase*). Unipolar steppers energize the coils all at the same *polarity*, or direction of current flow (hence the term *unipolar*).

A five-wire stepper is the same as a six-wire stepper with the center connections (wires 5 and 6 in Figure 6-34) joined. The six-wire configuration shown in Figure 6-34 is the most popular and probably what you'll find when you pull a stepper motor out of a printer. If you come across an eight-wire, or universal stepper motor, it actually has four independent coils with two connections to each. These can be wired as a unipolar or bipolar stepper.

FIGURE 6-34 Schematic of a six-wire unipolar stepper motor



NOTE ***A six-wire unipolar stepper is just like a bipolar stepper motor but with center connections on each coil. It can also function as a bipolar stepper motor if the manufacturer has designed it that way.***

There are many options for controlling your stepper motor. To minimize time spent with breadboards and programming, it's best to consider ready-made modules that can handle all the hard work of feeding current to the correct wires the right way. Here are some suggestions:

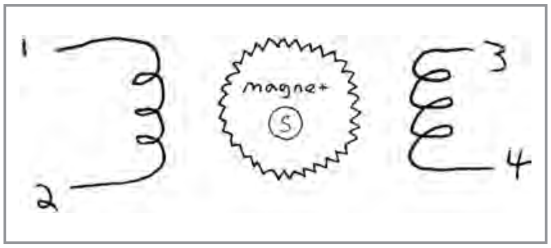
- SparkFun's EasyDriver (ROB-09402) will work with unipolar stepper motors with six or eight wires that are wired as bipolar steppers. This module will work with anything that can generate a 0 to 5V pulse (your Arduino comes in handy here).
- You can use the Arduino to drive the motor directly, but there is more programming involved and you need some extra chips and a breadboard. Luckily, this is mostly done for you. Check out the Arduino stepper tutorial and library at www.arduino.cc/en/Tutorial/Stepper. The code works the same for unipolar and bipolar stepper motors.

- Another option is the Adafruit Industries (www.adafruit.com) Motor/Stepper/Servo Shield for Arduino. All you need to do is plug the shield in on top of your existing Arduino, attach the stepper motor wires in the correct spots, download the library, and copy a few lines of code. The shield works for five- and six-wire unipolar steppers as well as bipolar steppers.

Bipolar Stepper Motors

Bipolar, or two-phase, stepper motors have four wires (see Figure 6-35). They have two independent sets of wire coils in alternating positions around the housing (hence the term *two-phase*). Bipolar steppers move by energizing the coils first in one direction and then reversing the direction as the shaft is turned (hence the term *bipolar*). A bipolar stepper motor will always be stronger than a unipolar motor of the same size. The same options are available to control the motor as with the unipolar type: SparkFun's EasyDriver, the Arduino with the stepper library and some breadboard work, the Adafruit motor shield, and plenty of others.

FIGURE 6-35 Schematic of a bipolar stepper motor



Project 6-9: Control a Bipolar Stepper Motor

In this example, we'll use a bipolar stepper motor and control it with SparkFun's EasyDriver.

Shopping List:

- Arduino with USB cable
- Breadboard (like All Electronics PB-400)
- Jumper wires (like SparkFun PRT-00124) or hook-up wire to make your own (see Project 6-4)

- Stepper motor (SparkFun ROB-09238)
- EasyDriver (SparkFun ROB-09402)
- Male header pins (SparkFun PRT-00116)
- Diagonal cutters (like SparkFun TOL-08794)

Recipe:

1. Break or cut off one set of four, one set of three, and one set of two male headers.
2. Solder male headers onto the EasyDriver (see Figure 6-36). The set of four lines up with the four motor holes, the set of three lines up with the GND/STEP/DIR holes, and the set of two lines up with the GND/M+ holes.

NOTE ***It's easiest to solder if you stick the long ends of the headers into the breadboard, slide the EasyDriver on the short ends, and then heat up the little solder pads around the holes while you add solder. Be careful not to add so much solder that the pins connect to each other!***

3. Break or cut off another set of four male headers and solder them onto the end of the stepper motor wires (see Figure 6-37). Make sure red and green are next to each other on one side, and blue and yellow on the other.
4. Plug the stepper motor header into the breadboard in line with the motor pins on the EasyDriver. The red and green wires should be next to A on the EasyDriver, and the blue and yellow wires next to B.
5. Jump a ground pin from the Arduino to the GND pin on the EasyDriver.
6. Connect Arduino pin 8 to DIR.
7. Connect Arduino pin 9 to STEP.

FIGURE 6-36 Soldering headers onto the EasyDriver board before (top) and after soldering a few header pins (bottom)

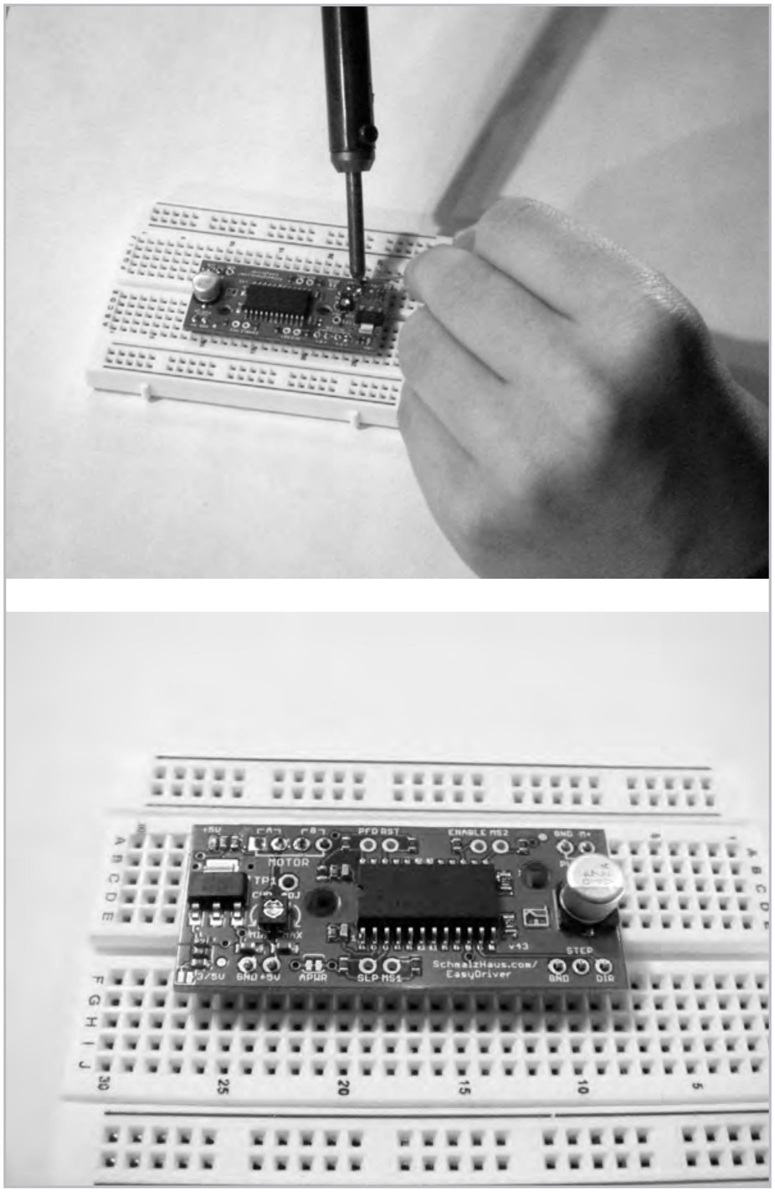


FIGURE 6-37 Soldering male headers onto the stepper motor wires before (top) and after strain relieving with a cable tie and insulating with hot glue (bottom)

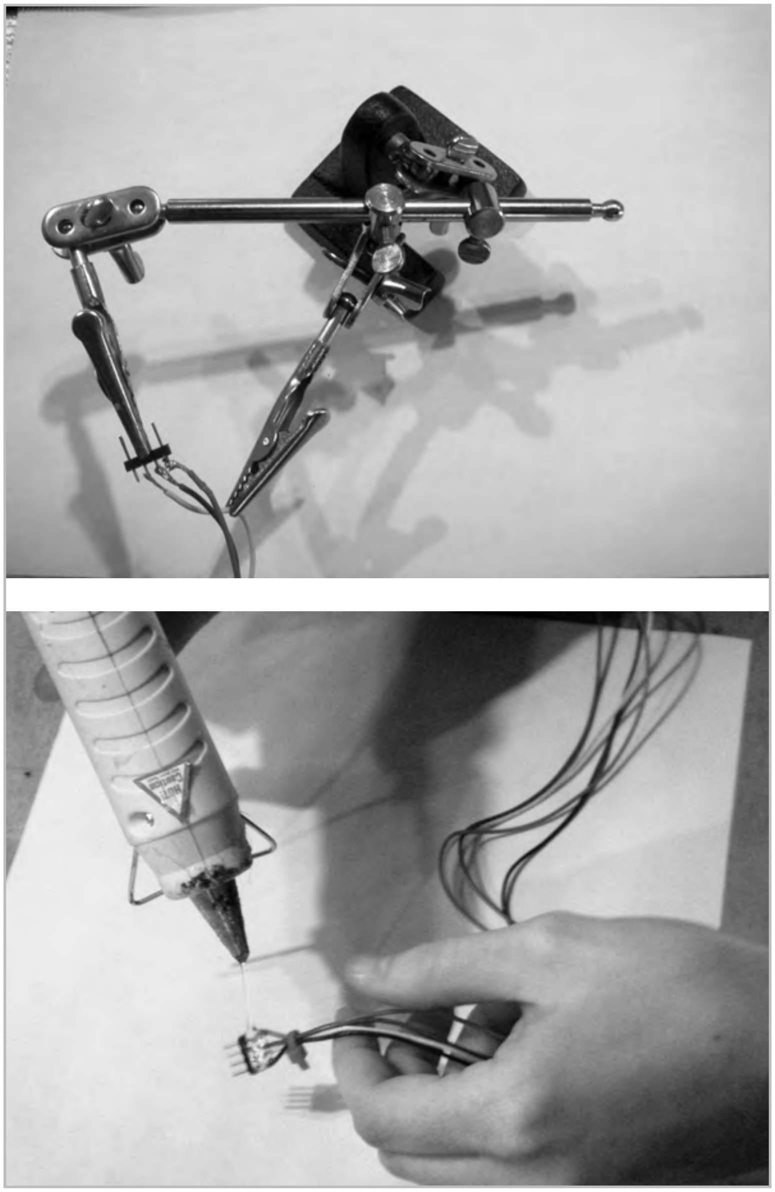
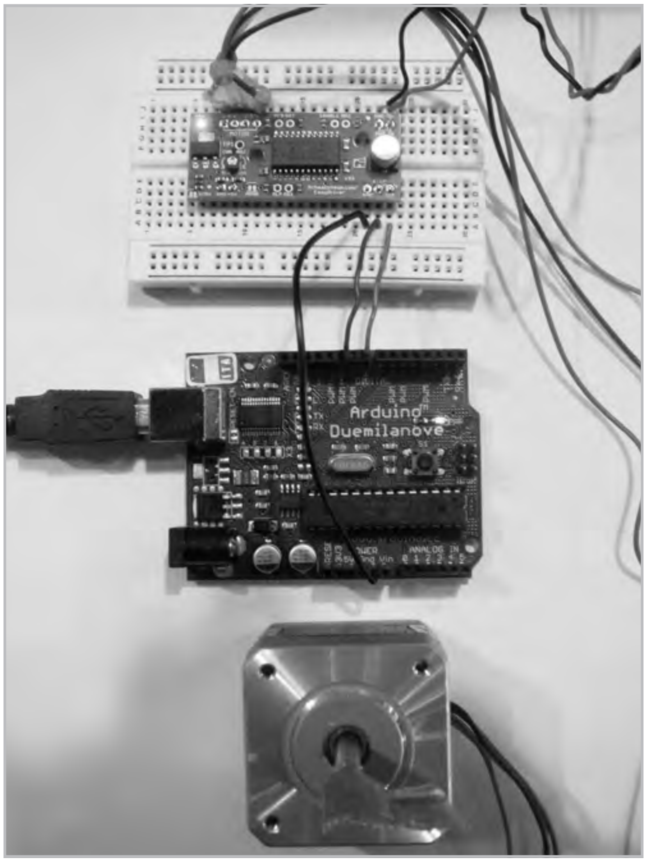


FIGURE 6-38 Stepper motor circuit all wired up



8. Connect a 12V power supply to the M+ and GND pins on the EasyDriver. The example uses a benchtop supply set to 12V. (Refer to the appendix for other ways to get power to your breadboard.) Your circuit should now resemble Figure 6-38.

9. Type the following code, verify it, and upload it to your Arduino.

```
/*
This code is to get one stepper motor moving through
SparkFun's EasyDriver board.
```

```
Created 2010.06.01
By Ben Leduc-Mills and Dustyn Roberts
*/

#include <Stepper.h>  //import stepper library

#define STEPS 200  //this should equal the number of steps our
                   //motor is rated for
Stepper stepper(STEPS, 8, 9); //goes to Arduino digital pins 8
                              //(DIR) and 9 (STEP)

void setup()
{
stepper.setSpeed(200);  //set speed of stepper in RPM
}

void loop()
{
  stepper.step(6400);  //turn one full rotation
  delay(100);  //wait 1/10th of a second
  stepper.step(-6400);  //turn one full rotation the other way
  delay(100);  //wait 1/10th of a second
}
```

10. Your motor should now rotate back and forth! Try putting a little flag of tape on the motor to help you see what's going on.

Linear Motor Control

Linear motors are essentially DC motors that interact with a power screw assembly to make a plunger go in and out, so you can control them in the same way that you control DC motors. Just apply a voltage across the two wires within the stated operating range, and you're set.

The total distance the plunger travels from out to in (or in to out) is called the *stroke distance*. You may want to use the Arduino and/or switches to limit the stroke distance to create whatever movement you want. Most linear motors come with integrated switches and/or a potentiometer to help you control the speed and position.

Solenoid Control

A solenoid is controlled like a DC toy or gearhead motor, in that all you need to do is apply the correct voltage with enough current across the two connections, and it will move. Pull-type solenoids pull the plunger into the housing, and push-type solenoids push the plunger out.

Solenoids are either continuous duty or intermittent duty. *Continuous duty* means that you can turn the solenoid on, and the plunger will either push or pull, and then stay there as long as it's powered. *Intermittent duty* means that when you turn on the solenoid, it will either push or pull for only a set amount of time (sometimes called *maximum on time*). When you remove power from a solenoid, the plunger does not return to its original position on its own. Usually, there will be a spring to return it after it has been pushed or pulled.

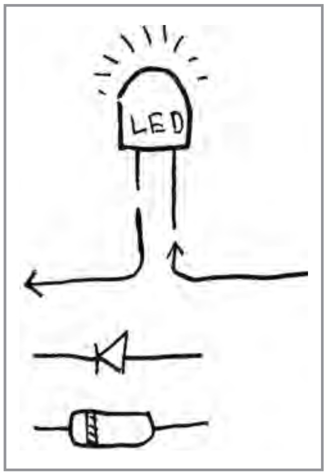
Helpful Tips and Tricks for Motor Control

Whenever you turn a motor on or switch directions, you create mechanical stress on the motor, as well as electrical stress on the attached cables, circuits, and batteries. Current can flow backward through the motor, something called *blowback* or *back voltage*, causing back electromotive force (EMF), and that's all bad.[2] Mechanical and electrical stress decrease the life span of the motor and can wreak havoc on any connected control electronics. Many of the ready-made modules mentioned earlier in the chapter limit this stress and prolong motor life with tactics like slowly ramping up speed, regulating voltage, smoothing the current flow, and using some of the following helpful tips and tricks.

Diodes Are Your Friends

When you reverse the direction on a DC motor or turn on a solenoid, you create a power spike that can sometimes be harmful to other components in your circuit. When this happens, electrical energy can flow in directions you didn't intend. Diodes are little electronic components that let current flow through them in only one direction. They help protect your circuits by ensuring electricity can flow only the way you want it to flow. To use a diode like the one shown in Figure 6-39, all you need to do is put it in line with the intended direction of current flow and make sure it's facing the right direction.

FIGURE 6-39 How to use a diode and LED in a circuit



Light-emitting diodes—LEDs for short—emit light when current runs through them. Like all diodes, these need to be put in a circuit in the correct orientation, or else they will act as a wall and not an open gate. Luckily, most LEDs come with a short leg (ground) and a long leg (power).

The side of the LED over the shorter ground leg is usually flat, so you can still tell which side is ground, even if you clip the legs shorter. Install the LED in the circuit so that current flows through it from the long leg to the short leg.

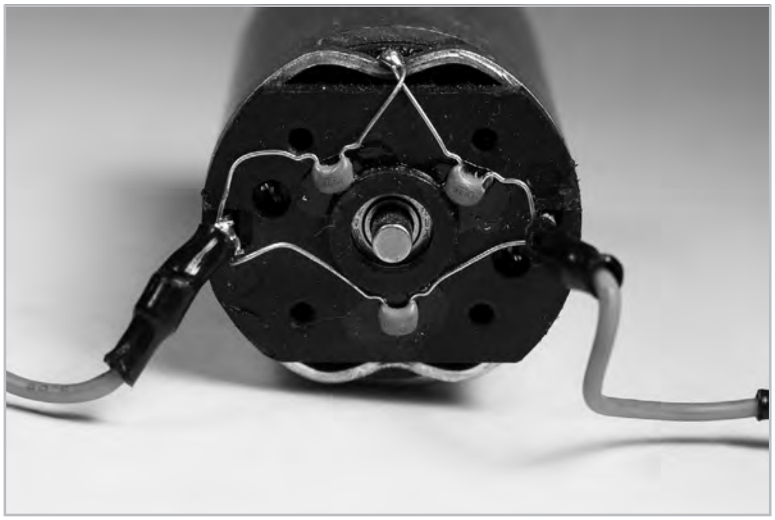
LEDs don't limit voltage on their own, so you need to put a resistor in series with your LED before hooking it up to most power sources, or else you'll fry it. For example, if you're using a 5V power supply, a 220KΩ resistor will work well with most LEDs. At higher voltages, you'll need proportionally higher resistors to protect your LED.

Decoupling Capacitors

You can use capacitors to your advantage in circuits with motors to smooth out the energy spikes that happen when motors are turned on or change directions. Capacitors used for this purpose are called *decoupling capacitors*.

A popular approach is to solder a capacitor across the two leads of a DC motor before ever using it, just to be safe. You may also see capacitors soldered from the motor leads to the motor housing (see Figure 6-40). Make sure to use ceramic (not electrolytic) capacitors for these applications, since they don't care which way the current flows into them. This will allow you to run the motor clockwise or counterclockwise with no worries.

FIGURE 6-40 Decoupling capacitors soldered to DC motor leads



Using decoupling capacitors is a quick-and-dirty method to keep energy spikes in your mechanism's circuit under control, rather than using some of the ready-made smart modules we talked about earlier that do this kind of thing for you. A 0.1 μF capacitor generally works well for bridging the connections on a DC motor and smoothing energy spikes.

Separating Logic and Motor Power Supplies

It's a good idea to separate the power supplies for your motor and the logic—the breadboard circuit or Arduino—that is controlling it. There are a few good reasons for this:

- Most controllers (like the Arduino) and chips (like the 555 timer and H-bridge) take power at 5V. Your motor will most likely want something different. If you try to power your 5V Arduino and your 12V motor from the same battery pack, one of those paths is going to waste a lot of energy or not work at all. Isolating the power supplies means you can choose the right supply for each job.
- If your motor needs more current and voltage than you can safely run through an Arduino (anything over 500mA), you absolutely need a separate power supply.
- Even if your circuit or Arduino can supply the voltage and current your small motor needs, you will still see noise in the system from turning motors on and off and switching directions. Diodes and decoupling capacitors can help this situation, but it's still a good idea to keep the power supplies separate and avoid the problem altogether.

NOTE ***Even if you separate the power supplies for your controller and motor, you must connect the ground wires together. It's good practice to keep the circuit and the power supply grounds at the same low energy level in order for the logic to talk to the motor effectively.***

Relays and Transistors

Transistors and relays are like electronic switches. Mechanical switches are switched on and off with your finger. Transistors and relays are switched by an electrical signal. You need them when working with motors, because most of the time, the amount of current the motor needs and the amount of current allowed to flow through your circuit or Arduino (500mA for the Duemilanove model) are different. By using a transistor or relay, and a separate power source for your motor, you can just tell the transistor or relay to open when you want your motor to get power.

The TIP120 is a common transistor to use for this purpose, and SparkFun's COM-00100 is an easy-to-use relay that will plug directly into your breadboard. The relay will allow you to turn on a motor that needs as much as 5A at 12V with only a 12mA and 5V signal, which an Arduino can easily send.

Motorless Motion

Although motors are the most common actuators, there are a few other options worth mentioning. These include fluid pressure and artificial muscles.

Fluid Pressure

We talked about fluids in the alternative energy section of Chapter 5, so you know a fluid is anything that flows—air, water, or maple syrup. Fluids always take the shape of the container they're in, so they exert pressure in all directions in that container. This pressure depends on the depth and weight of the fluid:

$$\textit{Pressure} = \textit{Depth} \times \textit{Density} \times \textit{Gravity}$$

Viscosity is a measure of the thickness of a liquid. Water has a low viscosity, maple syrup has a medium viscosity, and silly putty has a high viscosity.

Both hydraulic fluid and compressed nitrogen are used in the open-assist gas springs (such as McMaster 9416K14) common on lids, windows, and car trunks. Close-assist gas springs are common on screen doors to avoid slamming. These allow smooth motion in one direction and resist motion in the other, providing help in the stroke direction where help is needed. The compressed gas does the work, and the hydraulic fluid stops the plunger from slamming at the end of stroke.

Hydraulics

Hydraulics are concerned with liquid-driven mechanisms. Liquids are incompressible, so when you try to squish them, they push back. Hydraulic cylinders are normally operated at high pressures (about 1,000 psi or more) and used in backhoes and industrial machinery.

Using hydraulics is a bit like working with AC power. You want to make sure you know what you're doing before you try, or the consequences could be lethal. The equipment also tends to be very expensive. The high pressures and forces that hydraulic systems can create aren't usually necessary for the kinds of projects this book encourages.

Pneumatics

The field of pneumatics deals with gas-driven mechanisms. Gases *are* compressible, and they can store the energy it takes to compress them for later use. Pneumatics are normally used at much lower pressures (around 100 psi) than hydraulics, which makes them much safer. Pneumatic actuators are used where electric motors are dangerous (as in underground mines) or impractical (as in common dentistry tools). Pneumatic drills and nail guns are commonly used for DIY construction projects.

Cheap resources for compressed air include air brush kits, bike tire pumps, car tire pumps, and portable tabletop compressors. Compressors need electricity to squish the air before you can use it as a source of power. Black & Decker (among others) sells a small, portable inflator (Model ASI300) you might find useful for projects in this area.

Air muscles are another technology that uses compressed air. Think of them as a sealed version of those mesh-woven Chinese finger traps you played with as a kid. They work by inflating the mesh-woven tube so the overall result is contraction. The ones from Images (www.imagesco.com/catalog/airmuscle/AirMuscle.html) can contract to 75% of their relaxed length. You will need an air pump that can reach at least 50 psi, so a small compressor or even a bike tire pump would work fine.

Artificial Muscles

There are two flavors of materials emerging that contract when you feed them electrical energy: electroactive polymer actuators and nitinol. Since they mimic human muscle motion, both technologies are commonly referred to as *artificial muscles* or *muscle wire*. They are attractive options for engineers and designers because they could potentially take up much less space and be lighter than motors, leading to mechanisms with more human-like actuators and motion. However, the technologies are immature, require high current, and can be hard to apply.

Shape Memory Alloy

Wire made of nitinol (a nickel-titanium mix) is one example of a material that will shrink when heated past a certain point, and then return to its original length at room temperature. This effect is called *shape memory*, since the wire "remembers" what it's supposed to look like. The metal mix is known as shape memory alloy (SMA).

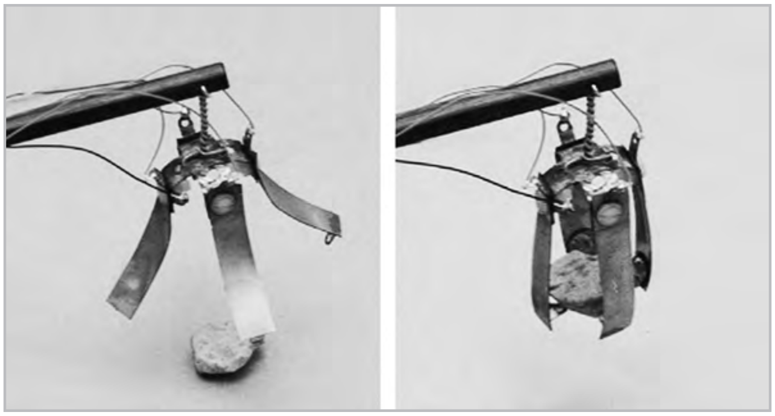
Dynalloy (www.dynalloy.com) is the main manufacturer of an SMA called Flexinol, which is designed to be durable enough to create movement in mechanisms. SparkFun carries some actuators from Miga motors, like the Miga NanoMuscle (ROB-08782), which can be used to make small linear movements.

The recurring complaint about muscle wire is that it contracts only about 3% to 5% of the original wire length, which limits its practical applications, but you can probably make an inchworm robot with a few LEGO pieces and a short SMA wire. For project ideas, check out the *Muscle Wires Project Book* by Roger Gilbertson (Mondo-Tronics, 2000).

Electroactive Polymer Actuators

Electroactive polymer actuators (EAPs) are similar to SMA, but based in plastic instead of metal (although some metal-plastic composites are emerging). They have the ability to contract up to 380%, which is extreme in comparison to nitinol. Researchers have tried using it to make arm-wrestling robots and even control a fish-shaped inflatable blimp, but again, the technology is immature and has not gone mainstream yet, Figure 6-41 shows a four-fingered gripper actuated by EAPs. See the article at www.empa.ch/plugin/template/empa/*/74071) for more information about EAPs.

FIGURE 6-41 Gripper actuated by EAPs (courtesy of Yoseph Bar-Cohen, Jet Propulsion Laboratory/Caltech/NASA)



References

1 Mike Passaretti, Honeybee Robotics, introduced me to this project.

2 Dan O'Sullivan and Tom Igoe, *Physical Computing: Sensing and Controlling the Physical World with Computers* (Boston: Thomson, 2004).

3 Dennis Clark and Michael Owings, *Building Robot Drive Trains* (New York: McGraw-Hill, 2003).

4 Tom Igoe, "DC Motor Control Using an H-Bridge" (http://itp.nyu.edu/physcomp/Labs/DCMotorControl).

5 Tom Igoe, "Using a Transistor to Control High Current Loads with an Arduino" (http://itp.nyu.edu/physcomp/Tutorials/HighCurrentLoads).

6 Gordon McComb, *The Robot Builder's Bonanza*, ed. Michael Predko (New York: McGraw-Hill, 2006).

7 Tom Igoe, "Components" (http://itp.nyu.edu/physcomp/Labs/Components#toc4).

8 Dennis Clark and Michael Owings, *Building Robot Drive Trains* (New York: McGraw-Hill, 2002).

9 Tom Igoe, "Servo Motor Control with an Arduino" (http://itp.nyu.edu/physcomp/Labs/Servo).