7 Motors

7.1 Variety Abounds

A few years ago, a computer was the largest and most expensive component of a robot, while motors and batteries consumed only small percentages of the budget. These days, while motors and batteries have changed little, the relationship has flipped. Microelectronics have shrunk in size and cost so drastically that, for the types of mobile robots we describe in this book, the motors and gears will typically be the most costly items.

An electric motor converts electrical energy to mechanical energy. Motors come in all manner of shapes and sizes. There are electromagnetic direct current (DC) motors and electromagnetic alternating current (AC) motors and a number of variations of each. AC motors are typically used for large machinery (such as machine tools, washers, dryers, and the like) and are powered from an AC power line. You might run across AC motors with titles such as single-phase, split-phase, capacitor start, permanent split-capacitor, shaded-pole and three-phase motors. AC motors are seldom used in mobile robots because a mobile robot's power supply is typically a DC battery.

We will focus on DC motors in this book. DC motors are commonly used for smaller jobs and will suit our purposes well. They also appear in a large variety of shapes and sizes: permanent magnet iron core, permanent magnet ironless rotor, permanent magnet brushless, wound field series connected, wound field shunt connected, wound field compound connected, variable reluctance stepper, permanent magnet stepper, and hybrid stepper motors.

For a robot's needs, a DC motor usually runs at too high a speed and too low a torque. In order to swap these characteristics, a DC motor must be geared down. Connecting the shaft of a motor to a geartrain causes the output shaft from the geartrain to rotate much more slowly and to deliver significantly more torque than the input shaft. A geartrain can be assembled discretely and attached to the motor shaft, or a DC motor can be purchased with the geartrain already prepackaged inside the motor housing.

These compact motors are termed *DC gearhead motors* and will be most useful in putting together a small robot. DC gearhead motors are normally based on permanent magnet ironless rotor motors in order to be as lightweight as possible. They can also be purchased with position encoders integrally connected. Figure 7.1 illustrates two conventional DC gearhead motors.

Most DC motors have two electrical terminals. Applying a voltage across these two terminals will cause the motor to spin in one direction, while a reverse polarity voltage will cause the motor to spin in the other direction. The polarity of the voltage determines motor direction, while the amplitude of the voltage determines motor speed.

However, some DC motors, such as stepper motors, have more than two electrical terminals, often up to six or eight. Signals are applied to these wires, which energize different coils inside the motor sequentially. The rotor is subsequently attracted to each portion and "stepped around" in a continuous fashion. Thus, the timing of these signals determines the motor speed, the phase between the signals determines the motor direction, and the number of commands determines the motor position.

Another type of DC motor with more than two electrical terminals is an assembly known as a *servo motor*. Although the term *servo motor* is used in a variety of contexts, what we are talking about here

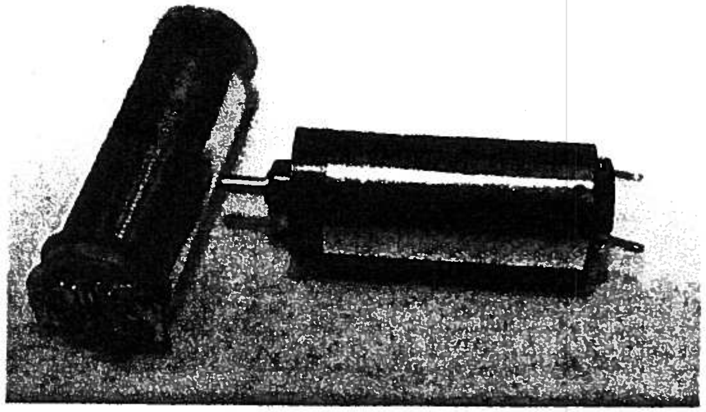


Figure 7.1. These DC gearhead motors manufactured by Escap are permanent magnet ironless rotor models with 54:1 and 27:1 geartrain ratios. The motor on the left has an attached printed circuit board, which interfaces to a position encoder encapsulated in the motor housing.

is the three-wire DC servo motor that is often used for a control surface on a model airplane or for a steering motor on a radio-controlled car. This type of assembly incorporates a DC motor, a geartrain, limit stops beyond which the shaft cannot turn, a potentiometer for position feedback, and an integrated circuit for position control. Of the three wires protruding from the motor casing, one is for power, one is for ground, and one is a control input where a pulse-width signals to what position the motor should servo. When we speak about a motor servoing to a position, we mean that an electrical circuit directs the motor to rotate to the commanded position and keeps it there. If you try to grab the motor shaft while the servo loop is running, and forcibly rotate the shaft to a different position, the electrical circuit will read the angle of the potentiometer, realize that the shaft is no longer at its commanded position, and increase the current to the motor. This will increase the torque the motor puts out and the motor will push back against the torque you are applying with your hand. The servo motor will continue to do this until the shaft has rotated back to its commanded position. A servo motor then is an assembly which consists of a DC gearhead motor,

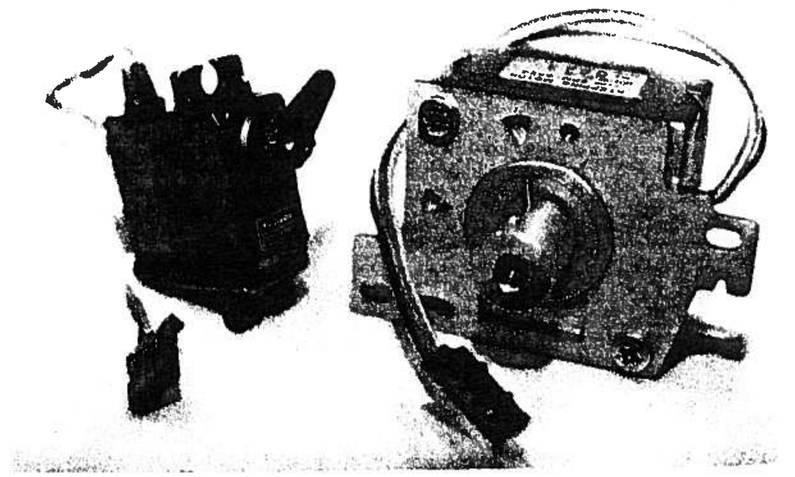


Figure 7.2. Shown on the left is a Futaba servo motor and on the right, a stepper motor. Note the three-wire lead on the servo motor and the six wires protruding from the stepper motor.

a position sensor on the shaft, and an integrated circuit for control, all packaged into the casing of the servo motor.

The flaps and control surfaces on model airplanes do not have to rotate continuously, so limit stops are added to these motors and a single-turn potentiometer then suffices to provide position information back to the integrated circuit that controls the motor position. Servo motors can be extremely compact and easy to control, and because they are mass produced for the toy industry, they are often cheaper than other DC gearhead motors. Although they rotate less than 360 degrees and hence are not suitable for wheeled robot propulsion, these model airplane servo motors often find their way into robot grippers, arms, and legs. Figure 7.2 shows both a servo motor and a stepper motor.

If you want to skip ahead to building Rug Warrior's locomotion system, we will tell you right now that our choice was to take Royal Titan Maxi Servos, available from Tower Hobbies, strip out the controller chips and potentiometers and remove the limit stops, and use these motors as continuously revolvable DC gearhead motors to

drive Rug Warrior's wheels. This is the cheapest, simplest solution we could find for this book's example robot.

DC motors are also characterized another way: as either brush-type or brushless motors. These designations refer to the manner of commutation used that converts direct current from the battery into the alternating current required to generate motor action. If the DC current is commutated mechanically with brushes, the commutator segments at the ends of the rotating rotor coil physically slide against the stationary brushes that are connected to the motor's terminals on the outside of the case. If the DC current is converted into AC current in the rotor electronically, with position sensors and a microprocessor controller, then no brushes are needed. Brush-type motors are more common and cheaper. Brushless DC motors have an advantage over brush-type motors in that friction is reduced, leading to longer life and finer control for the motor. Also, brushless motors can produce less radio frequency interference. The trade-off is that brushless DC motors require more extensive control circuitry in order to do the commutation electronically.

In addition to electromagnetic DC and AC motors, there are a few other types of motors that are not electromagnetic. Piezoelectric ultrasonic motors, which can be found in autofocus lenses in some Japanese cameras, work on the principle of mechanical bending of a piezoelectric ceramic, using frictional coupling to a rotor. The Japanese have also introduced these motors into headrest actuators in new luxury cars, paper pushing mechanisms in copiers, and in tinier versions in wristwatches for use as silent (vibrating) alarms. Ultrasonic motors, in contrast to conventional electromagnetic motors, spin at lower speeds and with higher torques, alleviating the need for geardown. This means they can be compact and lightweight, but the frictional coupling between rotor and stator results in problems of wear. A small piezoelectric ultrasonic motor is shown in Figure 7.3.

Also, in research labs around the world, *electrostatic* motors are being micromachined out of silicon in dimensions on the scale of a human hair. Electrostatic motors work on the principle of charge attraction, where a force is created as two charged plates slide past each other. At small scales, electrostatic forces can be relatively strong, but for large motors, electromagnetic forces are more effec-

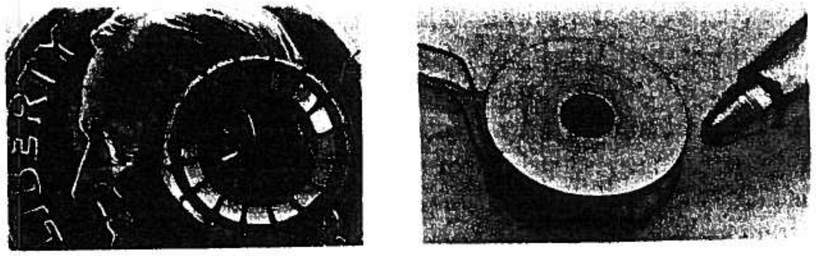


Figure 7.3. This 8 millimeter (mm) diameter piezoelectric ultrasonic motor, built at the MIT Mobile Robot Lab, is composed of two pieces: the stator and the rotor. The *stator*, shown on the left, is a steel ring with piezoceramics bonded onto the bottom that causes a wave to travel around the ring. The top piece, the *rotor*, is made of brass and, when pressed against the stator, is dragged along and spins. The stator with a rotor on top is illustrated on the right.

tive. Although micromotors have not reached the stage of practical use, they are intriguing.

Shape memory alloys can also be used for robot actuation. A shape memory metal such as Nitinol changes shape reversibly on being heated and cooled. Mondo-tronics, Inc., sells a small, six-legged robot (shown in Figure 7.4) that is actuated by these materials. When the wire is heated by passing current through it, the wire changes shape and shrinks, causing a leg to lift. When the wire is cooled (i.e., when no current is passing through it) the wire changes back to its original longer shape and the leg goes back down. The wires are attached to the legs in such a way that, while three legs lift the others push backward. Alternating this pattern between the two sets of three legs causes the robot to propel itself forward. Plans and instructions for building a similar microrobot called *Stiquito* are available to the public. If you have access to the Internet, you may acquire this information via anonymous FTP. Connect to site cs.indiana.edu, and look in the pub directory.

Even more esoteric is a new class of actuators that are starting to a ppear in research laboratories around the world. These are cotton-like fibers that act similarly to artificial muscles. With the alternating addition of acidic and basic solutions, these actuators can shrink and expand up to 1,000 times their original volume with strength and speed equal to those of human muscle. While still a laboratory curiosity, these polymer gels may prove to be the technology of the future.

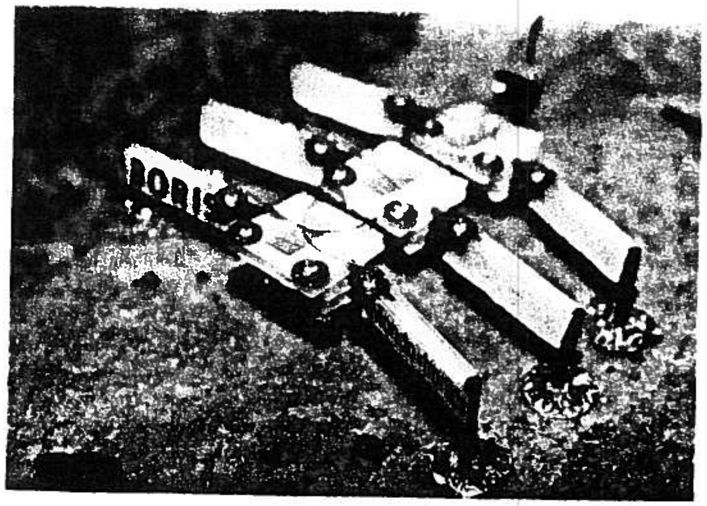


Figure 7.4. This 10 centimeter (cm) robot from Mondo-tronics weighs 50 grams (g) and is actuated by shape memory wires which are wrapped around various screws mounted on the legs and body. Passing 200 milliamperes (mA) of current through a sequence of wires causes alternating legs to lift up and move forward.

7.2 How a DC Motor Works

For the project at hand, let us focus on how permanent magnet DC gearhead motors work. Understanding the mechanism behind the production of torque is helpful when trying to read a motor specification sheet for choosing the correct-sized motor. Such understanding will be helpful again later, when designing the power electronics for controlling the motor from a microprocessor.

Electromagnetic forces in DC motors come about when current-carrying conductors are placed in magnetic fields, as illustrated in Figure 7.5. Magnetic fields can be generated by permanent magnets. Flux lines across an air gap flow from one magnet's north pole to another magnet's south pole. The Lorentz force law states that current-carrying conductors placed in magnetic fields create forces. The force, F, created is perpendicular to both the direction of the

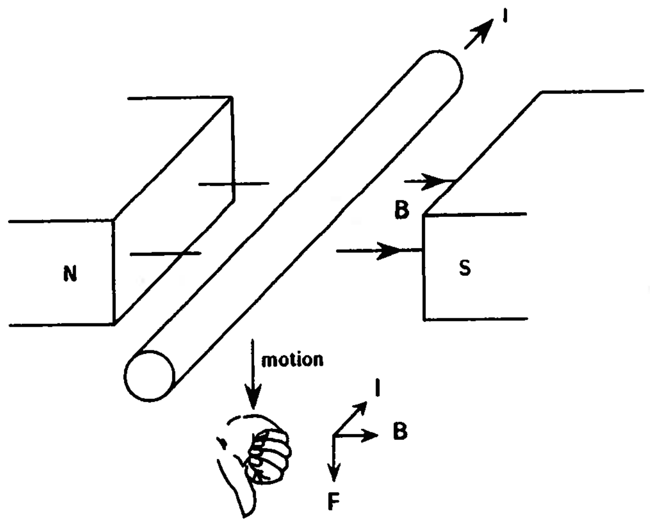


Figure 7.5. A magnetic flux field, B, is set up by the permanent magnets in the direction from north pole to south pole. A current-carrying conductor placed in such a field experiences a force acting on it. The resultant force is directed downward.

current, I, and the direction of the flux field, B. The direction of F is determined by the right-hand rule, where the fingers curl from the direction of the current toward the direction of the flux field and the thumb points in the direction in which the resultant force is created. In the case of Figure 7.5, the force produced is in the downward direction.

Rotary motion requires a loop of wire. Figure 7.6(a) shows a loop of wire mounted on an axis of rotation and situated in the flux field set up by the permanent magnets. Figure 7.6(b) illustrates the resulting forces. Because forces are created in a direction perpendicular to both the current's direction and the magnetic field's direction, current going into the loop along the top generates, according to the right-hand rule, a force acting downward. Current coming out along the bottom portion of the loop creates a force acting upward. The force disparity, acting at a distance from the center of rotation, causes the loop to experience a torque. The loop will rotate until a force disparity no longer exists. That point would be reached when the plane of the loop is vertical and the forces on the top and bottom

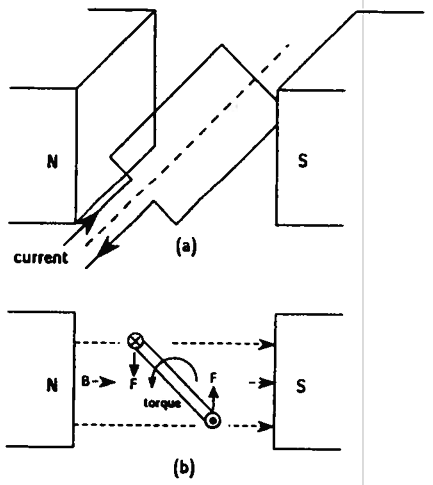


Figure 7.6. (a) This loop of wire has current flowing into the page on the left side and out of the page on the right. (b) The resulting oppositely directed forces, acting at a distance from the center of rotation, cause the loop to rotate until it is vertical.

portions of the loop would both act through the center of rotation, resulting in zero torque.

Continuous rotary motion can be achieved by reversing the direction of the current just as this point is about to be reached. The process of deriving this necessary alternating current from a DC battery is called *commutation*. Mechanical commutation requires a set of brushes that allow the ends of the loop of wire to slip across the contacts of the battery. The commutator setup is shown in Figure 7.7.

A disassembled DC gearhead motor is shown in Figure 7.8. A large number of loops of wire are usually incorporated in order to increase the torque of the motor. These loops are wrapped around an armature that can contain an iron core for increased flux or be ironless for lighter weight. Two half cylindrically shaped permanent magnets are housed along the inside of a steel casing, which provides a flux return path. The wound armature is fitted between the magnets, leaving a small air gap. As the current through the armature alternates, a force is created, causing the armature and the shaft to rotate.

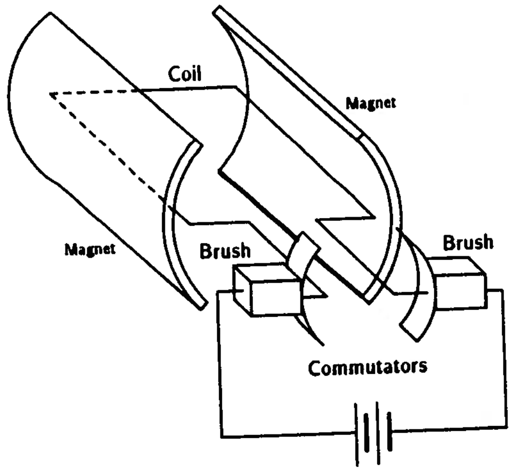


Figure 7.7. A commutation system using brushes is one way to make a DC motor. The commutator segments are attached to the loop of wire and rotate with it, while the brushes remain stationary as the commutator segments slide past.

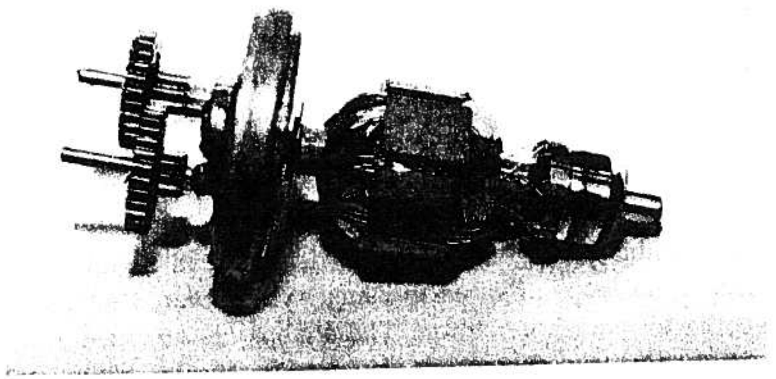


Figure 7.8. A permanent magnet DC gearhead motor shown here has been removed from its housing. Windings of the armature around a central core with ends connected to commutator segments can be seen at the right, while the geartrain is mounted on the shaft at the left. A cylindrical housing (not shown) fits around the armature and holds two permanent magnets along its inner shell.

7.3 Sizing a DC Motor

Selecting an appropriate motor for your robot involves both under standing the loads that the robot will impose on the motor and th performance that the motor can deliver, as detailed in the manu facturer's data sheets. Some manufacturers present the pertinen characteristics in the form of a graph, while others list the specifi cations in table format. Sometimes, if the motor is obtained from surplus dealer or extracted from a toy, it is not possible to obtai data sheets, in which case simple experiments can be performed t measure the pertinent characteristics. Whatever the case may be, it is useful to have a clear understanding of the motor language and to brush up on the conversions between various units of measurement.

7.3.1 Torque, Speed, Power, and Energy

Torque is the angular force that a motor can deliver at a certain distance from the shaft. For instance, 5 oz.-in. of torque means that, at a distance of 1 inch away from the shaft of a motor, the motor is strong enough to pull up a weight of 5 ounces through a pulley (see Figure 7.9). In metric units, motor torques are often specified in Newton-meters (Nm). (When you try to imagine how much force a Newton is, think of the weight of an apple. A force of 1 Newton is about equal to the force that gravity exerts on one apple's mass.) Alternatively, metric units for torque can also be found specified in terms of *gram-force-centimeters* (gf-cm), where a gram-force is meant to signify the force that gravity exerts on 1 gram of mass. We will stick to metric units in this book, but some conversions to keep handy are:

$$1\,\mathrm{N} = 1\,\frac{\mathrm{kg\text{-}m}}{\mathrm{sec}^2} = 0.225\,\mathrm{lb}$$

$$1\,\mathrm{kg} = 2.21\,\mathrm{lb}\,(\mathrm{mass}) \qquad \text{and} \qquad 1\,\mathrm{in} = 2.54\,\mathrm{cm}$$

Also, when we begin to talk about electrical power being converted to mechanical power in a motor, it is useful to keep straight the relationships involving *power* (in watts) and *energy* (in joules). *Power* is the rate at which you are using up *energy*. The relationship between power and energy is expressed as:

$$1\,\text{Watt}{=}1\,\frac{\text{Joule}}{\text{sec}}$$

Figure 7.9 illustrates the electrical to mechanical power conversion of a DC motor. The electrical power supplied to the motor, P_e, equals the voltage, V, across the motor's terminals times the current, I, through the motor. The current, measured in units of amperes, is the amount of charge, in coulombs, passing through any cross-section of a conductor per second:

$$P_e = VI$$

$$1\,\text{Ampere} = 1\,\frac{\text{Coulomb}}{\text{sec}}$$

$$1\,\text{Watt} = 1\,\text{Volt}\cdot\text{Ampere} = 1\,\text{Volt}\cdot\frac{\text{Coulomb}}{\text{sec}}$$

Mechanical power, P_m, equals the torque, T, output by the shaft times its angular speed, ω, where the torque is taken in Newton-meters and the angular speed is measured in units of radians per second:

$$P_m = T\omega$$

$$\frac{2\pi\,\text{rad}}{\text{sec}} = 1\,\frac{\text{rev}}{\text{sec}}$$

$$1\,\text{Watt} = 1\,\frac{\text{Nm}}{\text{sec}}$$

Since power is energy per unit time, this tells us that one joule of energy can be expressed in two ways: either as 1 Newton-meter or as 1 coulomb-volt:

$$1\,\text{J}{=}1\,\text{Nm} \qquad \text{and} \qquad 1\,\text{J}{=}1\,\text{CV}$$

This is just reaffirming the fact that energy is energy, whether it comes from a mechanical origin or an electrical origin. A motor is just a transducer transforming energy from one form to another.

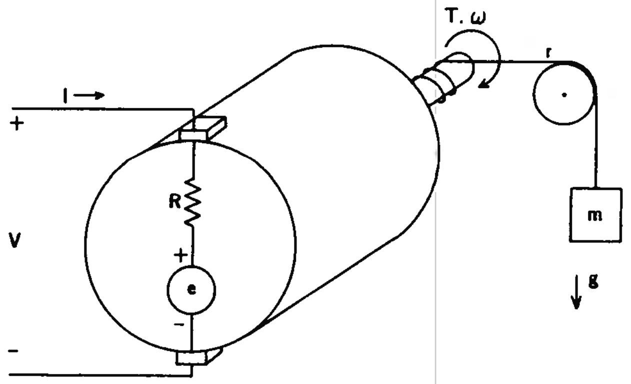


Figure 7.9. A simple model of a DC motor is an equivalent circuit that model the motor windings as having a resistance, R, and generating (when running a back-emf voltage, e. The electrical power input to the motor is the produc $P_e = VI$, and the mechanical power output is the product of torque and rotationa speed, $P_m = T\omega$.

7.3.2 A Motor Model

These relationships, describing the conversion of electrical power tc mechanical power in a permanent magnet DC motor, can be clearly seen by the equivalent circuit model shown in Figure 7.9. The mechanical output power (due to losses from friction, windage, heating in the coils, and so on) will be some fraction of the electrical input power. This percentage is given as the efficiency, η, where:

$$P_m = \eta P_e$$

The rotor coil that we saw in Figure 7.6 is essentially an inductor with a resistance R. When the rotor is turning, the commutator segments sliding past the brushes create an alternating current in the armature windings. A changing current, $\frac{di}{dt}$, through an inductor induces a voltage across it:

$$v = L\frac{di}{dt}$$

where L is the proportionality constant called the inductance. As the motor turns, this voltage is induced and opposes the applied driving voltage. The faster the motor turns, the more often the current switches direction, and so the larger the induced voltage becomes. Since this voltage opposes the applied drive voltage, as it increases, it tends to limit the current through the resistance, R. As the current falls, less flux is created around the conductor and the torque also falls. In summary, as the speed goes up, the torque goes down.

The rotating motor then can simply be modeled by the induced voltage, e, called the *back-emf* (*emf* stands for electromotive force) and the winding resistance, R. The applied voltage is related to the back-emf and the current through the motor by:

$$V = IR + e$$

Note that, when the motor is not rotating, e is 0 V and the current through the motor is just equal to the applied drive voltage divided by the resistance. This is the current required to start the motor from zero speed, called the *starting current* or *stall current*, I_S:

$$I_S = \frac{V}{R}$$

When the rotor is rotating, e increases proportionally with the speed of the armature:

$$e = k_e\omega$$

where k_e is called the back-emf constant. The applied voltage is then related to the current and the armature speed by:

$$V = IR + k_e\omega$$

The negative feedback provided by the back-emf causes the motor to settle to a steady-state operating point of speed and torque, as determined by the load and the applied voltage. The torque that the motor produces is dependent on the flux field around the loop of the conductor, and that flux is controlled only by the current. The torque increases linearly with current with a proportionality constant k_t, known as the *torque constant*:

$$T = k_t I$$

Solving for I and plugging it into the equation above, we get:

$$V = \frac{TR}{k_t} + k_e\omega$$

It turns out that k_t is actually equal to k_e. We can see this fror the fact that the mechanical power output by the shaft will be th electrical power input, minus the I^2R losses due to heating in th resistor:

$$P_m = P_e - I^2R$$

$$T\omega = VI - I^2R$$

Plugging in for T and V,

$$k_t I\omega = (IR + k_e\omega)I - I^2R$$

gives

$$k_t = k_e = k$$

The applied voltage is then related to the torque and speed b the constant k:

$$V = \frac{TR}{k} + k\omega$$

Rearranging, we find that the speed-torque relationship is linea with a negative slope:

$$\omega = -\frac{R}{k^2}T + \frac{V}{k}$$

These relationships can be more clearly seen when plotted alon with the motor performance curves.

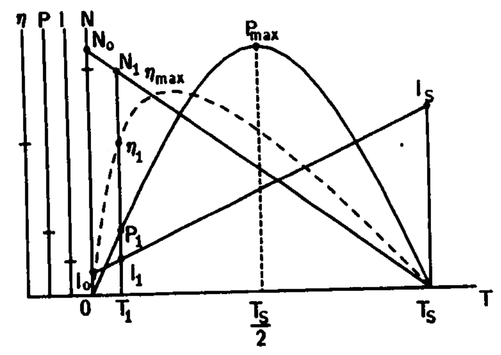


Figure 7.10. For a given voltage, a DC motor has the typical drooping characteristics of speed, N, decreasing linearly torque, T. As the current, I, is increased, the torque is increased, also linearly. Power output, P, is the product of torque and speed and has a quadratic characteristic. Maximum efficiency, η_{max}, occurs at a lower torque than the maximum power output torque.

7.3.3 Speed-Torque Curves

The speed and torque characteristics for a DC motor depend on a variety of parameters that have to do with the geometry of the motor, the materials involved, the number of windings, and the voltage at which the motor is driven. Typically, a manufacturer provides a data sheet showing the pertinent characteristics. These are usually illustrated in a speed-torque graph for a given applied drive voltage. Efficiency, current, and power output are often plotted along with speed on the vertical axis against torque on the horizontal axis, as shown in Figure 7.10.

We can see in Figure 7.10 that the speed-torque curve is linear, with a negative slope as we derived, and has a y-intercept dependent upon the applied voltage. Also, the current increases linearly with torque and is independent of applied voltage, as we showed earlier. The power curve has a negative quadratic form which is understood by remembering that:

$$P_m = T\omega$$

and plugging in our equation for ω:

$$P_m = -(\tfrac{R}{k^2})T^2 + \tfrac{V}{k}T$$

where we see the negative quadratic dependence of power on torque

You will find it useful to check a few points of interest on a moto data sheet in choosing the most appropriate motor for your robot The *no-load speed*, marked N_o in Figure 7.10, is the speed, at a give voltage, at which the torque is 0. (N usually refers to the angula speed in units of rpm. Remember to convert to $\frac{\text{rads}}{\text{sec}}$ when pluggin into these equations for ω.) This is the speed of the motor wit nothing attached to the shaft. That is, the no-load speed, the valu of ω for $T = 0$, is just

$$\omega_{max} = \tfrac{V}{k}$$

The current in this no-load condition, I_o, is called the *no-loa current* and is that required to overcome motor friction and windage.

At the other end of the scale, the torque that the motor can deliver just as it stalls and can no longer rotate is known as the *stal torque*, T_s. The current at this condition, I_S, is the stall current. Since the motor is not moving when stalled, the back-emf is 0 and the maximum current, I_S, is just the applied voltage divided by the coil resistance, as mentioned earlier. Torque being proportional to I, the maximum torque is:

$$T_S = \tfrac{kV}{R}$$

At any given point of operation of torque and speed, the mechanical power output is the product of the two. The torque at which the maximum power occurs can be found by taking the derivative of the power with respect to the torque, setting the result equal to 0, and solving for T:

$$\tfrac{dP_m}{dT} = 0 = -\tfrac{2RT}{k^2} + \tfrac{V}{k}$$

$$T = \tfrac{kV}{2R}$$

or

$$T = \tfrac{1}{2}T_{max}$$

Thus, the point of maximum power output is attained at half the stall torque. The corresponding speed at this operating point is then found to be:

$$\omega = -\frac{R}{k^2}\frac{kV}{2R} + \frac{V}{k} = \frac{V}{2k}$$

or

$$\omega = \tfrac{1}{2}\omega_{max}$$

The maximum power then is simply:

$$P_m = \tfrac{1}{4}\omega_{max}T_{max}$$

The ratio of mechanical power output to electrical power input is the *efficiency*, η. Note that maximum efficiency cannot be achieved at maximum power output. In fact, the point of maximum efficiency, where you would like to drive your motor, is a low-torque, high-speed operating point. Consequently, we typically select an oversized motor so that it can run at an efficient operating point while supplying enough torque.

It turns out that the maximum efficiency, for reasons we will not go into here, can be calculated from the measurements of the no-load current, I_o, and the stall current, I_S:

$$\eta_{max} = (1 - \sqrt{\tfrac{I_o}{I_S}})^2$$

This can be useful for characterizing a motor for which you do not have data sheets.

The data shown in Figure 7.10 are for one given value of applied voltage. If the motor is run at a lower voltage, the speed-torque line shifts downward as shown in Figure 7.11(a). As the voltage is decreased, the speed and the torque are both decreased. Changing the voltage changes the speed of the motor. Another way to change the speed without having such an adverse effect on the torque (in fact, a method that has an advantageous effect on the torque) is to use a geardown. As shown in Figure 7.11(b), gearing down the motor by, say, a factor of 2, cuts the no-load speed in half while doubling the stall torque. Thus, power is maintained constant through

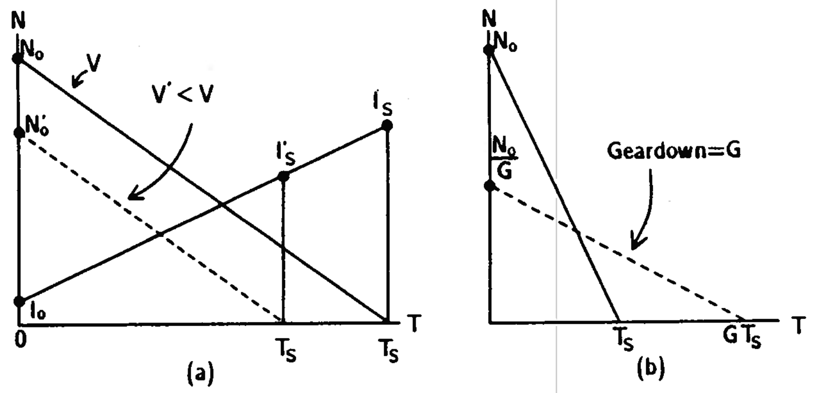


Figure 7.11. (a) Running the motor at a lower voltage causes it to slow down for all values of torque output. (b) Gearing down the motor reduces the speed by the gear ratio, G, and increases the torque by the same factor, G.

a lossless (frictionless) geartrain. Typically though, there are losses both through the motor and again through the geartrain. Good motors these days might have efficiencies of 90% or better, but cheap toy motors (like the ones we will use in the Rug Warrior prototype) might be only 50% efficient, or less. Adding these losses to those through the geartrain and then taking into account the losses between wheels and the ground (from friction, slippage, etc.) results in a system that is not very efficient. For Rug Warrior, most of the energy from its battery pack goes into the propulsion system. Powering Rug Warrior's computers and sensors will be practically insignificant in comparison.

7.4 Gears

Geartrains and transmission systems come in a variety of forms, such as spur gears, planetary gears, rack-and-pinion systems, worm gears, lead screws and belt-and-pulley drives. Figure 7.12 illustrates a number of these mechanisms. High-quality geartrains are usually metal, but plastic gears are often found in toys.

The DC gearhead motor shown earlier in Figure 7.8 had a *spur gear* set on the output shaft. Spur gears are the most common forms of gears found in DC gearhead motors. A schematic of a two-level

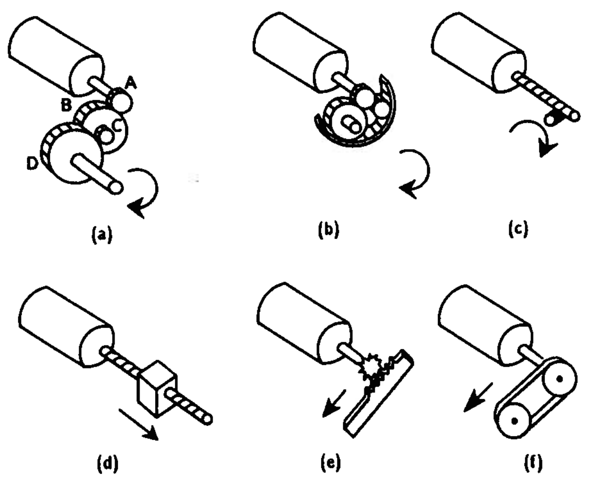


Figure 7.12. (a) Spur gears mesh pairs of gears with different numbers of teeth to achieve speed reduction. (b) Planetary gears have several gears meshed in an outer ring for large reduction. (c) Worm gears produce rotary motion at right angles to the shaft. (d) A lead screw and nut can create linear motion as can (e) rack-and-pinion systems and (f) belt-and-pulley drives.

spur geartrain is shown in Figure 7.12(a). The small gear, mounted directly on the motor shaft, is called a *pinion* and has to rotate many times to turn the gear it is meshed to once. Thus, even though the pinion may spin very quickly, the gear it is attached to spins very slowly. If A, B, C, and D denote the number of teeth on each corresponding gear in the figure, then the speed of the output shaft is related to the speed of the input shaft by:

$$\omega_{out} = \frac{A}{B}\frac{C}{D}\omega_{in}$$

where the final speed has been decreased by the geardown ratio.

Planetary gears are similar to spur gears but are less common in low-end gearhead motors and are slightly more expensive. The difference between planetary gears and spur gears is that planetary gears, as shown in Figure 7.12(b), fit a number of gears concentrically inside a toothed ring. This configuration produces greater efficiency and higher output torques in a smaller package. Planetary geartrains are sometimes found in portable battery-powered screwdrivers and drills.

Worm gears are another means for achieving large geardown in a small space. Worm gears, shown in Figure 7.12(c), instead of having teeth, are threaded and match to a lead screw attached to the shaft of the motor. In this way, the output motion is turned to right angles from the motor shaft.

For linear motion, a *lead screw and threaded nut* can be used. Figure 7.12(d) illustrates how the motor shaft turns the lead screw and a threaded nut moves linearly down the shaft, depending on the number of threads per inch on the lead screw. Lead screws can give very large geardown but are not very efficient.

Rack-and-pinion systems, Figure 7.12(e), also turn rotary motion into linear motion. In this case, a small pinion gear on the motor shaft rotates against a straight length of rack having matching teeth, propelling the rack linearly back and forth.

Another linear motion mechanism is the belt-and-pulley system, shown in Figure 7.12(f). This is the mechanism used to drive a tank-treaded vehicle, such as we will describe later for Rug Warrior II.

7.5 Motor Data Sheets

While it is possible to buy a plain DC motor and attach any number of gear-reduction mechanisms for propelling your robot, we will focus on DC gearhead motors here for Rug Warrior (typically with spur gears) because it makes life easier when geartrain and motor are packaged together. There is no need to find a machine shop and spend time making gearboxes.

Picking an appropriate motor involves understanding a manufacturer's data sheets. A data sheet is usually given for the motor alone, and then another data sheet is supplied for the type of gearbox (with an assortment of reduction ratios) that will fit that motor. The gearbox specification can place constraints on the motor, such as for maximum allowable input speed or maximum deliverable output torque.

Actual data sheets for the small Escap motors (shown in Figure 7.1 of this chapter) are given in Figure 7.13 and Figure 7.14. The data are given here in tabular form instead of graph form, but the reader can reconstruct the graphs that were discussed earlier, (as

D.C. motor
escap® 16 M 11

Standard types available from stock			-210	-208	-207	-206
Measuring voltage		V	6	7.5	9	15
No-load speed		rpm	8400	7800	8300	8200
Stall torque		mNm	3	2.5	2.3	2.4
		oz-in	0.42	0.35	0.33	0.34
Power output		W	0.7	0.5	0.5	0.6
Av. no-load current		mA	7	5	4	2.5
Typical starting voltage		V	0.06	0.1	0.1	0.2
Max. continuous current		A	0.4	0.28	0.24	0.14
Max. recommended speed		rpm	12000	12000	12000	12000
Max. angular acceleration		10^3 rad/s^2	96	114	120	102
Back-EMF constant		V/1000 rpm	0.7	0.94	1.1	1.8
Rotor inductance		mH	0.5	0.8	1	3
Motor regulation R/k^2		10^3/Nms	300	330	380	350
Terminal resistance		ohm	13.4	27	39.5	105
Torque constant		mNm/A	6.7	9	10.2	17
		oz-in/A	0.949	1.28	1.44	2.41
Rotor inertia		kgm$^2 \cdot 10^{-7}$	0.7	0.56	0.5	0.6
Mechanical time constant		ms	20	18	19	21
Thermal time constant	rotor	s	6	5	4	4
	stator	s	380	380	380	380
Thermal resistance	rotor-body	°C/W	10	10	10	10
	body-ambient	°C/W	35	35	35	35

Figure 7.13. The Escap model 16M11-210 DC motor is a 6 volt (V) motor with a no-load speed of 8,400 revolutions per minute (rpm) and a stall torque of 3 milli-Newton-meters (mNm). (By courtesy Portescap, Inc.)

illustrated in Figure 7.10), since the major features, such as no-load speed, stall current and stall torque are given in these tables. The torque constant given in the table can be used to find the slope of the I-T curve, and the back-emf constant can be used to determine the slope of the ω-T curve. (Note that, if these constants are converted to the same units, they are equal.)

For instance, Figure 7.13 describes the performance of the motor by itself without a gearhead. Four models of this motor are available, each with a different winding and therefore intended to be run at a different voltage. The voltage for which the specifications are given is called the *measuring voltage* or sometimes the *rated voltage*. Thus, the 16M11-210 motor, when run at 6 V, will have a no-load speed of 8,400 rpm, a stall torque of 3×10^{-3} Nm, and a maximum possible output power of 0.7 W.

If the 16M11 motor is purchased with an attached gearhead, the part number for the gearmotor is M1616M11; its specifications, as shown in Figure 7.14, recommend that the -210 winding version be run at 5 V so that the no-load speed of the motor stays within the

D.C. gearmotor
escap® M 1616 M 11

Standard types available from stock		M 1616 M 11					
Max. recom. dynamic output torque	mNm (oz-in)	50 (7.1) at 20 rpm					
	mNm (oz-in)	30 (4.2) at 150 rpm					
Max. recom. static output torque	mNm (oz-in)	250 (35.4)					
Max. recommended input speed	rpm	7500					
Available reduction ratios		9	27	54	243	486	2190
				81		729	
Average efficiency		0.8	0.7	0.65	0.6	0.55	0.5
Nr. of geartrains / direction of rotation		2/=	3/≠	4/=	5/≠	6/=	7/≠
Length L	mm	37.1	38.6	40.1	41.6	43.1	44.6
Mass	g	28	29	29	30	31	32
Motor specifications		-210			-207		
Measuring voltage[1]	V	5			8		
No-load speed	rpm	7000			7200		
Stall torque	mNm (oz-in)	2.5 (0.354)			2.1 (0.297)		
Terminal resistance	ohm	13.4			39.5		
Torque constant	mNm/A (oz-in/A)	6.7 (0.949)			10.2 (1.44)		
Other motor characteristics	see page	17			17		
Gearmotor specifications							
Av. no-load current	mA	10			8		
Typical starting voltage	V	0.1			0.3		
Mechanical time constant	ms	21			19		

Figure 7.14. The Escap model M1616M11-210 gearhead motor should be driven at 5 V (instead of 6 V) in order to keep the no-load speed of the motor within the maximum allowable input speed of the gearbox. (By courtesy of Portescap, Inc.)

allowable input speed of the gearbox. The gearmotor with the 54:1 reduction will weigh 29 g, be 40 mm long, be 16 mm in diameter, and have an efficiency of 65%. The no-load speed will be:

$$N_o = \frac{7000\,\text{rpm}}{54} = 130\,\text{rpm}$$

and the stall torque will be:

$$T_S = (54)(2.5\,\text{mNm})(0.65) = 88\,\text{mNm}$$

Earlier, we showed that the maximum possible output power was:

$$P_{m,max} = \tfrac{1}{4}\omega_{max}T_{max}$$

We can calculate this maximum power by converting the no-load speed and the stall torque to the appropriate units. If we want to know how many radians per second are equal to 130 revolutions per minute, the easiest way to keep all the conversions straight is to set up the question this way:

$$?\frac{\text{rads}}{\text{sec}} = 130\frac{\text{rev}}{\text{min}}$$

Since multiplying the righthand side by 1 does not change the equality, we can multiply 130 rpm by identity relationships, converting revolutions to radians and minutes to seconds in such a way that the old units cancel out:

$$?\frac{\text{rads}}{\text{sec}} = 130\,\frac{\text{rev}}{\text{min}} \cdot \frac{2\pi\,\text{rads}}{\text{rev}} \cdot \frac{1\text{ min}}{60\,\text{sec}} = 13.6\frac{\text{rads}}{\text{sec}}$$

This gives:

$$P_{m,max} = \tfrac{1}{4}T_{max}\omega_{max} = \tfrac{1}{4}(88 \times 10^{-3}\,\text{Nm})(13.6\tfrac{\text{rads}}{\text{sec}}){=}0.3\,\text{W}$$

Escap motors are fairly high quality, and like many DC gearhead motors, can cost over \$100 each. Escap (actually, Portescap is the name of the company) sells old-inventory motors (catalog motors but ones that have sat on the shelves for too long to be sold as new) for a fraction of their original cost. Although the selection is limited, this source can be useful for hobbyists. Maxon, Micro Mo, Pittman, Inland, Globe, Canon, Copal, and Namiki are a few of the other numerous manufacturers that sell DC motors and have readily available catalogs with specification sheets. Surplus dealers often buy out remains of original equipment manufacturers' (OEMs) unused motors and sell them at significantly reduced costs. Dealers such as Burden's Surplus Center, Herbach and Rademan, America's Hobby Center, Edmund Scientific, Sheldon's Hobbies, Stock Drive Products, and Tower Hobbies sell wide assortments of smaller, cheaper DC gearhead motors.

Most of the low-cost permanent magnet DC motors, such as those found in toys, are made by one company–Mabuchi. Mabuchi produces over 3 million motors a day and sells them in lots of 5,000 or more. They make strictly stand-alone motors, not gearhead motors, but sell them to OEM manufacturers who then incorporate motors into toys, model airplanes, and the like. Typically, a toy manufacturer will use the molding of the toy itself to be the gearbox for the plastic geartrain they add to the motor so it is not always convenient to extract the motor and build it into your robot.

Model airplane servo motors, on the other hand, are very modular and convenient for this purpose. While most model airplane

servos continue to be high-priced, mass production of the most con mon models has led to lower prices for servo motors. Futaba, Roy Products Corporation and Airtronics are a few of the manufacture of these servo motors. Catalogs from hobby stores, such as Tow Hobbies and Sheldon's Hobbies, list a wide range of models. Highe quality servos with metal gears and ball bearings are available, als

Servo motor data sheets (which are typically printed on the bac of the packages) look different from the data sheets for DC gearhea motors. Servo motors usually run from a 5 V supply. For instanc for the Royal Titan Maxi Servo, the specifications are described th way:

Royal Titan Maxi Servo

Weight	3.7 oz.
Output Torque	112 oz.-in.
Current Drain	8 mA
Transit Time	$\frac{0.22\text{sec}}{60^\circ}$

A *transit time* (in $\frac{\text{sec}}{\text{deg}}$) is given instead of a no-load speed (i rpm) because the integrated circuit servos the motor to a specifie position and it never spins all the way around. However, if the serv is stripped down to being just a DC gearhead motor (potentiome ter, limit stops, and integrated circuit removed), this transit time i equivalent to the no-load speed. The output torque listed above i simply the stall torque. Converting to proper units to find powe output:

$$?\frac{\text{rads}}{\text{sec}} = \frac{60^\circ}{0.22\text{sec}} \cdot \frac{2\pi\,\text{rads}}{360^\circ} = 4.8\frac{\text{rads}}{\text{sec}} = 46\,\text{rpm}$$

$$?\text{Nm} = 112\,\text{oz.-in.} \cdot \frac{\text{lb.}}{16\,\text{oz.}} \cdot \frac{1\,N}{0.225\,1\,\text{lb.}} \cdot \frac{2.54\,\text{cm}}{1\,\text{in.}} \cdot \frac{1\,\text{m}}{100\,\text{cm}} = 0.79\,\text{Nm}$$

The maximum possible power then is:

$$P_{m,max} = \tfrac{1}{4}T_{max}\omega_{max} = \tfrac{1}{4}(0.79\,\text{Nm})(4.8\tfrac{\text{rads}}{\text{sec}}) = 0.95\,\text{W}$$

which is 3.2 times as large as the earlier Escap motor — but then this is a larger motor. To compare weights, we convert to grams:

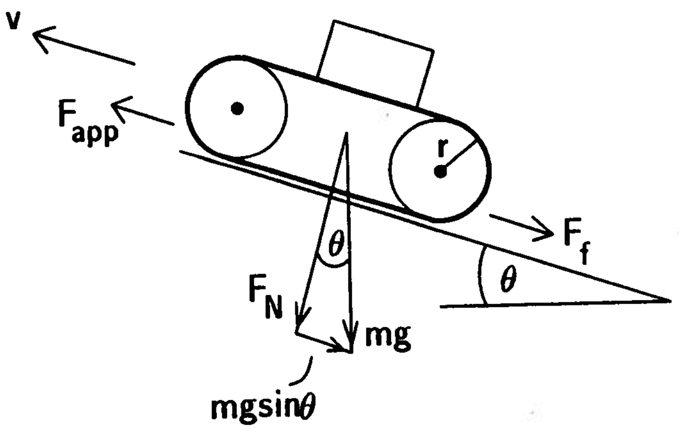


Figure 7.15. This free-body diagram of a tracked-drive Rug Warrior illustrates the forces acting on the vehicle as it climbs a hill. Use of this diagram helps to determine the maximum torques that the robot's motors will be required to deliver.

$$?\text{g} = 3.7\,\text{oz.} \cdot \frac{28\,\text{g}}{1\,\text{oz.}} = 104\,\text{g}$$

The Royal Titan servo motor then is about 3.6 times heavier than the 29 g Escap DC gearhead motor discussed previously. It turns out, however, that the Royal Titan with the potentiometer and circuit board removed, leaving essentially a comparable DC gearhead motor, weighs only 78 g. This seems to make sense, as the Royal Titan gears are plastic and the Escap gears are metal.

7.6 Motors for Rug Warrior

7.6.1 A Vehicle Model

In order to get a rough idea of how much power the motors for Rug Warrior must be able to deliver, we can sketch the scenario illustrated in Figure 7.15. Assume that Rug Warrior uses a differential drive mechanism (two motors) and needs to climb a ramp of angle θ at constant velocity, v. The free-body diagram makes explicit the forces

Because the vehicle moves at constant velocity, there must be net force on the car. That is, since:

$$F = ma$$

and the acceleration, a, is 0 (the car moves at constant velocit the net force F must be 0. This means that the applied force, F_{a} from the wheels acting in the direction up the hill must balance t forces down the hill resisting that force. These resisting forces a the friction force and the force that is the component of the vehicl weight acting in the direction down the hill. Thus:

$$F_{app} = F_f + F_W$$

where F_f is equal to the coefficient of friction, μ, times the norm force, F_N:

$$F_f = \mu F_N = \mu mg\cos\theta$$

and F_W is $mg\sin\theta$ (where mg, mass times the acceleration due gravity, is just the weight of the robot). This leaves:

$$F_{app} = \mu mg\cos\theta + mg\sin\theta$$

The power required from the motors is the product of the for that needs to be applied by the wheels times the velocity, v, t robot travels up the hill:

$$P_m = F_{app}v$$

where each motor must supply half that power, as Rug Warrior h two motors.

The torque and speed requirements of each motor can be calc lated from:

$$\frac{P}{2} = T\omega \quad \text{and} \quad \omega = \frac{v}{r}$$

where r is the radius of the wheel.

The range and the running time of the robot are dependent up the battery pack, since power is the rate of energy usage. If t battery has E joules of energy, then the battery lifetime t will b

$$t = \frac{E}{P}$$

The range of distance, D, the robot can travel will be: constrained by

$$D = vt$$

Typically, battery capacity is not given in joules but in units of ampere-hours. To find the energy contained in a battery pack, we must multiply the capacity rating in ampere-hours by the nominal voltage rating of the battery. (Recall that 1 joule equals 1 coulomb-volt and 1 ampere equals 1 coulomb per second.) For instance, suppose a 3 V battery has a 1,300 milli-ampere-hours (mAh) capacity. How many joules does it contain?

$$?\mathrm{J}=3\,\mathrm{V}\cdot 1300\times 10^{-3}\,\mathrm{Ah}\cdot \frac{\frac{\mathrm{C}}{\cancel{\mathrm{sec}}}}{\mathrm{A}}\cdot \frac{3600\mathrm{sec}}{\mathrm{h}} = 14,040\,\mathrm{CV} = 14{,}040\,\mathrm{J}$$

7.6.2 Selecting a Motor

The model we just described for Rug Warrior is hardly realistic. We certainly do not expect that our robot will be climbing up a ramp forever. Rather, because reality is so complicated (e.g., uneven terrain, stop-and-go crises, unknown coefficients of friction, accidents with chair legs, etc.), we use this model simply to attempt to size the peak power requirements.

Let's say that our goal is for Rug Warrior to weigh under 1.5 pounds, which is roughly 650 g. Furthermore, assume that we would like our robot to climb a 30 degree grade at a steady half foot per second, which is $\frac{0.15\,\mathrm{m}}{\mathrm{sec}}$. We will use two motors and a tank-drive locomotion system. Picking a value for μ is a way of trying to account for slippage and friction from the treads and the like. It is not clear what this coefficient of friction will be, but we can make some assumption and pad our result by oversizing the motors at the end. Let's pick μ to be 0.3. The power required then is:

$$P_m = F_{app}v = mg(\mu\cos\theta + \sin\theta)v$$

$$[illegible] \ 30°)(0.15\ \frac{\mathrm{m}}{}) = 0.73\,\mathrm{W}$$

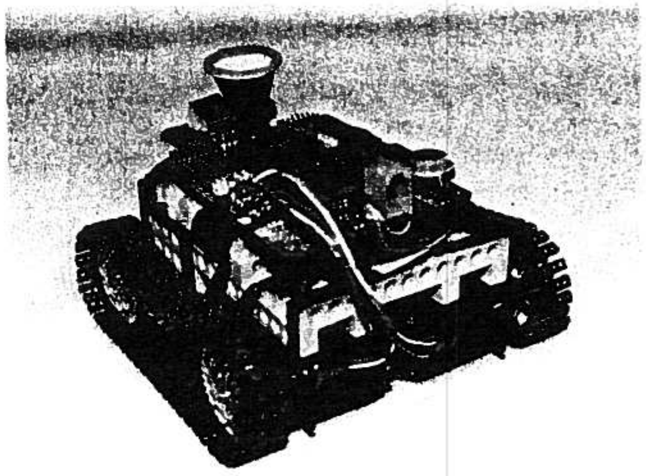


Figure 7.16. The easiest way to build a Rug Warrior is to start with mode airplane servo motors; add LEGO parts for bearings, axles, and treads; and thei place the batteries, electronics, and sensors on top.

We want to oversize our motors quite a bit, both because ther are so many unknowns and because the maximum efficiency poin is at a much lower torque than the maximum power point. If w multiply our power requirement by a whopping factor of 3, tha would give:

$$P_m = 2.1\,\mathrm{W} \qquad \text{or} \qquad \frac{P_m}{2} \cong 1\,\mathrm{W}$$

7.6.3 Converting Servo Motors

What we have chosen, as we mentioned earlier, is to use model air plane servo motors. We recommend these motors for the Rug War rior project of this book because they are fairly inexpensive an easy enough to modify. Although servo motors are not as cheap a toy motors, the fact that they come with gearboxes already built i means that we need not bother with machining a custom gearbox.

Figure 7.16 illustrates the tank-tread version of Rug Warrior tha we built using two Royal Titan Maxi Servos, which cost $25 each LEGO gears for wheels, LEGO tracks for tank treads, and LEGC axles and blocks for bearings and chassis. The PC board on top i

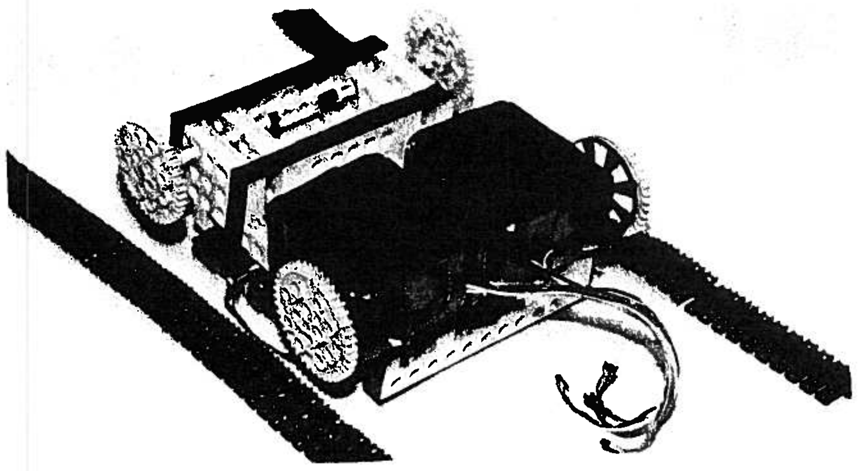


Figure 7.17. The underside of Rug Warrior contains two servo motors taped to the chassis, and LEGO gears mounted on the motor shafts for wheels. LEGO tracks are then used to make tank treads.

3.4"×4.5" and contains an MC68HC11A0, with the 10 sensors and the accompanying control electronics.

The tank drive is made up of two motors, connected to the back wheels in a differential fashion. The front wheels are passive, each having its own axle and bearing. The tank treads are wrapped around from back wheels to front wheels, so the robot can pivot in place.

Figure 7.17 is a view of Rug Warrior from the underside. The two black boxes are the servos. Attached to each is a LEGO gear for a wheel. The gear acts to mesh easily with tracks also supplied by LEGO. It is possible to build a sturdier and lighter-weight chassis, perhaps something made from aluminum sheet metal using a sheet metal bender and a punch for forming sides and placing holes. Real ball bearings and ground shafts could be used for the front wheels (obtainable from suppliers such as Berg, Small Parts Inc., etc.), but it turns out that ball bearings can cost as much as the MC68HC11A0 computer chip! Instead, we elected to use the LEGO building system, not just for gears and tracks but to continue with it for front wheel axles and bearings, as the axles that come with LEGO are

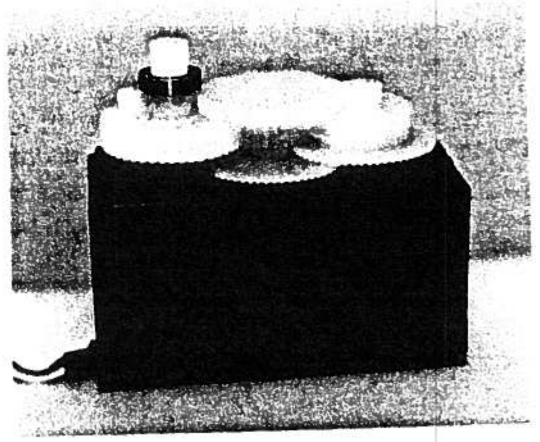


Figure 7.18. Some servo motors are easier to convert to continuous rotation than others. The gearhead of a Royal Titan Maxi Servo is shown here. The leftmost gear is above the potentiometer, and the ball bearing ring is mounted on its top for support of the output shaft. A plastic limit stop is molded onto the gear just below and to the left of the ball bearing.

bricks that we use for the chassis. We used double-sticky tape or black electrical tape to hold the chassis together.

To build this propulsion system for Rug Warrior, first modify the servos so that they can spin all the way around. Figure 7.18 shows the gearhead portion of the Maxi Servo; it has four stages of reduction for a 143:1 geardown. The motor shaft is at the right, the potentiometer shaft is at the left (the motor and potentiometer are below, inside the case), and the third shaft is in the middle. The output power is taken off at the potentiometer shaft. A plastic nib, molded onto the gear there, prevents the shaft from turning multiple revolutions. Above that nib is a metal ring, which is the ball bearing that supports the load.

Next, cut that plastic nib off. A pair of dikes (i.e., diagonal wire cutters) will work fine for the job. Then take that gear off and remove a plastic inset from its underside, which the potentiometer shaft's flat is held against. Not all servo motors have this feature of the removable inset. Some have the inset molded into the gear and have the gear turn directly on the potentiometer's shaft, which means it is not possible to easily make it continuously revolvable.

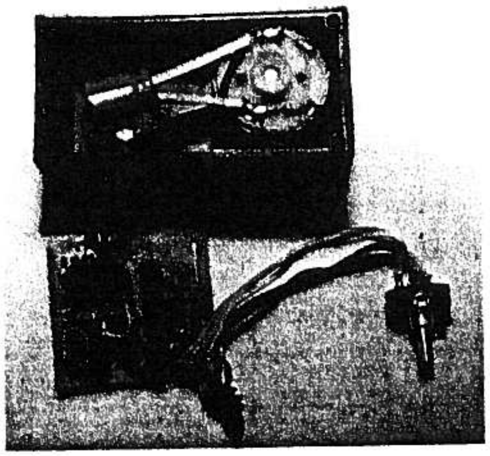


Figure 7.19. A bottom view of the servo in the previous picture shows a Mabuchi motor is in the righthand portion of the casing. The lefthand portion holds the potentiometer and a small circuit board containing an integrated circuit for servo control.

The Royal Titan servos have the removable inset and also have the gear resting on a bushing around the pot's shaft, which means you can actually remove the potentiometer completely. This brings us to the next step; removing the potentiometer.

Figure 7.19 is a view from the underside of the servo motor, with the cover removed and the potentiometer and servo circuit pulled out. Clip the wires for your motor, removing the circuit board. Take out the potentiometer by removing the screw holding it in place. Note the motor on the right. It is a Mabuchi motor and comes equipped with a capacitor across its leads and two resistors to ground to suppress noise spikes from the motor. Desolder the remains of the wires from the servo circuit, and solder on two new wires to the two terminals of the motor. Replace the cover over the gears, making sure the shafts sit properly in their holes. Try hooking a power supply or a battery pack up to two motor leads. The motor should spin continuously. Reversing the polarity of the applied voltage should reverse the direction of spin.

Adding wheels to a servo motor is convenient because servo motors come with an assortment of attachments (plates, levers, star- [illegible] mounting brackets, etc.) that are designed to fit snugly

onto the output shaft. Figure 7.20 illustrates a servo motor wi
the lever attachment. A simple way to mount the LEGO gear is
use the circular plate attachment (instead of the lever attachmen
which is roughly the same size as the gear; sand off any small rid
on the plate and/or the gear and glue them together.

It may seem odd to actually throw away a few components fr
a servo motor and still wind up with the lowest-cost route to a I
gearhead motor. Such are the benefits of mass markets. We will
a MC68HC11A0 and some power electronics (in a form known
an H-bridge) to drive the motors for steering Rug Warrior's trea
However, first let us digress a moment to explain how and where
unmodified servo motor would normally be used.

7.6.4 Unmodified Servo Motors

Typically, a radio-controlled model airplane servo motor is used
adjust a control surface on a wing of a model airplane to a certa
position. The integrated circuit and potentiometer are used to i
plement a closed-loop position control system. The radio sends wh
is known as a *pulse-code modulated signal* to a receiver on the mo
plane. As stated earlier, of the three wires emanating from the ser
motor, one is for power, one is for ground, and one is connected
this pulse-code modulated signal. Figure 7.20 illustrates the proto
for commanding the servo to a given position.

Basically, a servo motor expects a train of pulses of vary
widths. These pulses are repeated at a given period, typically s
to 20 ms. The width of the pulse is the code that signifies to wh
position the shaft should turn. The center position is usually
tained with 1.3 ms wide pulses, while pulse widths varying from 0
milliseconds (ms) to 1.7 ms will command positions all the way
the right and all the way to the left, respectively.

These position servo motors can be very useful for robot acce
sories (such as fingers, grippers, legs, and squirt guns) where t
range of motion does not require continuous revolution. For conti
uous motion, we described how to modify the servo and reduce it
a simple DC gearhead motor by throwing away the control circu
and power electronics that come with it and adding our own. Ho
ever, there is a way to use these motors as continuous revolution D

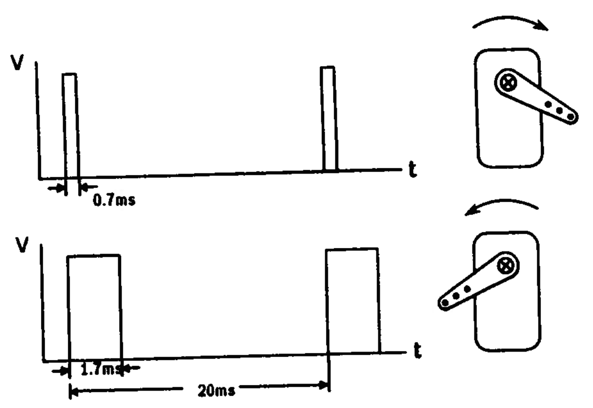


Figure 7.20. An unmodified servo is a three-wire device that takes power, ground, and a pulse-code modulated signal, such as the one shown above. Wider or thinner pulses tell the servo to move to a designated position, either clockwise or counterclockwise from center.

gearhead motors without having to add our own H-bridges and control electronics. The trick is to remove the inset in the plastic gear as before, which affixes itself to the flat of the potentiometer's shaft, but do not actually remove the potentiometer. Set the potentiometer to its central position. Now the gears will turn continuously but the potentiometer will never move. With this configuration, if we send the motor a pulse-code modulated signal to move all the way to the right, the motor will try to comply, never get any feedback, and never stop. Similarly, a pulse-code modulated signal to move to a position to the left will cause continuous rotation all the way to the left. This is an elegant trick (hack, to use the proper term) but we do not pursue it any further for Rug Warrior, because we want to explain how to attack the more common problem of driving a regular DC motor in general, and how to implement a servo loop.

7.7 Interfacing Motors

A microprocessor cannot drive a motor directly, since it cannot supply enough current. Instead, there must be some interface circuitry

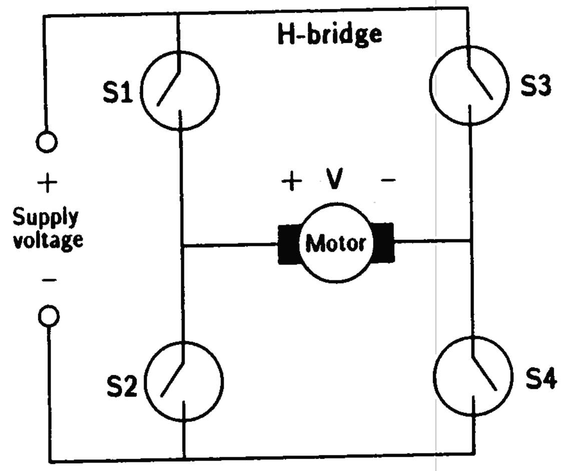


Figure 7.21. A circuit topology known as an H-bridge is used to control a motor. Four switches are controlled by a microprocessor and determine the direction in which current is allowed to pass through the motor. Changing the direction of the current changes the direction of the motor rotation.

only the control signals derive from the microprocessor. This interface circuitry can be implemented in a variety of technologies, such as relays, bipolar transistors, power MOSFETs (metal oxide semiconductor field effect transistors), and motor-driver integrated circuits. In all technologies, however, the basic topology of the circuit is usually the same. This circuit is known as an *H-bridge* and merely consists of four switches connected in the topology of an H, where the motor terminals form the crossbar of the H, as shown in Figure 7.21. You can imagine the abstraction of each switch as being implemented by either relays or transistors, where the power is supplied by the battery and the control signals by the microprocessor.

7.7.1 H-Bridges

In an H-bridge, the switches are opened and closed in a manner so as to put a voltage of one polarity across the motor for current to flow through it in one direction (setting up magnetic fields and causing it to turn) or a voltage of the opposite polarity, causing current to flow through the motor in the opposite direction for reverse rotation

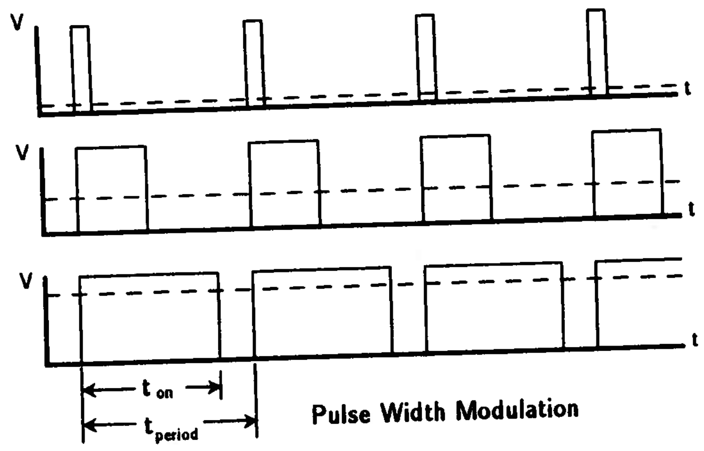


Figure 7.22. Pulse-width modulation of the voltage, by turning switches in the H-bridges on and off for various lengths of time, creates a different average voltage across the motor. Solid lines represent voltages applied when the switches are closed. Dotted lines represent the resulting average voltage applied across the motor.

For example, if switches S1 and S4 in Figure 7.21 are closed while switches S2 and S3 are open, current will flow from left to right in the motor. When switches S2 and S3 are closed and switches S1 and S4 are open, current will flow from right to left, reversing the motor. If the terminals are floating, the motor will freewheel, and if the terminals are shorted, the motor will brake.

To control the speed of the motor, the switches are opened and closed at different rates in order to apply different average voltages across the motor. This technique, called *pulse-width modulation*, is illustrated in Figure 7.22, where V is the voltage across the motor and t is time. For instance, if switches S1 and S4 are used for pulse-width modulation while switches S2 and S3 are left open, the voltage across the motor (as defined in Figure 7.21) will be equal to and of the same polarity as the supply voltage when S1 and S4 are closed and 0 V when they are open. The speed of a DC motor can be adjusted by changing the pulse-width ratio:

$$\text{Pulse-Width Ratio} = \frac{t_{on}}{t_{period}}.$$

Note that what we are describing here is different from pulse-code modulation for servo motors, discussed earlier. There, some "intelligence" was added so that the pulse width was a code signifying to what *position* the servo should move. Here, we are merely using varying pulse widths to create different average voltages across the motor to change its *speed*.

We mentioned before that the abstractions of switches in Figure 7.21 can be implemented in a number of ways. Relays can be used to turn motors on and off and reverse their directions as we saw in the TuteBot example, but relays are seldom used in pulse-width modulation speed controllers because they typically cannot switch quickly enough. Relays also tend to wear out. Solid-state switches, such as power bipolar transistors and power MOSFETs, are more convenient for pulse-width modulation schemes, and we will concentrate on these implementations here.

It is possible to design your own solid-state H-bridge controller, but there are also a number of single-chip solutions on the market. We chose one of these for Rug Warrior, and the anxious reader can skip ahead to the section on motor-driver power integrated circuits (see Section 7.7.4). However, if your particular project has requirements not available in a commercial H-bridge chip or if you are simply curious, the following sections give a bit of background on what is inside a motor-driver integrated circuit.

7.7.2 Switching Inductive Loads

Whether using solid-state switches or relays, problems arise wher switching inductive loads such as motors, as illustrated in Figure 7.23. We know that the voltage induced across an inductor is proportional to the rate of change of current through it:

$$v = L\frac{di}{dt}$$

If the current through an inductor has reached a steady state and is not changing, the voltage across it is 0 V and the inductor behaves like a straight piece of wire. Figure 7.23(a) shows what happens if that steady-state current is upset by the opening of a switch. Namely, the current cannot instantaneously go to 0 A sc a voltage, $v = L\frac{di}{dt}$, is induced in a direction opposing the flow o

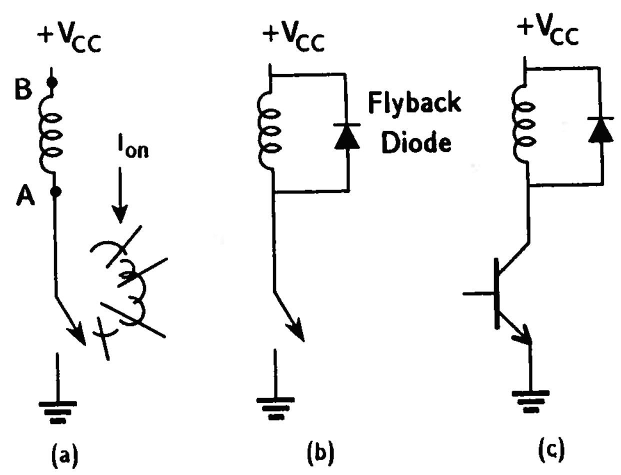


Figure 7.23. (a) The steady-state current through an inductor, I_{on}, cannot immediately go to 0 A when the switch is opened. The changing current induces a voltage across the inductor, making the potential at A greater than at B, causing the switch or relay to arc over. (b) Flyback diodes protect switches from blowing up. (c) Transistor switches must be protected in the same manner.

current. That is, the point marked A will be at a potential positive with respect to point B (which is at V_{cc}). Although the current does not change instantaneously when the switch is opened, it does change very quickly, and the faster the rate of change, the larger the induced voltage spike. Depending on the size of the inductor, the magnitude of the current, and how quickly the switch is opened, these voltage spikes can temporarily reach several hundred volts or more, enough to cause the switch to arc over and blow up.

The solution to this problem is to put what is known as a *flyback diode* in the reverse direction across the inductive load (Figure 7.23[b]) so that the voltage spike will forward bias the diode, creating a return path for the current. In this way, the power will "fly back" to the power supply.

Solid-state switches are just as susceptible to voltage spike destruction as mechanical switches, which is why transistor circuits switching inductive loads are usually shown with appropriate flyback diodes, as illustrated in Figure 7.23(c).

7.7.3 Power Electronics

As we discuss controlling motors from a microprocessor and the power electronics needed for the interface, we will talk about transistors used as switches. In Chapter 5 on sensors, we saw transistors, or collections of transistors in the form of op-amps, used as linear amplifiers to add gain to a circuit for amplifying small signals from sensors into larger signals understood by a microprocessor. The microphone circuit and the sonar circuit were examples. In addition to transistors used as linear amplifiers, we have also seen transistors used in another way: as CMOS (complementary metal oxide semiconductor) logic-gate switches. All the circuitry making up the internals of the 6811, its associated RAM and various discrete NAND gates and inverters, are simply composed of low-power, *n*-channel and *p*-channel MOSFET transistors used as switches. MOSFETs are similar to bipolar junction transistors in some sense, yet different in many ways. We will give some comparisons and contrasts between MOSFETs and transistors in a moment.

First, however, transistors can be classified another way, either as signal-level devices or as power devices. Transistors used for linear amplification of sensor signals or for logical manipulation of bits are concerned with processing information and are generally low-power devices. Power transistors, on the other hand, are capable of handling larger currents and voltages. They might be used as linear amplifiers in output stages of high-fidelity audio systems to drive speakers or they might be used as switches in H-bridges to pulse-width modulate motors requiring large currents. Power devices are typically larger than signal-level devices, as they require more silicon area for higher current-handling capability and larger packages for heat dissipation.

Semiconductors and Charge Carriers

Solid-state switches and power electronics are semiconductor devices. What is a *semiconductor* exactly, and why is silicon the material of choice for the semiconductor industry?

In a normal *conductor*, for instance, a metal such as aluminum, free electrons act as charge carriers and move in a direction toward a

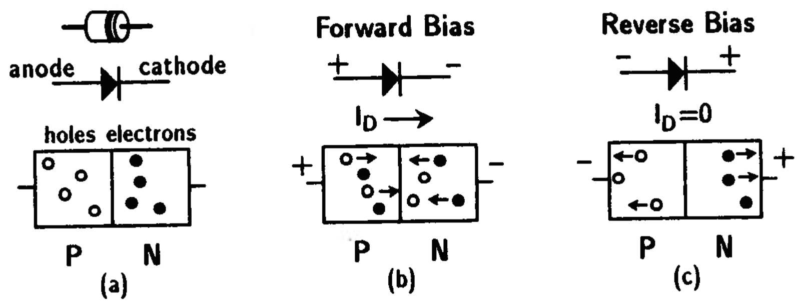


Figure 7.24. (a) A diode is simply a PN junction, where the *p*-type region is the anode and the *n*-type region is the cathode. (b) When forward biased, holes and electrons cross the junction, causing current to flow. (c) When reverse biased, no current flows.

positive potential. (Recall that *positive current* flows in the direction opposite to that of electron flow–so positive current moves away from a positive potential.) An *insulator* such as glass is the complement of a conductor, has no free charge carriers, and does not conduct current. A *semiconductor* on the other hand, lies somewhere in between. It is neither a perfect insulator nor a perfect conductor.

Because silicon has four valence electrons in its outer ring, it loves to bond covalently with other silicon atoms and create a perfect crystal lattice, much like diamond. Silicon is a semiconductor, and by adding various levels of impurity atoms, such as phosphorus or boron, silicon can become increasingly conductive. The reason that silicon is the material of choice for the semiconductor industry is that it is the only semiconductor that grows a native oxide layer. That is, when exposed to air, the silicon at the surface combines with oxygen to form a thin layer of silicon dioxide, essentially, a glass. Thus, in silicon processing, it is very convenient to create both conductors and insulators, a feature useful for patterning devices.

Another important characteristic of a semiconductor such as silicon is that two types of charge carriers are available to conduct current. Not only are electrons available to conduct current, but charge carriers called *holes* can also be developed. When impurity atoms of phosphorus are implanted in silicon, the five valence electrons in phosphorus's outer ring cause phosphorus atoms to bond into the

crystal silicon lattice, giving up one free electron as a charge carrier. Since electrons carry negative charges, regions of silicon doped with phosphorus are called *n*-type regions.

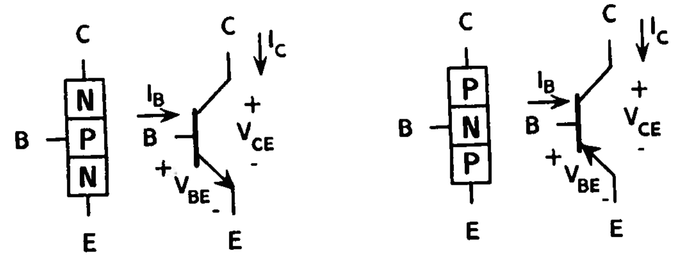
When impurity atoms of boron are added to single-crystal silicon, the three valence electrons of boron's outer ring cause boron atoms to bond into the silicon lattice, leaving a vacancy or hole. If electrons from other covalent bonds leave and fill these holes, the holes have essentially moved, creating a passage of positive charge carriers. Regions of silicon doped with boron are then termed *p-type regions.*

All the interesting behavior in silicon devices comes about at junctions of *n*-type and *p*-type regions. In fact, a diode is nothing more than a single PN junction, a junction of *p*-type and *n*-type material. Figure 7.24 illustrates how a diode works. When forward biased, holes and electrons each cross over the PN junction, attracted to the far terminals. They then mix and recombine, becoming neutral. New charge carriers are supplied by the terminals, and a continuous flow of both types of charge carriers is maintained, resulting in a steady-state current. When the PN junction is reverse biased, holes and electrons are each attracted to their nearby terminals and absorbed by them. The charge carriers move away from the junction, and the device becomes depleted of charge carriers. Thus, no current flows. This ability to allow current to flow or not flow, depending on the polarity of applied voltage, is the essential characteristic of a diode.

Bipolar Transistors

We saw that a diode is a single PN junction. A *bipolar junction transistor* is simply two PN junctions, back to back. There are two possible combinations of two PN junctions, *npn* or *pnp*, as shown in Figure 7.25.

Although simply having two PN junctions instead of one would seem only a minor addition at first glance, the realization and implementation of this technology has changed the world, for the third terminal on this dual-charge-carrier device allows the current to be *controlled.* The current can be either amplified when used in an analog fashion or switched when used in a digital manner.



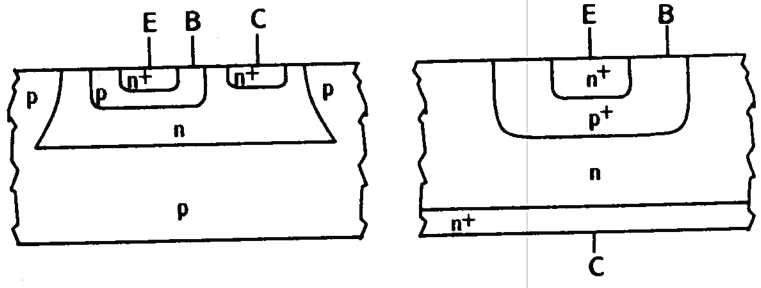
npn **bipolar transistor**
(a)

pnp **bipolar transistor**
(b)

Figure 7.25. Bipolar junction transistors are made up of two PN junctions, back to back. (a) In an *npn* bipolar transistor, the collector and emitter are *n*-type while the base is *p*-type. (b) In a *pnp* bipolar transistor, the collector and emitter are *p*-type while the base is *n*-type.

We mentioned before that transistors can be either signal-level devices or power devices. It turns out that, while these two types of transistors arise from the same semiconductor physics, they are fabricated differently. Figure 7.26 illustrates silicon cross-sections through a signal-level *npn* bipolar junction transistor and a power *npn* bipolar junction transistor. Plus signs on the *n* and *p* regions designate heavily doped areas (very conductive). A minus sign would denote lightly doped areas (slightly conductive).

In a signal-level bipolar device, all the terminals are patterned from the top side of the silicon wafer and the voltage between the base and emitter controls the flow of current from the collector to the emitter. For instance, in an *npn* device, when the base-emitter diode is forward biased, negative charge carriers "emitted" by the *n*-type emitter region travel toward the base but then are swept across into the collector region (before having a chance to get caught and recombine with any holes in the *p*-type base region) when a larger positive voltage is applied to the collector. Some small current must be supplied by the base to replenish any holes that did recombine with passing electrons, but this base current is much smaller than the collector current (which is why a bipolar transistor is a current amplifier). For signal-level devices, base, emitter, and collector all of the silicon wafer and the backside is not



signal-level ***npn*** **bipolar transistor**
(a)

power ***npn*** **bipolar transistor**
(b)

Figure 7.26. (a) In a signal-level device, all electrical terminals are on the top side of the silicon wafer and current flows along the surface, from collector to emitter. (b) In a power transistor, the backside is used for one of the electrical terminals and current flows vertically through the chip.

connected to anything. By having all terminals on the top side, it is easy to fabricate many different signal-level devices and interconnect them, allowing for very-large-scale integration (VLSI) for complex information-processing systems.

In a power device, on the other hand, the backside of the silicon wafer is used for one of the electrical terminals (the collector) and current flows vertically through the chip. Since power devices must handle more current and more heat, they are typically larger, often use backside connections and seldom integrate large numbers of different devices. More often, the tendency is to fabricate hundreds or thousands of vertical power transistors in parallel on one chip, creating in effect one very *big* transistor.

The cross-sections shown in Figure 7.26 are for *npn* bipolar transistors. The *pnp* bipolar transistors would have similar topologies but *p* regions would be replaced by *n* regions and vice versa. Since turning on a bipolar transistor requires forward biasing the base-emitter diode, turning on an *npn* version requires that the base be more positive than the emitter (at least 0.6 V more positive to be precise, as that is a diode's turn-on threshold). Conversely then, turning on a *pnp* version of a bipolar transistor requires that the base be 0.6 V more negative than the emitter.

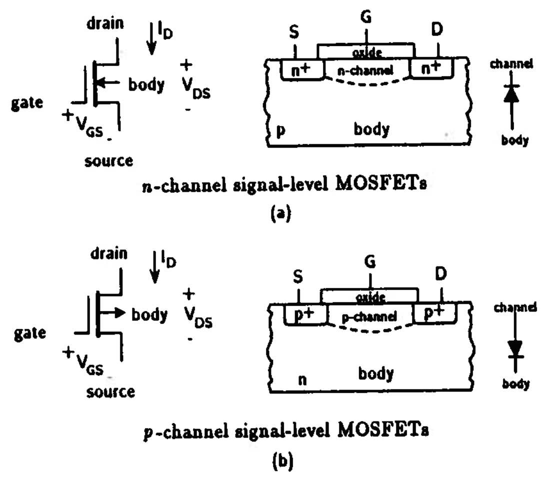


Figure 7.27. Signal-level MOSFETs also have all electrical terminals on the top side of the silicon wafer. (a) In an *n*-channel MOSFET, when the gate is positive with respect to the source, holes in the *p*-type body region move away from under the gate, leaving an *n*-type channel and allowing electron current to flow from drain to source. (b) In a *p*-channel MOSFET, when the gate is negative with respect to the source, electrons in the *n*-type body region move away from under the gate, leaving a *p*-type channel and allowing hole current to flow.

MOSFETs

Bipolar junction transistors rely on having two PN junctions in the main current path, which is why they are called *bipolar devices.* In contrast, a MOSFET has no PN junctions in the current path and is a *monopolar device.* Figure 7.27 illustrates symbols and cross-sections for *n*-channel and *p*-channel signal-level MOSFETs.

In the monopolar device, MOSFET junctions are fabricated to maintain separate regions of charge carriers when the device is off, but when an electric field is applied to the gate to turn on the device, the channel region separating two regions of like charge carriers is inverted making it the same "flavor" of charge carrier.

To be precise, we are speaking of enhancement-mode MOSFETs here (as opposed to depletion-mode MOSFETs) where, when the gate-source voltage is 0 V, the device is off. In this way, the entire current path is a region of the same type of majority carriers.

The two types of MOSFETs are then called *n-channel* and *p-channel* MOSFETs, and the three electrical terminals that corre-

spond in many ways to the base, emitter, and collector of bipolar transistors are called the *gate*, *source*, and *drain*, respectively.

Notice that signal-level MOSFETs are similar to signal-level bipolar transistors in that the backside again is not used for any of the three electrical terminals. However, in the schematic symbol for a MOSFET, the body terminal is explicitly drawn in, whereas in the bipolar schematic, it is omitted.

One reason for this is that the body forms a PN junction with the channel when the MOSFET is on. The arrow on the body terminal connection is pointed in the direction that a diode's arrow points (from p to n), signifying the direction of the PN junction between the inverted channel and the body when the MOSFET is on.

Because of the formation of a diode when the channel is inverted, the body must be held at a voltage that will not allow it to conduct. The body can be tied to the source (as is done in a power MOSFET) or to a more negative voltage in the circuit for an *n*-channel MOSFET. (For a *p*-channel MOSFET, the body can be tied to a voltage more positive than the source.) Sometimes, schematics leave the body connection out and we must assume that it is tied to a voltage that will keep the body-channel diode from conducting. Note, however, that the arrow on the body terminal is the only way to distinguish whether the MOSFET is *n*-channel or *p*-channel.

The gate terminal in the schematic is drawn with a horizontal line extending from the source end of the gate. This is to clarify which end of the device is intended to act as the source and which end is intended to act as the drain. Actually, a signal-level MOSFET is symmetric and can be used reversibly (and often is used this way in analog multiplexors and pass transistors for memories). For this reason, some schematics use a symmetrical gate connection, where the gate terminal is midway between the drain and the source.

Another reason the body terminal is drawn explicitly is that, if a power device is fabricated instead of a signal-level device, the backside connection is used as the drain. The central body region is then connected to the source, and this connection creates another device, a *source-drain diode.* A power MOSFET then is not symmetric. Symbols and cross-sections for *n*-channel and *p*-channel power MOSFETs are illustrated in Figure 7.28.

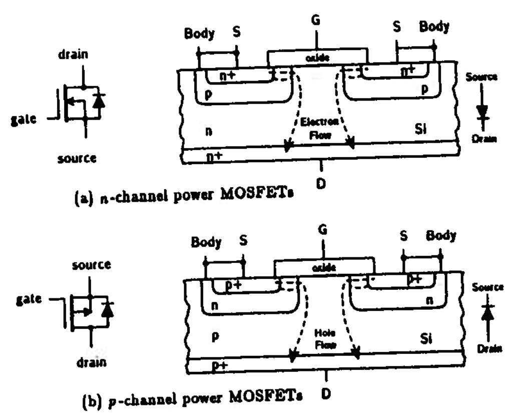


Figure 7.28. Power MOSFETs use the backside of the wafer as the drain. (a) When an n-channel power MOSFET is turned on, the p-type body region is inverted under the gate leaving a channel for *electron* current to flow vertically through the chip. (b) When a p-channel power MOSFET is turned on, *hole* current flows vertically through the chip.

Comparisons and Contrasts

Bipolar transistors and MOSFETs are similar in many respects, but a number of differences are worthy of note. First, though, let's take a moment to point out the general differences between n-type and p-type devices.

It turns out that the two types of charge carriers, holes and electrons, are not completely symmetric. Holes are not as mobile as electrons, and p-type devices, whether *pnp* bipolar transistors or p-channel MOSFETs, are never quite as good as n-type devices. In a bipolar transistor, a *pnp* device's high-frequency operation is poorer than a *npn* transistor's operation. In a MOSFET, a p-channel device does not exhibit as low on-resistance as an n-channel device. In fact, in the early days of MOSFETs, processes typically only gave the designer the option of having n-channel MOSFETs (often abbreviated as *NMOS* transistors). Later, when p-channel MOSFETs, or *PMOS* transistors, were introduced into the same process, the process became known as *CMOS* (complementary metal oxide semiconductor), since complementary n-type and p-type devices were then both available. Because p-type devices are poorer than n-type devices, this lack of performance has repercussions in the design of H-bridges for

One of the main differences between a MOSFET and a bipolar transistor is that a MOSFET is essentially a voltage-controlled device while a bipolar transistor is a current-controlled device. In a MOSFET, the gate oxide creates a capacitor between the gate and the source, so the steady-state gate current is 0 (although some charging and discharging currents flow when turning-on and turning-off the device). Since very little gate current is required, MOSFETs are fairly easily driven from microprocessors or CMOS logic gates.

In contrast, bipolar transistors are current-controlled devices. Instead of having a capacitor between the gate and source, as in a MOSFET, the bipolar transistor has a diode between the base and emitter. Once the base-emitter diode is forward biased, the collector current is controlled by the base current. The ratio of collector current to base current is the current gain, β:

$$\beta = \frac{I_C}{I_B}$$

For signal-level bipolar transistors, the current gain might be 100 or 200, but for power bipolar transistors carrying large numbers of amps, current gains are typically much lower, possibly on the order of 20 or 50.

Data sheets for devices under consideration should be checked for more specific numbers, but even so, current gains can differ widely from piece to piece (for the same part number of transistor) due to process variations between manufacturers. In general, though, power bipolar transistors require significant amounts of base current. Since these magnitudes of base current cannot be delivered directly from microprocessors or logic gates, another level of interface circuitry is often needed to drive the H-bridges that are driving the motors. In addition to the added complexity involved in the bias network the base current through the base resistor dissipates power (not to mention the power dissipated by the additional layer of interface circuitry).

In order to compare the efficiencies of bipolar power transistors and power MOSFETs for driving motors, return once again to the illustration of the H-bridge in Figure 7.21. For the ideal switches in that diagram, the voltage across the motor is always equal to the full magnitude of the supply voltage when opposite sets of switches

(S1 and S4, or S2 and S3) are closed. That is, there is no voltage drop across an ideal switch.

Real solid-state switches, however, do have finite voltage drops. The voltage drops associated with bipolar power transistors and power MOSFETs come about in different ways, however. In a power MOSFET, there are no PN junctions in the main current path from drain to source once the device has been turned on. Consequently, the only thing holding back charge carriers are factors such as their mobility, the width of the channel, and the like. These factors can be characterized as an effective resistance from drain to source. When the device is turned on as hard as possible, the channel becomes as wide as possible, giving the smallest on-resistance. This leads to the lowest voltage drop across the device, so this is the region where power MOSFETs should be run when switching motors. Figure 7.29(a) shows the I_D vs. V_{DS} characteristics for an n-channel power MOSFET.

The area to the left of the dotted line, where I_D increases with V_{DS}, is known as the *constant-resistance* or *linear region*. Typical MOSFETs have a threshold voltage on the order of 3.0 V-5.0 V, below which the MOSFET is cut off. To the right of the dotted line, for larger drain-source voltages and depending on V_{GS}, the channel becomes maximally opened and the current, I_D, reaches a saturation condition, where it remains constant even as V_{DS} is increased. If the gate voltage is high enough, usually around 10.0 V, the drain current stays in the constant-resistance region and the voltage drop from drain to source is minimal, as shown in the figure. This is the region in which a power MOSFET is run when switched to the "on" state, as the voltage drop, V_{DS}, across the switch is minimized.

The inverse of the slope of an I_D-V_{DS} curve in this linear region is the on-resistance ($\frac{1}{r_{DS}} = \frac{I_D}{V_{DS}}$) of a power MOSFET. The proper gate-to-source voltage should be chosen given the drain-source voltage and the desired current, so as to maintain the device biased in the constant-resistance region for the most efficient utilization of the power MOSFET.

The voltage drop across a turned-on bipolar power transistor comes about for a different reason. Whereas a turned-on power MOSFET has a continuous region of like charge carriers from drain to source, a bipolar transistor has two PN junctions in the current

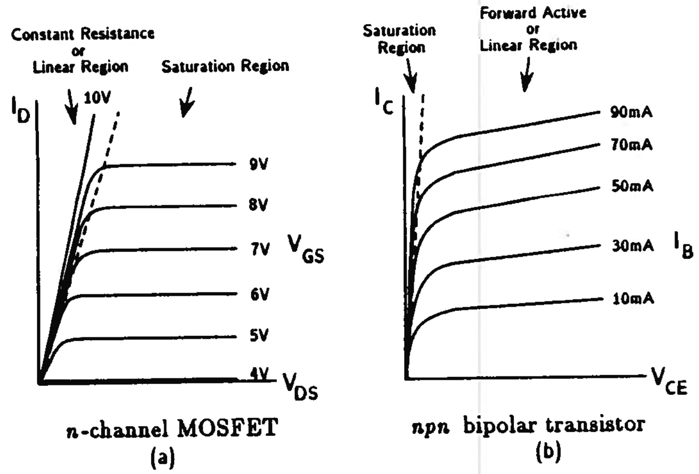


Figure 7.29. (a) An n-channel MOSFET shows these typical I_D-V_{DS} characteristics. When biased in the constant-resistance region, a MOSFET can be modeled as a resistor, where I_D varies linearly with V_{DS}. (b) An npn bipolar transistor is controlled by the value of the base current rather than the voltage as in case of a MOSFET.

path from collector to emitter, as was shown in Figure 7.25. In a turned-on bipolar transistor, the base-emitter diode is forward biased and the collector-emitter diode is reversed biased (at least, in the usual case of linear region operation). However, if the bipolar transistor is completely on, the collector potential should be close to that of the emitter potential (approaching the case of an ideal switch where there would be 0 V between collector and emitter). The closest a bipolar transistor can come is to have the collector-base diode no longer reverse biased but forward biased. With the base-emitter diode forward biased (transistor turned on) and the collector-base junction also forward biased, the bipolar transistor is in what is known as the *saturation region* of operation, where V_{CE} is almost constant and very small ($V_{CE} = V_{CE(SAT)}$) for any value of base current. Figure 7.29(b) illustrates the I_C versus V_{CE} characteristics for an npn bipolar transistor.

This saturation region is to the left of the dotted line in Figure 7.29(b) and is the region where a bipolar power transistor should be run when switched to the "on" state in order to provide the minimum voltage drop across the switch. The region to the right of the dotted line is known as the bipolar transistor's *forward active region*. The forward active or linear region is the region in which a bipolar transistor is used as a linear amplifier.

In comparing the graphs in Figure 7.29(a) and (b), note that the MOSFET is a voltage-controlled device, where I_D is determined by the value of V_{GS}, and the bipolar transistor is a current-controlled device, where I_C is determined by the value of I_B. Note, too, that the MOSFET's saturation region looks like what is called the linear region for a bipolar transistor, and the MOSFET's linear region looks like what is called the saturation region for the bipolar transistor. Again, this has to do with the MOSFET being a voltage-controlled device and the bipolar transistor being a current-controlled device and what parameter in each is actually being saturated.

Nevertheless, the point to be made is this: when transistors are used as switches, they should be biased in the regions to the left of the dotted lines in the figures so that they approach the function of ideal switches as closely as possible. That is, when an ideal switch is closed, it should have 0 V dropped across it. Solid-state switches cannot completely meet this goal, but when turned on hard enough, they can come as close as possible.

Any voltage drop appearing across a closed solid-state switch contributes to wasted power. For instance (referring again to Figure 7.21), if switches S1 and S4 are on and are implemented with bipolar transistors each having 0.3 V saturation voltage drop and if the supply voltage is 5.0 V, then only 4.4 V appears across the motor. Additionally, if the motors draws 500 milliamps (mA), then 2.2 W is delivered to the motor while 300 milliwatts (mW) is dissipated as heat in switches S1 and S4.

Deciding whether to choose power MOSFETs or power bipolar transistors when designing an H-bridge depends largely on which type of device will yield the most efficient solution. The answer depends on the power required by the motors and the choice of devices available. If MOSFET devices can be found that have low enough on-resistances and if, for the required current, they produce voltage drops less than saturation voltages of comparable bipolar devices, then power MOSFETs may be the right choice for designing an H-bridge

When comparing and contrasting bipolar transistors and MOSFETs, another characteristic to take into consideration is how each type of transistor responds to temperature increases, as running large amounts of current through a transistor causes it to heat up.

Bipolar transistors are subject to a condition known as *thermal runaway*. When current flows through the device, it gets warmer and the temperature rise affects the bipolar transistor in such a way that more current flows. With additional flow of current, the device gets even warmer and the problem escalates. *Thermal runaway* means that bipolar transistors cannot share current when configured in parallel. If one bipolar transistor has slightly more current running through it, it will heat up, allowing more current to flow; it will eventually hog all the current, resulting in thermal runaway.

In contrast, MOSFETs do not suffer from thermal runaway and lend themselves nicely to parallel configurations. The on-resistance of a MOSFET increases with temperature, providing a negative feedback effect. As more current flows through a MOSFET, its resistance increases and the current through the device decreases until a stable operating point is reached. Consequently, MOSFETs do not suffer from current hogging.

This feature is often taken advantage of in motor drives for electric vehicles and solar cars. Instead of purchasing one very large power MOSFET to switch current from an electric vehicle's battery to its engine, designers often buy the most economical power MOSFETs and place them in parallel. Up to 150 discrete devices are often paralleled in this way.

For a small mobile robot, however, where space is a primary concern, power MOSFETs do have some disadvantages. Because typical power MOSFETs need 8 V to 10 V for full-on gate drive, it may be inconvenient to drive a power MOSFET from a battery-powered robot using a single battery pack. Alkaline batteries come in 1.5 V cells and nickel-cadmium batteries come in 1.2 V cells; many of the design decisions for a small mobile robot revolve around the issues of battery pack selection, motors, and motor drivers, as the weight of the robot is primarily composed of these elements. If four alkaline cells are used as a 6 V power supply for the electronics, either more batteries or a charge pump must be provided to create the 8 V gate drive

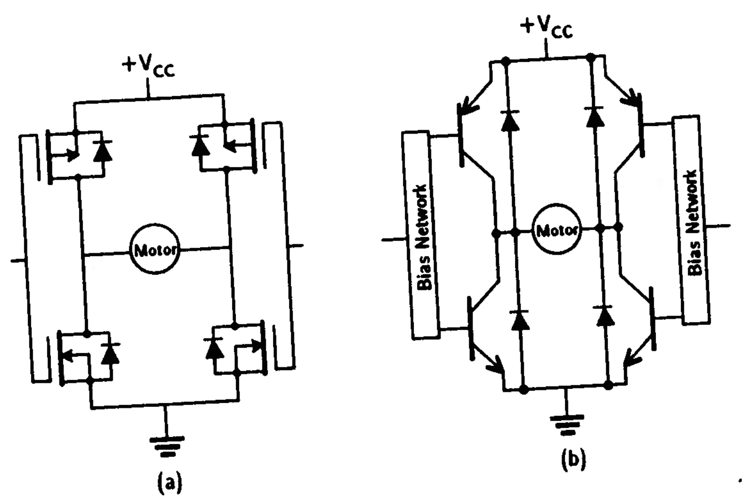


Figure 7.30. (a) A MOSFET implementation of an H-bridge, with *p*-channel devices on top and *n*-channel devices on bottom is shown here. (b) A bipolar transistor implementation of an H-bridge, with *pnp* devices on top and *npn* devices on bottom, requires more complex biasing circuitry to provide level shifting and base currents for the bipolar transistors.

One way around this problem is to use special low-threshold MOSFETs. These devices use very thin gate oxides to bring the turn-on voltages down to ranges from 1 V to 2 V. With threshold voltages that low, full-on gate drives can usually be achieved at 5 V. Such devices are called *logic-level* MOSFETs. Supertex makes a wide line of low-threshold MOSFETs. Motorola and International Rectifier also carry a variety of MOSFET devices.

H-Bridge Implementations

Whether MOSFETs or bipolar transistors are chosen to implement the H-bridge, the topologies are very similar. One convenient way to set up an H-bridge is to use *p*-type devices for the high-side switches and *n*-type devices for the low-side switches.

Figure 7.30 illustrates H-bridges in both bipolar and MOSFET technologies. If the gating signal on the left in each schematic is pulled low, the left-side bottom switches will be off and the left-side top switches will be on. If, at the same time, the right-side gat-
ulled high, the right-side bottom

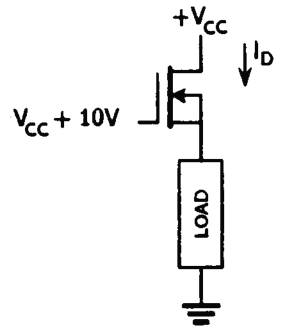


Figure 7.31. An *n*-channel MOSFET used as a high-side switch must have its gate voltage pulled higher than that of the positive supply in order to be on hard enough that the voltage drop between drain and source approaches 0 V.

switches will be on and the right-side top switches will be off. This configuration is exactly the scenario described in Figure 7.21 when switches S1 and S4 were on and switches S2 and S3 were off, allowing current to flow from left to right through the motor. Note that, in a MOSFET version of an H-bridge, flyback diodes do not have to be added discretely, as the built-in source-drain diodes provide the flyback function.

However, because *p*-type devices have higher on-resistances than *n*-type devices, it is possible to design more efficient H-bridges if *n*-type devices are also used for the high-side switches. The only problem with this design decision is that, if an *n*-type device is used for the high-side switch, the gating voltage to turn on the high-side switch must be pulled higher than that of the positive rail. For instance, in a MOSFET (see Figure 7.31), if an *n*-channel MOSFET is switching a load between the source and ground, the voltage at the source when the switch is on, should be very close to that of the positive supply rail. Since the gate turn-on voltage must be approximately 10 V higher than the source, the gate voltage must be $V_{CC} + 10$ V. Even if low-threshold devices are used, the gate voltage must be $V_{CC} + 5$ V, still requiring a separate power supply.

One solution to this problem is to add additional circuitry to the gate-drive network in the form of a charge pump. Charge pumps use switched capacitors to create voltages higher than the supply voltage. This type of design adds extra complexity to the input of the MOSFET implementation of an H-bridge, but fortunately, many manufacturers solve this problem by integrating all the required subsystems on a single motor-driver chip.

7.7.4 Motor-Driver-Power Integrated Circuits

Motor-driver-power integrated circuits (ICs) make it very convenient to interface motors to microprocessors. Typically motor-driver-power ICs also have circuitry that provides current-limiting and over-voltage protection. One single-chip solution is the MPC1710A motor driver from Motorola. This chip, whose block diagram is shown in Figure 7.32, uses an H-bridge composed of four n-channel MOSFETs. A level shifter and charge pump circuit are included on the chip to drive the high-side switches.

Three capacitors and an inverter are the only external components required to interface an MPC1710A to Rug Warrior's MC68HC11A0. We could use port D pin PD5 to set the forward or reverse direction of the motor and port A pin A5 to pulse-width modulate the enable input for speed control. The Motorola MPC1710A can deliver up to 1 A of current with a 0.4 Ω on-resistance when sourcing current and 0.2 Ω on-resistance when sinking current.

For the two motors on Rug Warrior, two MPC1710A chips would be needed, one for each motor. One motor could be controlled by pins PD5 and PA5 and the other motor by pins PD4 and PA6. The MPC1710A comes in a small 16-pin surface-mount package, which makes it very compact when used in a printed circuit board design but rather difficult to use when prototyping with Speedwire or Scotchflex wiring technologies. For this reason, on Rug Warrior, we chose to use a chip that would be more amenable for our readers, the SGS Thompson L293D.

The L293D was chosen because it comes in a normal 16-pin dual-inline package (DIP). This selection, which has two H-bridges on board, minimizes the parts count and delivers enough power for Rug Warrior's motors. The L293D, shown in Figure 7.33, uses a bipolar H-bridge instead of a MOSFET H-bridge. Again, all switches are made from n-type devices and a step-up circuit is incorporated on [illegible] Flyback diodes are integrated

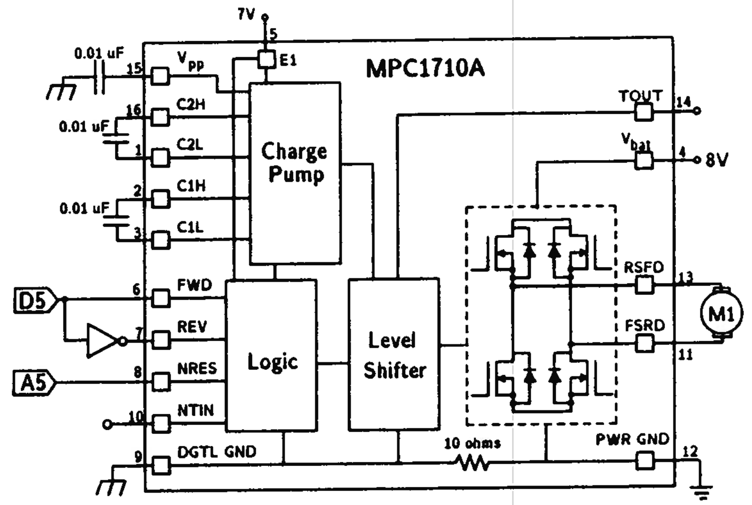


Figure 7.32. Two Motorola MPC1710As can be used to drive Rug Warrior' two motors. One MPC1710A is needed for the left motor and one MPC1710 is needed for the right. This surface-mount integrated circuit motor driver chi uses an H-bridge made from n-channel power MOSFETs.

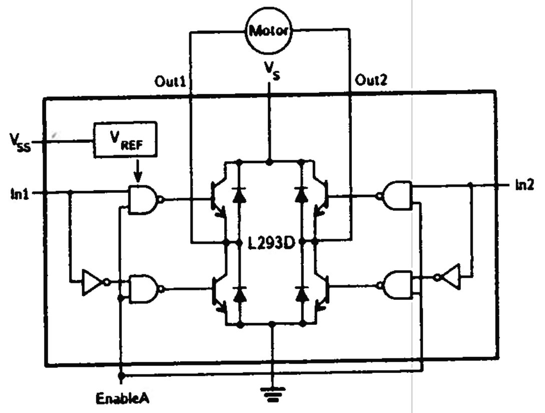


Figure 7.33. One motor-driver-power IC is an SGS Thompson L293D. Thi power IC incorporates a motor driver using an H-bridge made from bipolar tran sistors. While this illustration only shows one H-bridge, two full H-bridges ar actually incorporated in an L293D.

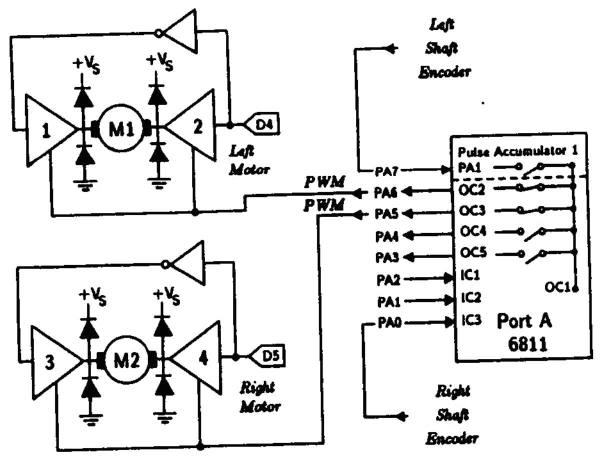


Figure 7.34. A single L293D chip is used to drive both of Rug Warrior's motors. The MC68HC11's port D pins PD4 and PD5 select forward and reverse for the left and right motors, respectively, while port A pins PA6 and PA5 pulse-width modulate the left and right enable pins. Note that OC1 here is used to also control OC2 and OC3.

on chip in this circuit. The L293D can deliver 600 mA to the motor, with a saturation voltage drop of 1.4 V when sourcing current and 1.2 V when sinking current.

Figure 7.34 illustrates how we have interfaced the L293D to Rug Warrior's 6811. The L293D has some on-chip logic that provides an Enable signal. In this way, the Inputs to the H-bridge can be used to set the direction of the motor and the Enable signal can be used for pulse-width modulation. We use port D pin PD5 to set the direction for the right motor. An inverter is used to set one side of the H-bridge to the opposite polarity voltage of the gating signal of the other side. This ensures that if switches S1 and S4, for instance, are on, then switches S2 and S3 will be off and vice versa. Note that this means that the motor is never actively braked. The H-bridge is pulse-width modulated by tying the right motor driver's Enable pin to port A pin PA5. The output compare function of PA5 is used to facilitate timing. One advantage of the L293D is that two full H-bridges are incorporated on chip, which means that only one

PD4 is used to set the direction of the left motor, and port A pin PA6 is tied to the Enable signal for pulse-width modulation.

Many other motor-driver-power integrated circuits are available. As mentioned in E, the place to begin searching is the *IC Master*. The *IC Master* lists integrated circuits both by part number and by function. Listings under "Motor Drivers" include a number of suppliers, such as Unitrode, Siemens, Motorola and International Rectifier, among others.

Another avenue to pursue is to purchase motor controllers for radio-controlled cars. These are often called *speed controllers*, which is a bit of a misnomer, since it is only the human who provides the speed control. However speed controllers do incorporate the power MOSFETs or power bipolar transistors in discrete H-bridges for driving larger motors. They are sold by Futaba, Tower Hobbies, and Sheldon's Hobbies, and are often advertised in radio-control hobbyist magazines.

7.8 Software for Driving Motors

The software for controlling Rug Warrior's motors must do two things. First, the software needs to control the speed of the robot in the manner desired by the programmer. For instance, a higher pulse-width ratio of voltage across the motor is needed to keep the robot moving up a ramp at one foot per second than would be required to make it move along a flat tile floor at one foot per second. To maintain a desired speed, regardless of terrain, means that the robot needs to count the number of pulses from one of the shaft encoders to see how fast the wheels are turning and then update the pulse width accordingly.

The second function that the software needs to perform is to make the two wheels actually revolve at the same speed so that the robot will move in a straight line. Recall that, in TuteBot, innate differences between the two motors caused TuteBot to move in an arc, even when the same voltage was applied to both motors. In that case, we simply added resistors in series with one motor until both motors went at the same speed. That analog solution was fine for TuteBot, but here, we implement a digital solution, since Rug

Warrior has a microprocessor right at hand. In this way, the solution is general, and if many Rug Warriors are manufactured, they do not all have to be individually tweaked with resistor trials. Again, the solution is to read the shaft encoders from each wheel and increase or decrease the speed of the right motor, say, to match its speed with that of the left motor.

7.8.1 Pulse-Width Modulation

The software we have configured for controlling Rug Warrior's motors takes advantage of timer-counter hardware associated with the MC68HC11A0's port A and succeeds in implementing a pulse-width modulation scheme without recourse to either polling or interrupts. Port A's eight pins have various output compare and input capture registers, as shown in Figure 7.34. Refer to the Motorola data books on the MC68HC11 for a more complete discussion than we will attempt here.

An output compare register can be set by the programmer so that, for instance, when the timer-counter's value matches the output compare register's value, a pin can be set high or an interrupt can be initiated. An input capture register has the opposite function. When a signal on a pin goes low for instance, the input capture register can store the value of the timer-counter register at the time that the event happened or initiate an interrupt.

Output Compare Registers

For pulse-width modulation, we will take advantage of the output compare registers associated with port A pins PA3-PA7, as shown in Figure 7.34. Pin PA7 happens to hold a dual role as either a pulse accumulator or as output compare register 1 (OC1). For Rug Warrior's right wheel, we have chosen to use PA5, which is associated with output compare register 3 (OC3) and for the left wheel, PA6, which is associated with output compare register 2 (OC2). We also take advantage of OC1 because it is a special output compare register in that it can control a given selection of the four other output compare registers. The closed connections between OC1 and OC2 and between OC1 and OC3 in Figure 7.34 illustrate how we intend

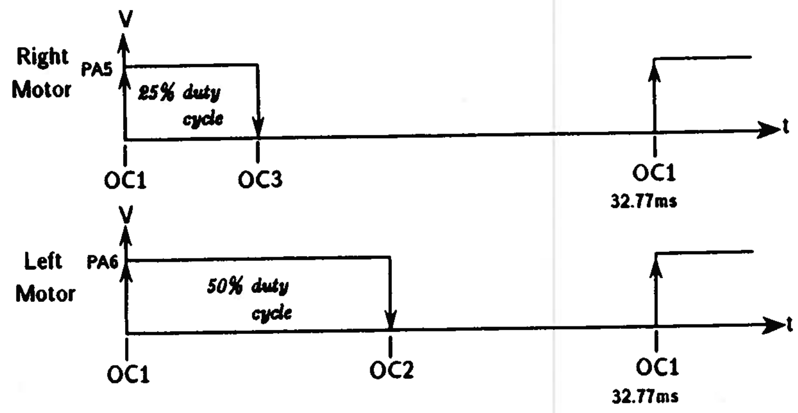
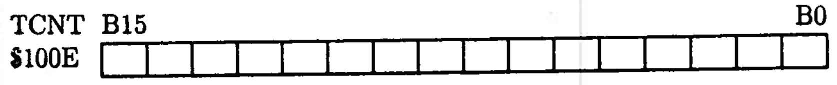


Figure 7.35. Pulse-width modulation can be conveniently accomplished using the MC68HC11A0's port A output compare registers. Here, we use three different output compare registers, where output compare register OC1 directs pins PA5 and PA6 to both go high at the beginning of each period. Output compare registers OC3 and OC2 each tell pins PA5 and PA6 when to go low, giving a programmable duty cycle for each motor.

Figure 7.35 illustrates the timing sequences for our algorithm that will be generated on PA5 and PA6 to implement pulse-width modulation.

The timer-counter itself is a 16-bit register, TCNT, where the high byte is at hex address $100E and the low byte is at $100F:



The timer-counter runs at a rate dependent upon Rug Warrior's crystal oscillator (and therefore the MC68HC11A0's E clock, which is on pin 5 of the MC68HC11A0 and can be checked with an oscilloscope). The E clock has a period of one-fourth that of the crystal oscillator frequency. TCNT is a free-running counter that starts at 0 when the MC68HC11A0 is reset and counts up to 2^{16}, which is 65,536 counts. The counter then overflows and starts again from 0. We use an 8 megahertz (MHz) crystal for Rug Warrior, giving the E clock a frequency of 2 MHz and a period of 0.5 microseconds (μs).

By default, the timer-counter counts at the same period as the E clock, but there is a way to prescale the timer-counter rate, which involves setting two bits in another register, TMSK2. The lowest two bits in the TMSK2 register, PR1 and PR0, are used to divide down the E clock for changing the rate at which the timer-counter runs.

TMSK2 $1024	Bit 7						Bit 0
	TOI	RTII	PAOVI	PAII	0	0	PR1	PR0

For our purposes, we will let the timer-counter run at its default setting and not bother with changing any values in TMSK2. This means that, after 2^{16} counts at $0.5\,\mu s$ per count, 32.77 milliseconds (ms) will have passed. We will use this standard overflow time as the period for pulse-width modulation, t_{period}, as was illustrated earlier in Figure 7.22.

Our plan is to create the pulse-width modulated signals for the left and right motors using waveforms generated by OC2 and OC3 associated with pins PA6 and PA5, as shown in Figure 7.35. In this case, we will use OC1 to set the bits high on both PA6 and PA5 when the timer-counter is at 0. We will use the OC2 and OC3 registers to clear the bits on both PA6 and PA5 when the value reached by the timer-counter matches the values stored in their 16-bit timer output compare registers, TOC2 and TOC3. So, for instance, if we want PA5 to have a 25% duty cycle, then we store 65,536 ÷ 4 = 16,348 in TOC3. If we want PA6 to have a 50% duty cycle, we store 65,536 ÷ 2 = 32,768 in TOC2:

TOC2 $1018	B15															B0
	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

TOC3 $101A	B15															B0
	0	0	1	1	1	1	1	1	1	1	0	1	1	1	0	0

To specify what action should be taken on PA5 and PA6 when the corresponding output compare registers match the timer-counter, we must set some values in another register, TCTL1. The appropri- … For the way we designed

OMx	OLx	Configuration
0	0	OCx Does Not Affect Pin
0	1	Toggle OCx Pin
1	0	Clear OCx Pin
1	1	Set OCx Pin

Figure 7.36. The four actions possible by any output compare pin are to nc change, to toggle, to go low, or to go high. Two bits in the TCTL1 register, th most significant bit (OMx) and the least significant bit (OLx), set the desire response for any successful output compare.

our algorithm in Figure 7.35, we want PA5 and PA6 to be set t 0 when OC3 and OC2 have successful output comparisons. To se this up, we store the two bits, %10 (which will make the pin go low) in TCTL1 in the locations associated with OC3 and OC2:

TCTL1 $1020	Bit 7							Bit 0
	OM2	OL2	OM3	OL3	OM4	OL4	OM5	OL5
	1	0	1	0	x	x	x	x

The x's in any bit position represent don't care's. With the fallin edge of the pulse configured (the signal transitioning from high tc low), now we just need to set up the OC1 rising-edge event (th transition of the signal from low to high). This is done by storing th value of time equal to 0 in TOC1, the 16-bit timer output compar 1 register:

TOC1 $1016	B15															B0
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

To configure the hardware so that OC1 will control PA5 and PA6, we set values in some auxiliary registers that control OC1. The output compare 1 mask (OC1M) register signifies which of the other four output compare registers OC1 will control. The high five bits in OC1M correspond bit for bit with a port A output pin. Therefore, we store the binary number %01100000 in OC1M to set up OC1 to control PA5 and PA6:

	Bit 7							Bit 0
OC1M $100C	OC1M7	OC1M6	OC1M5	OC1M4	OC1M3	-	-	-
	x	1	1	x	x	x	x	x

Once we have selected which pins will be active, we can program the action that we want to result when the timer-counter value matches the value of 0 in TOC1 by setting some values in the output compare 1 data (OC1D) register. By setting the bits corresponding to PA5 and PA6 to 1, whenever the timer-counter overflows and returns to the value 0, the PA5 and PA6 pins will be set to 1, which forms the rising edge of each pulse.

	Bit 7							Bit 0
OC1D $100D	OC1D7	OC1D6	OC1D5	OC1D4	OC1D3	-	-	-
	x	1	1	x	x	x	x	x

By using these built-in hardware features of the 6811, no interrupts or polling sequences are required to implement pulse-width modulation. We simply write to some registers in the timer-counter system and all actions on pins PA5 and PA6 take place in the background of other programs being run on the robot. The programmer merely writes new values to TOC2 and TOC3 when the speed has to be changed.

PWM Software Driver

The program below illustrates the **IC** code that implements this scenario of a 25% duty cycle signal asserted by pin PA5 and a 50% duty cycle signal asserted by pin PA6. This sequence will make one wheel rotate at half the speed of the other, causing Rug Warrior to move in an arc towards one side or the other.

```
int DDRD = 0x1009;   /* Port D data direction */
int OC1M = 0x100C;   /* Output Compare 1 Mask */
int OC1D = 0x100D;   /* Output Compare 1 Data */
int TOC1 = 0x1016;   /* Output Compare Timer 1, 16-bit reg */
int TOC2 = 0x1018;   /* Out Cmp Tmr 2, 16-bit reg (left motor) */
int TOC3 = 0x101A;   /* Out Cmp Tmr 3, 16-bit reg (right motor) */
int TCTL1 = 0x1020;  /* Timer Control 1, 8-bit reg */
```

```
/* motor_index:  0 => Left motor, 1 => Right motor */
int TOCx[2] = {TOC2,TOC3};          /* Index for timer register */
int sign[2] = {1,1};                /* Sign of rotation of motor */
int dir_mask[2] = {0b010000, 0b100000};/* Port D direction bit

/* Utility functions */
float abs(arg)                      /* Absolute value function */
{ if (arg < 0.0)
   return (- arg); else return arg; }

int get_sign(float val)             /* Find sign of argument */
{ if (val > 0.0)
   return 1; else return -1; }

/* Limit range of val */
float limit_range(float val, float low, float high)
{ if (val < low) return low;
   else if (val > high) return high;
   else return val; }

void init_pwm()                /* Initialize Pulse-Width Modulation *
{ poke(DDRD,0b110010);         /* D dir:  OUT 5,4,1; IN 3,2,0 */
  poke(OC1M,0b01100000);       /* Out Cmp 1 affects PA5 and PA6 */
  poke(OC1D,0b01100000);       /* Successful OC1 turns on PA5, PA6 */
  bit_set(TCTL1,0b10100000);   /* OC3 turns off PA5, OC2:  PA6 */
  pokeword(TOC1,0);            /* When TCNT = 0, OC1 fires */
  pokeword(TOC2,1);            /* Minimum on time for OC2 */
  pokeword(TOC3,1); }          /* Minimum on time for OC3 */


/* The sign is handled in a special way because */
/*  we have only a 1 channel encoder */
void pwm_motor(float vel, int motor_index)
{ if (sign[motor_index] > 0) /* Choose the direction of rotation
     bit_set(port_d, dir_mask[motor_index]);
  else
     bit_clear(port_d, dir_mask[motor_index]);
  vel = limit_range(vel, 1.0, 99.0); /* 1 ≤ duty fctr ≤ 99 */
  pokeword(TOCx[motor_index], (int) (655.36 * vel)); }

/* Top-level open-loop PWM command */
void move(float l_vel, float r_vel) /* Vel range [-100.0, 100.0]
{ sign[0] = get_sign(l_vel); /* Desired direction of rotation */
sign[1] = get_sign(r_vel);
pwm_motor(abs(l_vel), 0); /* Set pulse-width modulation cnst */
pwm_motor(abs(r_vel), 1); }
```

Now let's walk through this program. First, all the necessary registers are assigned and three data structures are created. The arrays TOCx[ ], sign[ ], and dir_mask[ ] are all two-element arrays. TOCx[ ] is an array whose first element is the address TOC2, where left-motor velocities are stored, and whose second element is the address TOC3, where right-motor velocities are stored. The array sign[ ] is an array whose elements are bits representing which direction the left and right motors are commanded to go. The array dir_mask[ ] is an array whose first element holds the mask for Port D, required to select the left motor, and whose second element holds the mask for Port D, required to select the right motor.

The next three functions also just lay the groundwork for the main part of this program. The functions abs(), get_sign() and limit_range() are functions that C does not happen to supply: abs() simply returns the absolute value of its argument; get_sign() returns the sign of its argument; and limit_range() returns a maximum or minimum value for its argument if it is out of range.

The actual pulse-width modulation of the motors is accomplished by the functions init_pwm(), pwm_motor(), and move(). The timer-counter hardware is set up and started by init_pwm(). The three pokeword() commands set an initial (small) pulse width.

To change the duty cycle, the calling routine uses the pwm_motor() function, which takes two arguments: a velocity command and a motor index. pwm_motor() then pokes the new velocity into the address, either TOC2 or TOC3, as specified by motor_index.

The function move() is the interface the programmer has for directing the robot. move() takes two arguments, a velocity for the left motor and a velocity for the right motor. These velocities should be given in the form of percentages of full speed. That is, they should be in the range [-100.0, 100.0]. A move(25.0, 50.0) command would make the left motor move at 25% of full speed and make the right motor move at 50% of full speed, causing the robot to arc to the left.

Setting up the pulse-width modulation scheme for each motor [illegible] to a few registers. Once

this has been done, the hardware associated with the timer-count system will run by itself - always setting pins PA5 and PA6 hig when the timer-counter reaches zero, always setting PA5 low whe the timer-counter reaches 16348, and always setting PA6 low whe the timer-counter reaches 32768. The central processing unit of tl MC68HC11A0 then is free to attend to other tasks, perhaps readir a sensor or calculating a new speed at which the robot should ru To change the speed, Rug Warrior's main program merely has t store new values in TOC2 and TOC3.

7.8.2 Feedback-Control Loops

The strategy we have just described for pulse-width modulating m tors is known as an *open-loop control scheme*. In open-loop contro there is no feedback from the motors, telling the robot's program ho fast the wheels are turning or how far the robot has gone. Rathe the motors are just given different commanded voltages. But de pending on terrain, surface obstacles, slippage in wheel contacts, c load on the robot, the commanded voltages do not necessarily impl particular speeds.

To implement a true velocity- or position-control algorithm, th robot needs sensors on the wheels, such as the shaft encoders men tioned earlier. Such feedback enables what are known as *closed-loo control algorithms*. Figure 7.37 illustrates the simple control loo we will use on Rug Warrior, called a *P-I controller*, for *proportiona integral controller*.

The basic idea of a control loop is to take the desired velocit command (such as one created in the way just described for ou pulse width modulation scheme), send that command to the motors see how fast the motors actually spin, and then measure that spee and compare it to the commanded speed. The difference is calle the error signal and it can be either positive or negative. There ar three error signals (marked e_1, e_2, and e_3) in the P-I control loop o Figure 7.37.

What makes a control loop a proportional controller or an inte gral controller depends on what computation the loop performs o the error signal. For instance, if the loop multiplies the error b some constant to produce a new command, then the controller is

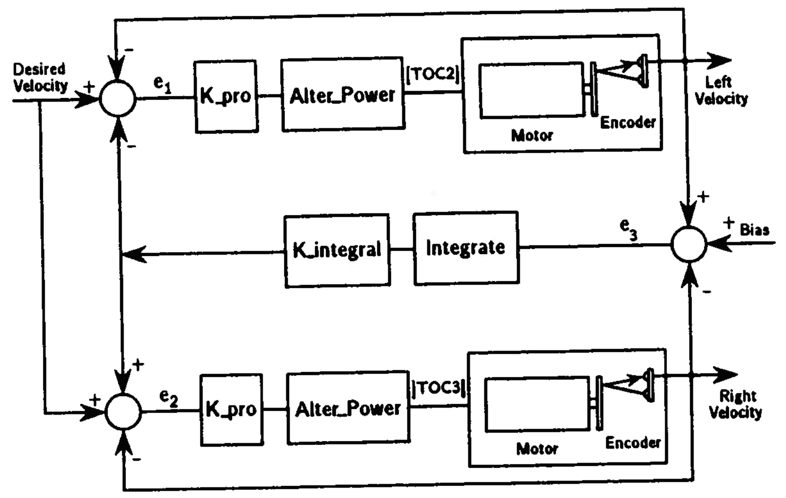


Figure 7.37. A simple proportional-integral control loop can be added in software to control the speed of the robot and to synchronize Rug Warrior's two wheels so that the robot will travel in a straight line.

proportional controller. In the controller shown in Figure 7.37, there are actually three separate feedback loops.

Imagine for a moment that the central feedback path is not there. The top loop, producing the error signal e_1 from the left motor, is identical in form to the bottom loop, producing the error signal e_2 from the right motor. Each of these loops is a proportional controller because the difference between the desired speed and the actual speed is multiplied by a constant, K_pro, and fed back to the motor to adjust the motor speed. If the actual speed is less than the desired speed, the difference is positive (as defined by the assignment of plus and minus signs on the feedback arrows), and if K_pro is also positive, a larger desired velocity is next sent to the motor. If the actual speed is greater than the desired speed, the error signal is negative and a smaller command is sent to the motor, slowing it down. This loop repeats until the error signal is small enough so that the motor is considered controlled at its desired speed.

We mentioned earlier that the software controlling Rug Warrior's motors should implement two things. The first was that Rug Warrior should be able to maintain a desired velocity (whether climbing

a ramp or traversing a flat space, for instance). The two separate proportional controllers just described for each motor essentially fulfill that requirement. We said that the second responsibility of the software would be to oversee that the two wheels would be slaved to each other. That is, if the robot were commanded to go straight, the velocities of the two wheels would be synchronized so that the robot really *would* go straight. This feature is implemented via the central feedback path of Figure 7.37, the integral controller.

The integral controller looks at the actual speeds of both motors and compares them. The difference between the two actual speeds is the error e_3, as can be seen at the right in Figure 7.37 where e_3 = left velocity − right velocity + bias. The bias term is used for inputting the turn command. While the bias is 0, the error signal only changes over time if the robot is not going straight but swerving one way or the other. An integral controller integrates, or sums, the error signal over time, multiplies this sum by a constant, K_integral; and feeds that new command back into the proportional-control loops for each motor. In this way, one motor is sped up while the other is slowed down until they each reach speeds sufficiently close together.

In Figure 7.38, we focus on just the upper third of the P-I controller diagram, the proportional-control loop for the left motor. We can implement the computation that this illustration conveys with a few simple **IC** routines. First, the data structure we will rely on for the input desired velocity is the function move() described earlier, which we constructed for our open-loop PWM controller. If the desired velocity was commanded by calling move(25.0, 50.0), this piece of the control system would try to servo the left motor such that every time **get_left_vel()** was called, it would return 25 counts.

To assist in the computations necessary to control the motor, we will create a variable, K_pro, and the function alter_power(). The input to alter_power is computed by left_error() which is the product of the difference between the desired velocity and the actual velocity (both in units of clicks per interval) and some constant, K_pro. alter_power() just calls pwm_motor() with this new velocity command. The main program then waits for a time interval, calls **get_left_vel()** again, and repeats the adjustment continuously. In this way, if the robot is servoing along a flat floor at one speed but then approaches a ramp and begins to climb, more power will be supplied to the mo-

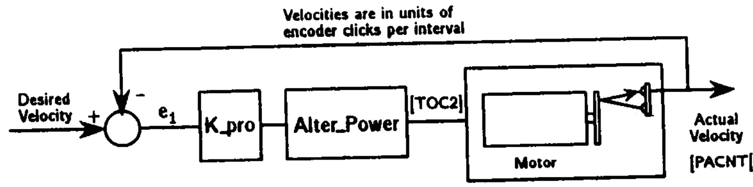


Figure 7.38. We focus on the top path of the P-I controller, which is the proportional-feedback control loop for the left wheel. The sequence of computations illustrated by this diagram are encoded in a few simple IC routines.

tor so as to keep the robot moving up the ramp at the same speed at which it was moving across the floor.

The IC program below illustrates both the proportional-control computation and the integral-control loop slaving the two wheels together:

```
float control_interval = 0.250; /* Run servo loop this often */
float des_vel_clicks = 0.0;     /* Desired vel, clicks/interval */
float des_bias_clicks = 0.0;    /* Desired bias, clicks/interval */
float power[2] = {0.0,0.0};     /* Power command to motor */
float integral = 0.0;           /* Integral of velocity difference */
float k_integral = 0.10;        /* Integral error gain */
float k_pro = 1.0;              /* Proportional gain */

/* Set and remember power level */
void alter_power(float error, int motor_index)
{ power[motor_index] = limit_range(power[motor_index]
     + error, 0.0, 100.0);
  pwm_motor(power[motor_index], motor_index); }

float integrate(float left_vel, float right_vel, float bias)
{ integral = integral + left_vel + bias - right_vel;
  return integral; }

void speed_control()
{ float left_vel, right_vel, integral_error,
     left_error, right_error;
  while (1)
    { left_vel = get_left_vel(); /* Get current vel */
```

```
    integral_error =
      k_integral *
      integrate(left_vel, right_vel, des_bias_clicks);
    left_error =
      k_pro * (des_vel_clicks - left_vel - integral_error);
    right_error =
      k_pro * (des_vel_clicks - right_vel + integral_error);
    alter_power(left_error, 0);
    alter_power(right_error, 1);
    sleep(control_interval); /* Run speed_control periodically */
    }}

void set_velocity(float vel, float bias) /* v,b:  [-100.0, 100.
{ des_vel_clicks = k_clicks * vel;   /* Convert from vel as % */
  des_bias_clicks = k_clicks * bias; /*  to vel as clicks/interva
  sign[0] = get_sign(vel - bias);    /* Sign of left vel */
  sign[1] = get_sign(vel + bias); }  /* Sign of right vel */


float k_clicks = 8.0 / 100.0;

void start_speed_control()
{ init_velocity();
  init_pwm();
  start_process(speed_control()); }
```

The integral controller works by representing the commanded robot velocity as two separate pieces of information, a common-mode desired velocity and a differential-bias velocity. That is, the desired velocity is the translational component and the bias velocity is the rotational component. Said another way, if the robot were commanded to go straight at 50% of full speed, its desired velocity would be [50.0, 50.0] and its bias velocity would be [0.0, 0.0]. This would coerce Rug Warrior to maintain a constant velocity of 50% of maximum speed, even as terrain or load on its wheels changed, as shown in Figure 7.39.

If the robot were commanded to spin in place about its right wheel at 35% of full speed, its desired velocity would be [0.0, 0.0] and its bias velocity would be [35.0, 0.0]. A command to arc forward and to the right would have both a desired velocity, say, [50.0, 50.0], and a bias velocity, say, [35.0, 0.0]

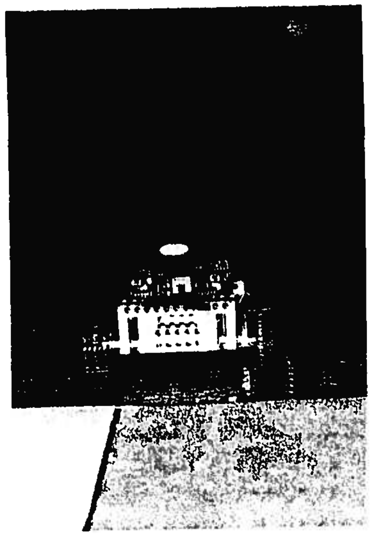


Figure 7.39. Rug Warrior is climbing up a ramp. Implementing a proportional-integral feedback controller keeps both wheels turning at the same speed and delivers more power to the motors as Rug Warrior begins to climb the ramp.

With this data structure for input, the integral-control loop adds the left velocity and the bias and subtracts the right velocity from that sum to calculate the error signal, e_3. The function integrate() accumulates this error over time, adding the new error to itself on each iteration. This running sum is multiplied by some constant, K_integral, and added into each motor's proportional controller. In this way, the new commanded velocity to each motor takes into account not only its own shaft-encoder's error signal but also the error signal between the two motors as it changes over time.

It becomes interesting now to play with the robot. Grab one wheel while the P-I controller is running, and try to keep it from spinning. The proportional control will try to raise the power level, and you will feel an increase in the torque output by the motor. If you hold the wheel tightly though, after a few moments, the other wheel will stop! This is because the program was not able to speed up the motor you were holding, and so the only way it could keep the two motors running at the same speed was to slow the other one

Try playing with the program in different ways. Change the values of the constants K_pro and K_integral. If these constants are made larger, the reaction time of the control loop will increase, but if you make them too large the system might become unstable and the motor will hunt, slowing down and speeding up but never converging on a steadily controlled speed.

Play with the time interval, too. The function speed_control is implemented as a process in **IC** that runs at a frequency specified by the variable control_interval. Changing the control interval modifies Rug Warrior's reaction time, also.

What we have implemented here on Rug Warrior, with a very minimal amount of hardware and an elegantly few lines of code, is a classical feedback-control system. These types of techniques have been well studied and are useful for a large number of problems. In Chapter 9, we will look at a different kind of control paradigm, a subsumption-style control system, which focuses on the problem of deciding which behaviors to select, given that many may be triggered from a large set of noisy and possible conflicting sensors.

7.9 References

A number of books which motor design and performance in great depth. *Electric Machinery*, by Fitzgerald, Kingsley and Umans (1990) gives a thorough treatment of the electromechanics of a wide variety of AC and DC motors. The three-volume set by Woodsen and Melcher (1985) *Electromechanical Dynamics*, delves into the physics behind the generation of electromechanical forces.

A comparative analysis of actuator technologies, spanning the range from electromagnetic motors to piezoelectrics and human muscle, can be found in the work of Hollerbach, Hunter, and Ballantyne (1991). They compare these alternatives from the point of view of applicability to robotics.

Our discussion of piezoelectric ultrasonic motors was rather brief. The piezoelectric ultrasonic motor of Figure 7.3 was made at the MIT Mobile Robot Lab by Anita Flynn. Further reading can be acquired in literature from a number of countries. Piezoelectric ultrasonic motors were invented by the Russians in the sixties (Ragulskis et al. 1988) and later commercialized by the Japanese (Sashida 1982). Recently, these motors have appeared in Japanese autofocus lens actuators (Hosoe 1989), paper-pushing actuators in copiers (Ohnishi et al. 1989) and as silent alarms in wristwatches (Kasuga et al. 1992).

Shape memory metals and artificial muscles are somewhat new to mobile robots. An informative booklet describing how to work with shape memory metals for small robots can be obtained from Mondo-tronics (1991). Artificial muscles also hold great promise for compact robotic actuators. Much of the pioneering work was done by Tanaka. Nice overviews can be found in Tanaka (1981) and Brock (1991).

For those interested in micromechanics, a review on silicon electrostatic microactuators can be found in the article by Howe, Muller, Gabriel, and Trimmer (1990). Progress in microfabricating ultrasonic motors and pumps can be found in papers by Moroney, White, and Howe (1989, 1990), Flynn et al. (1992) and Udayakumar et al. (1991).

Literature on power electronics, power MOSFETs, and motor-driver integrated circuits is available in application notes and data books of manufacturers such as Motorola, Supertex, Siliconix, and International Rectifier. The texts *Power MOSFETs*, by Grant and Gowar (1989) and *Power Electronics for the Microprocessor Age*, by Kenjo (1990) give excellent background on driving motors. For a practical guide to servo loops and interrupts, see Foster (1982).

8 Power

A mobile robot requires a power system that can meet several goa simultaneously. The power source must store energy sufficient t allow the robot to perform a useful amount of work. To ensur proper operation of the onboard electronic circuits, power must b provided at a constant voltage. Noise and power glitches produce by one circuit component must not be allowed to interfere with an other component.

8.1 Batteries

Batteries are by far the most common solution employed by mobil robots for the problem of energy storage. A battery converts chem ical energy into electrical energy on demand. From the chemica nature of batteries stems a complex variety of properties. We begi with a synopsis of those properties and subsequently delve furthe into selected properties.

Rechargeability A battery that cannot be recharged is a *primar* cell. One that can be is a *secondary* or *storage* battery