Appendix 3

Mobile Robots: Inspiration to Implementation
by J. L Jones and A. M. Flynn
(A K Peters Ltd, Wellesley, MA, 1993)

Chapter 2: Tutebot

2

TuteBot

2.1 A Tutorial Robot

Building a robot can be a lot of work. All the more so if the first plan is unnecessarily complex. This chapter is intended to help get you started with building robots while illustrating some key points about designing a robot's intelligence system. Our aim here is to keep you from getting too bemired in the myriad of details involved in creating a more sophisticated creature. We will show just how simple a robot can be and launch you on your way to building one.

Before proceeding to the more sophisticated Rug Warrior described in the next seven chapters, we will begin here by constructing TuteBot—a robot that is simple yet complete. Do not underestimate the elegance of simplicity. Often, the simplest solution takes the longest to comprehend, yet the simplest solution often illustrates the main lessons with the most clarity. Experienced designers agree that the first way they design something is usually the most complex way.

TuteBot will exemplify how a robot as a system (a collection of sensors, actuators, and computational elements) can be organized in such a way that intelligent actions result in response to certain stimuli. TuteBot will consist merely of a circuit, a chassis, a sensor, a battery, and two motors. It can be programmed by adjusting two potentiometers. The entire robot will be built from LEGO parts and a few electronic components that are readily available at Radio Shack and other electronic hobby stores.

What will TuteBot be able to do? Its repertoire of behaviors will endow it with the capabilities to explore its world, escape from objects with which it collides, and follow along walls that it detects with its bumper.

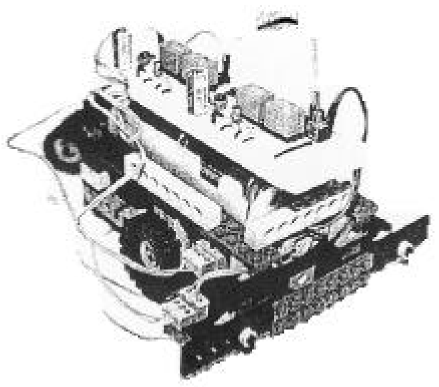

Figure 2.1: TuteBot is a robot that can explore its surroundings, escape from collisions with obstacles, and be programmed to follow walls. This front view of the robot shows the front bumper, mounted on two microswitches, for detecting collisions. The electronic breadboard containing the circuitry of TuteBot's brain is positioned on the chassis, just above the batteries.

A completed TuteBot is shown in Figure 2.1. The front fender acts as a bump sensor and detects collisions with obstacles in its path. Two wheels driven by separate motors are used for propulsion. (Only one drive wheel is visible in this photograph.) A trailing caster wheel maintains stability. Above the chassis is the battery case, and mounted on top of the batteries is the breadboard, containing TuteBot's electronic circuitry.

All the mechanical components used here are LEGO parts: motors, gears, axles, wheels, switches, and connectors. LEGO is a very good source of parts for building robots, as the designer can prototype mechanisms quickly without recourse to a machine shop. The LEGO Technics series kits come with even more advanced components, such as pistons, pumps, shock absorbers, differential gears, universal joints, battery cases, and even optical encoders.

It is probably worthwhile to order catalogs directly from LEGO, as their mail order and educational divisions sell some components that are not in neighborhood toy stores. Addresses and phone numbers are listed in Appendix C. Other types of mechanical building-block kits are also available, such as Fischer-Technic and Meccano.

TuteBot's brain is entirely analog circuitry. No integrated circuits are required, and all components, including the breadboard, can be found at

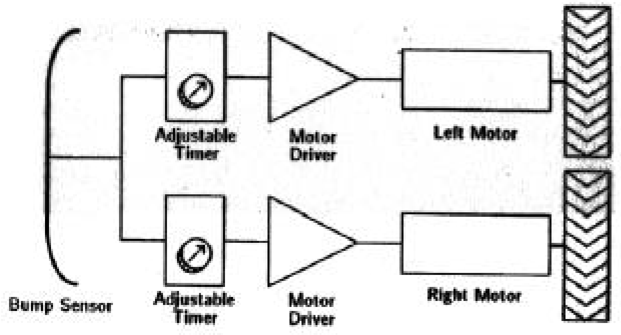

Figure 2.2: The essence of TuteBot. Two motors, two wheels, a bump sensor, two potentiometers for programming, and two motor drivers are enough to create a concrete example of a simple robot—an intelligent connection of perception to action.

a Radio Shack store. The only tools required to put together TuteBot are wire cutters, wire strippers, and possibly a soldering iron for making connectors. An oscilloscope is not necessary, although having one always makes debugging easier. A multimeter should suffice for debugging TuteBot.

A block diagram of TuteBot, shown in Figure 2.2, illustrates how the bump sensor is connected to the actuators. The signal created when the bump sensor detects contact is sent to the motor-driver circuitry for each wheel, signaling the robot to back up. An adjustable timer associated with each motor driver determines how long each wheel should reverse.

2.1.1 TuteBot Behaviors

With a minimal amount of hardware, obstacle avoidance can be implemented on TuteBot. Figure 2.3 depicts the sequence of actions that occur when TuteBot strikes an obstacle. The robot is initially moving directly forward, toward the shoe. As it strikes the shoe, both motors reverse and the robot backs straight up. However, one motor stays in reverse longer than the other, and the robot begins to turn; in this case, the right motor reverses longer, causing TuteBot to turn to the right. At some point, the right motor stops reversing and both motors go forward, leading TuteBot off in a new direction, hopefully, with a wide enough berth to avoid the shoe. If not, the robot bumps into the shoe again and the process repeats until TuteBot turns far enough to the right to avoid the shoe.

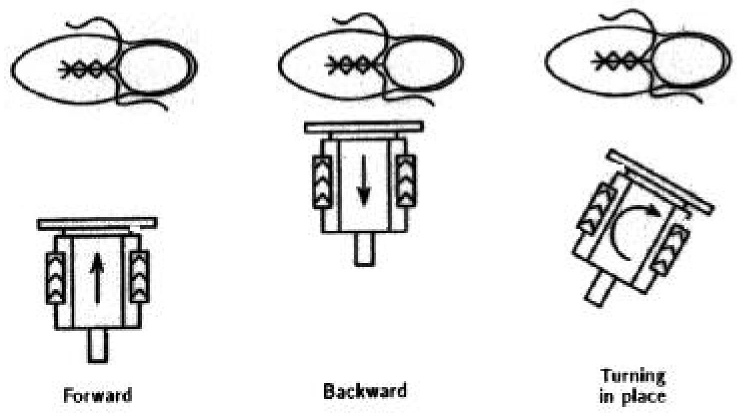

Figure 2.3: TuteBot's basic operation. When the power is switched on, TuteBot moves forward until it encounters an obstacle. TuteBot then backs up, turns in place, and resumes its forward motion. The time spent backing up and turning in place is programmed by the user.

A timing diagram that graphs this sequence of events is shown in Figure 2.4. The top graph depicts the signal generated by the front fender's bump sensor. The bottom two graphs illustrate the signals sent to the right and left drive motors, respectively.

Initially, both motors receive signals that direct them to go forward. The signal the bumper sends to the adjustable timers is binary—low for no contact, high when an obstacle is struck. The timers, in turn, provide binary signals to the motor drivers—high for forward rotation, low for reverse rotation. Once activated, each timer continues to supply the low signal for a characteristic time. The motor drivers interpret this high or low signal by providing forward or reverse current to the motors, respectively.

Assume that the timers are set for delays of t_r seconds and t_l seconds for the right and left motors and that $t_r > t_l$. After encountering an obstacle, the robot will back up for time t_l. It will then turn in place to the right (the left motor turns forward, the right motor stays in reverse) for time $t_r - t_l$. It will then resume moving forward on a different heading, thus avoiding the obstacle.

An additional behavior can be made to emerge from the robot. If we bias the motors so that, when going forward, one motor turns faster than the other, the robot will move in an arc. This slowdown in speed can be

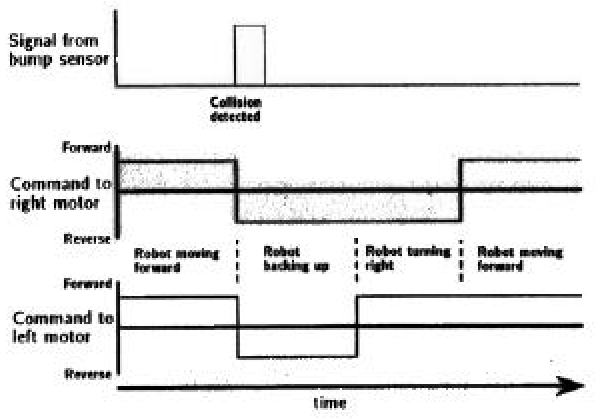

Figure 2.4: The timing sequence generating TuteBot's backup behavior. Both motors normally move in the forward direction, as shown in the bottom two graphs. When the bump sensor is activated, both motors reverse. The right motor stays in reverse longer than the left, causing the robot to turn to the right. When both motors resume forward motion, the robot moves on a new heading.

implemented by adding a resistor in series with one motor. If, for instance, the left motor is forced to turn significantly more slowly than the right, the robot will arc to the left. By combining this forward arcing behavior with the earlier back-and-turn behavior, TuteBot can be coerced to follow a wall, as was illustrated in Figure 1.2.

To demonstrate this, we would place the robot with a wall to its left and adjust the timers so that, after encountering a bump, the robot will back up and turn a bit to the right. Now, when going forward, the robot will arc to the left until it hits the wall; then it will back up, turn right, and head forward in an arc until it bumps the wall again. For suitable settings of the parameters, the robot should be able to turn through a doorway and negotiate either inside or outside corners.

It is important to note that nowhere in TuteBot's simple brain does it have knowledge of what a wall is or what is required to follow a wall. Rather, the superposition of a simple set of reflex actions allows a more complex behavior to emerge. This idea of seemingly complex behaviors emerging from a collection of simple rules is the underlying notion of a subsumption architecture, which was introduced earlier. We will see more complex examples when we get to the microprocessor-controlled Rug Warrior.

2	Motor, 4.5 volts (part 9859, $18.00 each)
1	Gear set (part 9853, $17.00)
1	Brick set (part 9858, $22.25)
1	Connectors and toggle (part 9851, $23.30)
1	Tire and wheel set (part 9855, $13.25)
2	Plate sets (part 9857, $19.10 each)
1	Axle set (part 9856, $14.85)
2	Touch sensors (part 9867, $9.55 each)
1	Connecting lead set (part 9861, $18.00)

Figure 2.5: TuteBot can be constructed from these or similar parts. Another possibility is to purchase LEGO sets 9605 ($196.20), 9851, and two 9867s. Technic Control 0 set ($161.00) with additional gears from set 9853 could also be used.

2.2 Building TuteBot

TuteBot senses the world through a front bumper mounted on two LEGO switches. It steers by individually changing the direction of its drive wheels, while a trailing caster wheel supports the robot in a three-point stance. A simple relay, transistor, and capacitor circuit provide all the computational power and memory TuteBot needs.

We will begin describing the construction of TuteBot by stepping through the mechanical layout of how to mount motors, attach wheels, and add gears. Figure 2.5 lists all the mechanical parts that will be needed. Except for the motors and switches, it is not necessary to follow the parts list exactly. Much can be learned by making creative use of whatever parts are available, and many of these parts can be reused later in building a chassis for Rug Warrior.

The chassis of TuteBot can be constructed by following the sequence of steps outlined in Figures 2.6 through 2.11.

The motors need to have gears attached to them because direct current (DC) motors usually spin too quickly and have too little torque to drive the loads of the wheels. Attaching a gear stage to the motor shaft, or "gearing down a motor," causes the motor to spin more slowly but with more torque at the output of the gear stage. Thus, the wheel can push against the floor with more force.

The first step is to start building the chassis and mount one of the motors with its associated geartrain. The geartrain connected to each motor is composed of a series of three stages. Each stage has an 8-tooth gear meshing with a 24-tooth gear. That is, the shaft from the 24-tooth gear fits into the 8-tooth gear of the second stage and so on. Figure 2.6 shows

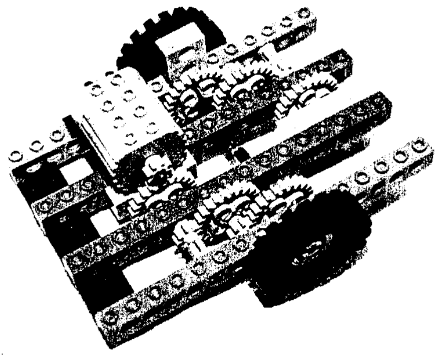

Figure 2.6: Begin the TuteBot by constructing the frame. The front of the robot is to the right in this photograph. In the LEGO motor shown, the output shaft is coupled to the right wheel (the nearer wheel in the photograph) through three stages of gears for a 27:1 geardown. This geartrain is needed to reduce the speed and increase the torque of the motor.

how the motor that will drive the right wheel is mounted on the left side of the chassis, leaving space for the geartrain between the motor and the wheel. The geartrain for the left wheel is in place, but its motor has not yet been incorporated. LEGO motors are used on TuteBot, as they mount easily with the LEGO axles and bricks.

The speed reduction provided by each stage is 8:24, or 3:1. Thus, the full-speed reduction of the three stages connected in series is 27:1. The torque the wheels can supply is correspondingly increased by a factor of 27, neglecting losses due to gear friction. If larger diameter wheels are chosen, the gear ratio must be increased. Gears and motors are explained in more detail in the later chapter on motors (see Chapter 7).

As you construct the chassis, it is important to make sure that the gears mesh properly and that the shafts do not bind. Install the motors as a last step, testing beforehand that wheels and gears spin freely. Small misalignments in the chassis and warped gearshafts can cause unnecessary friction and degrade the performance of TuteBot.

The next steps in building TuteBot's chassis are to add the left wheel's motor and then attach both wheels. After that, begin assembling the caster wheel. The caster is composed of two small wheels, mounted on spin freely on an L-shaped support as illustrated in Figure 2.7.

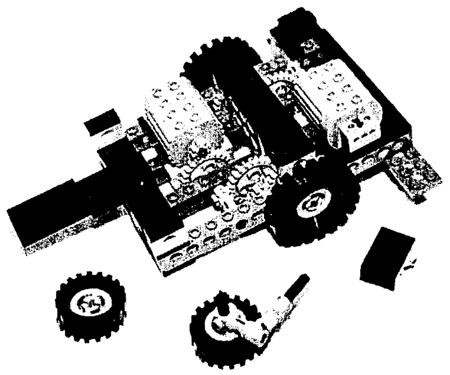

Figure 2.7: The caster wheel assembly is visible in the foreground. One of the two LEGO bump sensor switches can be seen next to the drive wheel.

In this figure, the front of the robot again points to the right. The caster support plate (the long black piece) has been attached to the rear. The caster support strut, a LEGO axle, will protrude upward through a hole in the support plate. This will allow the caster to swing freely as it follows the body of the robot. The castor's axle passes through a LEGO piston rod. The piston rod is connected at a right angle to a toggle joint, and the support strut is mounted in the toggle joint and separated from the support plate by spacers. Spacers on the vertical part of the L prevent the wheel from colliding with the horizontal support attached to the chassis when the wheel swings.

Once the caster wheel has been assembled and attached to the rear of TuteBot, begin work on the front bumper. Figure 2.8 is a schematic of how the front fender is connected, both mechanically and electrically. Although two physical bump switches are mounted to the chassis in order to hold up the fender, electrically, they are wired in parallel and so only deliver a single bit of information to the control system—whether or not an obstacle has been struck. No information about which side, left or right, made the contact is passed on to TuteBot's brain.

To build the front bumper, mount the two LEGO momentary contact switches on the front of the chassis, facing forward. The bumper, as shown in Figure 2.9, can be made from several long LEGO pieces. When connecting the bumper, make sure that the switches do not bind or stick when the

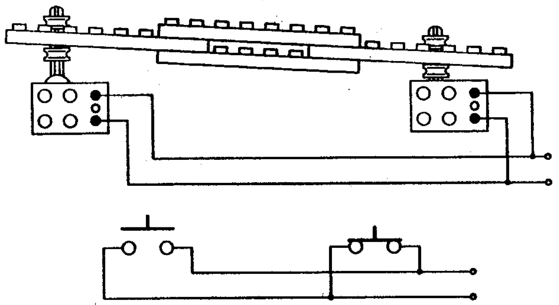

Figure 2.8: A schematic of the front bumper configuration. Two bump switches are required mechanically to hold the front bumper, but only one is required electrically to signal a bump, so the two switches are wired in parallel.

fender is pressed. If they do, loosening the small retainer that holds the short shafts to the bumper may solve the problem.

After the front bumper has been attached, most of the chassis has been completed. In the remaining steps, we will add a brick structure above the motors to fix them in place and to provide a level surface on which to place the batteries.

Next, we will make connectors for the motors and bump switches. LEGO provides connectors that fit with their component motors and switches. However, the other ends of these cables must be modified so that they can be plugged into TuteBot's breadboard.

There are a number of options for making connections between the breadboard and the motors and bump sensors. One simple solution is shown in Figure 2.10, at the left in the photograph. Strip some 22-gauge solid hookup wire, and use a pair of pliers to force two small pieces into the plugs on the LEGO connector. These pins will then fit nicely into the sockets on the breadboard.

Another possibility for making a suitable connector is to cut the cable or remove the screwed-on connector casing and then solder your own connectors to the leads. This method is shown at the right in Figure 2.10, where a terminal strip connector has been attached to the end of the LEGO cable.

After the cables have been connected to the motors and bumper, we still must make cables for connecting the battery pack to the breadboard. The battery holder wires can be stripped, twisted tightly, and inserted directly into the breadboard. Coating the wires lightly with solder will make this step easier and the connections longer lasting.

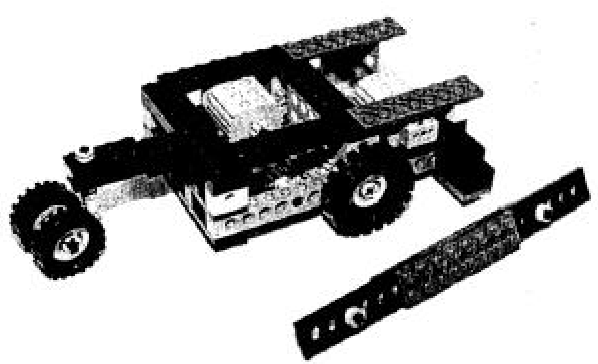

Figure 2.9: The bumper panel in the foreground attaches to two LEGO bump switches on the front of TuteBot. The rear caster (shown at the left) has been attached to the caster support plate and capped by a small, round retainer.

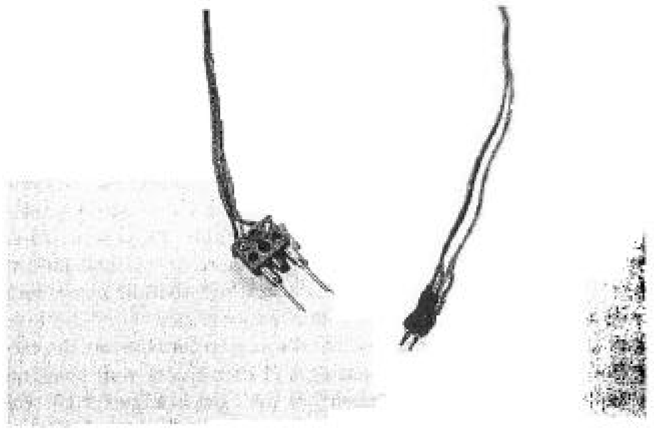

Figure 2.10: At the left is an example of inserting solid hook-up wire into the LEGO motor connector's plug. At the right is the other end of the LEGO motor cable, where the casing has been removed and a terminal strip, plug-type connector soldered on.

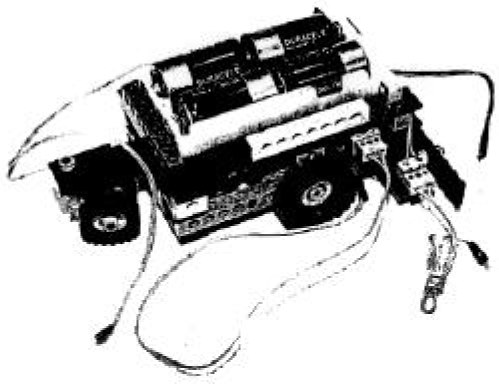

Figure 2.11: TuteBot has motors and batteries wired up and connectorized. The battery case (containing four 1.5 V alkaline C cells) has been mounted above the motors. Wires from the two motors and the battery case are ready to be plugged into the breadboard.

Now your TuteBot should look similar to that in Figure 2.11. Note that LEGO bricks have been added to secure the battery pack in place. Also note that the front bumper switches have been wired in parallel. Only one pair of wires goes from the two bump switches up to the breadboard, but using two switches rather than one made the mechanical mounting of the bumper simpler.

The final step is to mount the breadboard on top of the battery pack.

In the next section, we will discuss building the electronic circuitry for TuteBot's brain. Once this has been assembled, mounting it on top should produce a robot resembling that in Figure 2.1, shown at the beginning of this chapter.

2.2.1 Electronic Components

Before we get into the specifics of the control system for TuteBot, we will take a moment here to describe the basics of a few common electronic components, such as relays, transistors, resistors, capacitors, diodes, and the like. Figure 2.12 illustrates the relationships between the physical components we will use on TuteBot and their schematic symbols. (A *schematic* *diagram* illustrates the topology of how all the electronic components are connected into a circuit.)

First, the *relay* shown in the upper-left-hand corner of Figure 2.12, is a type of electrically controllable switch. TuteBot uses relays to switch the polarity of the voltage applied to its motors and thus reverse their directions. The idea behind a relay is that a small current flowing in the relay's coil can switch much larger currents flowing though its contacts. The way a relay works is that, when different voltages are applied to the two lines marked coil, the resulting current creates a magnetic field inside the device. This field attracts a metal lever to which the internal switch contacts are attached. Activation of the lever in turn disconnects one circuit and connects the other. (This is the meaning of SPDT—single-pole, double-throw. The relay can connect a single circuit in either of two ways.) When no voltage is applied, the line marked com, or *common*, is connected to nc, the *normally closed* pin. When voltage is applied across the coil, com is disconnected from nc and connected to no, the *normally open* line.

Next come bipolar *transistors*. A bipolar transistor has three terminals: a *base*, *b*; a *collector*, *c*; and an *emitter*, *e*. For a particular transistor case design, the correspondence between these symbols and the physical leads can be found in the manufacturer's data book. Transistors can be used as amplifiers or switches. TuteBot employs transistors to supply a current sufficient to activate the relay. There are a great variety of transistors. Two of the important parameters that differentiate among them are amplification factor and maximum power-handling ability.

A *diode* is a device that allows current to flow in one direction but not the other. If the + end of a diode, the *anode*, is connected to the + terminal of a battery and the − end of the diode, the *cathode*, is connected to the − terminal of the battery, a large current will flow through the diode, enough to damage the diode or battery. Usually, a resistor is placed in series with a diode to limit current to a safe level. If the connection is reversed, no current will flow. Diodes are rated according to the amount of current they can handle without damage and the maximum reverse voltage they can sustain. A band on the diode usually marks the − end. The triangle on the diode's schematic points in the direction current is allowed to flow. TuteBot uses diodes to isolate parts of the circuit and short out induced voltages of the wrong polarity.

A single-pole, single-throw (SPST) switch is shown at the left of the second row in Figure 2.12. Switches are characterized both by the number of connections that can be made or broken by moving the switch lever and by the number of different lever positions that make contact. An SPST switch is the simplest type of switch. With the switch lever in one position, connection between its two leads is broken. With the switch lever in the

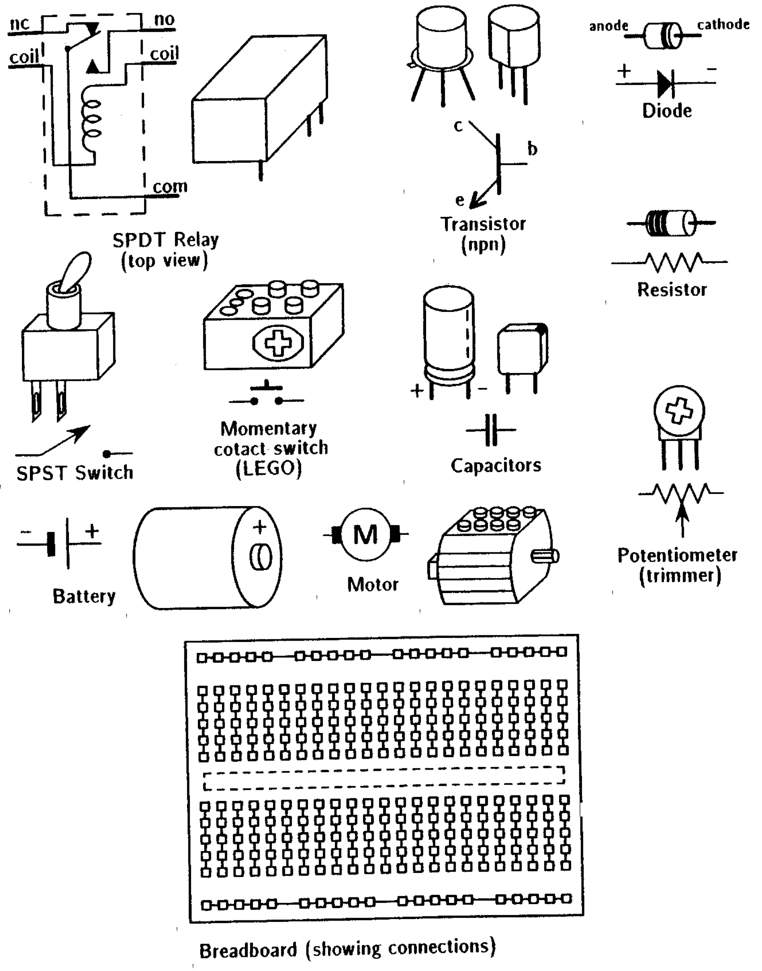

Figure 2.12: The relationships between schematic symbols and the physical components they represent. All of these components are used in TuteBot's brain. No other components are necessary, and the entire circuit will fit in a 6-inch-long breadboard mounted on top of TuteBot's chassis.

other position, connection is made. An SPST switch serves as the power switch for TuteBot.

To detect collisions, TuteBot uses *momentary contact switches.* This type of switch has an internal spring that endeavors to keep the switch in one state. As long as the switch lever or push button is pressed, the switch circuit is closed. When the lever is released, the circuit opens. Momentary contact switches with the opposite sense (open when pressed, closed when not pressed) are also available.

Resistors impede the flow of current. Their ability to do this is measured in ohms, Ω; kilohms, $K\Omega$; or megohms, $M\Omega$; The current, I, that will flow through a resistor with resistance R, given an applied voltage, V, is $I = V/R$. This is known as Ohm's law. When current flows through a resistor, it must dissipate power. A resistor's capacity for dissipating power is measured in watts. In general, a resistor with a higher wattage rating will be physically larger than one with a smaller wattage rating.

To block direct current but allow the passage of alternating current, we use a *capacitor.* Once connected to a voltage source, such as a battery, current flows into the capacitor until it has accepted as much charge as it can. This ability to accept charge is usually measured in units of micro- or picofarads (μF or pF). If the voltage supply is removed from the capacitor, the stored charge keeps the voltage across the capacitor constant. Shorting the leads together causes a current to flow until the charge has been depleted and the voltage across the capacitor has gone to zero. TuteBot uses capacitors as memory cells. The presence or absence of stored charge represents the robot's recent history, or state.

There are many different capacitor technologies. Most capacitors can be connected into a circuit without regard for polarity. One type for which polarity is important is the electrolytic capacitor. The leads on this type of capacitor are marked + and − so that it is clear which way they should be inserted into the circuit. Electrolytic capacitors can generally store more charge in a smaller volume than other types of capacitors. The maximum voltage that can be applied to a correctly connected capacitor before damage occurs is listed as the WVDC (working voltage, direct current).

A *potentiometer* is simply a resistor whose resistance is adjustable. As with fixed resistors, there are a large number of resistances and maximum power ratings to choose from. A potentiometer allows the user to manually alter some parameter of a circuit. We will use potentiometers in TuteBot to control its response to collisions—how long it backs up and how long it turns in place before proceeding forward again.

The first item found in the third row of Figure 2.12 is the *battery.* Batteries supply current as required at some characteristic voltage. The nominal voltage rating of a battery is normally stamped on its case. TuteBot, for

instance, uses four 1.5 volt (V) alkaline batteries. Many toys and portable appliances use nickel cadmium (NiCd) batteries; NiCd batteries produce 1.2 V per cell.

Motors convert electrical energy to mechanical energy. LEGO motors were chosen for TuteBot because they are easy to integrate into the chassis and they happen to provide sufficient power for this application.

The last component in Figure 2.12 is the electronic *breadboard.* Internal connections among its sockets are shown. A breadboard allows us to quickly connect components into a circuit and to make changes easily. Vertical columns are connected, as are the top and bottom horizontal rows. Typically, we would connect these rows to power (the positive side of the battery pack, in this case) and ground (the negative side of the battery pack). The space between the columns in the center is the correct width to accommodate standard integrated circuit chips. In TuteBot's circuit, these center positions are occupied by relays. The relays are the same width as standard chips.

Later on in this book, when we discuss Rug Warrior, we will introduce a number of other components, such as power MOSFET transistors, crystals, operational amplifiers, photoresistors, light-emitting diodes, logic gates, microprocessors, memories, and so on.

2.2.2 Electronic Construction

With device descriptions as background, now let us look at the circuit for TuteBot's brain. Figure 2.13 gives the schematic.

Transducers are typically connected on either side of the circuit for a robot's brain. For instance, on the input side, batteries and sensors act as input transducers. A battery converts chemical energy into electrical energy, and a sensor converts a physical phenomena from a mechanical form (say, the force acting on a bump switch) to an electrical form. On the output side, motors, speakers, lights, and so on act as output transducers. The motors on TuteBot convert electrical energy into mechanical energy. Between the input and output transducers is the electrical circuit, which does the information processing. The time variation in the signals, the voltages and currents in the circuit, provide information transfer.

In describing a circuit's behavior, we usually speak of voltage across a device and current through a device. One bit of confusion can arise due to a verbal shorthand of speaking of such things as "the voltage at point A." What is meant and what would be more precise would be to speak of "the voltage across the network between points A and ground." The verbal shorthand comes about because ground is usually taken to be the reference, 0 volts.

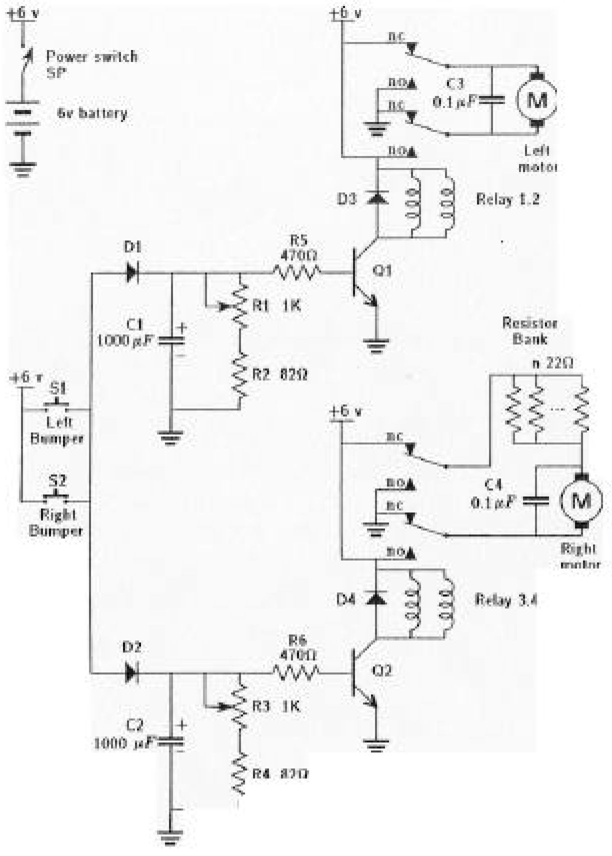

Figure 2.13: Schematic for TuteBot's brain. The two bump switches, $S1$ and $S2$, are connected in parallel. One side of each switch is tied to +6 V, and the other, to both motor-driving circuits. The diodes act to separate the two halves. For instance, in the circuit driving the left motor, the resistor, capacitor, and potentiometer network ($R1$, $R2$, and $C1$) charges up when a bump switch is activated, turning on transistor $Q1$ and reversing the motor through the relay. Once the switches are no longer activated, charge on capacitor $C1$ drains away at a rate that depends on the setting of the potentiometer. Thus, after some time, the motor will no longer reverse and TuteBot will resume forward motion.

1		Breadboard (at least 45 columns) (RS 276-174)
1	SP	SPST power switch (RS 275-645)
4	Relay 1, 2, 3, 4	SPDT relay with 5 V coil (RS 275-243)
2	$Q1$,$Q2$	2N2222A or MSP2222A transistors (RS 276-2009)
2	$C1$,$C2$	1000μF capacitors, 6.3 WVDC (or more)
2	$C3$,$C4$	0.1μF capacitors
2	$R1$,$R3$	1K potentiometers (RS 271-280)
4	$D1$,$D2$,$D3$,$D4$	1N914 or 1N4001 diodes
1		Battery Holder for 4 "C" cells (RS 270-390)
2	$R2$,$R4$	82Ω, $\frac{1}{4}$ watt resistors
2	$R5$,$R6$	470Ω, $\frac{1}{4}$ watt resistors
9	Resistor Bank	22Ω, $\frac{1}{4}$ watt resistors
3 ft.		22-gauge solid hookup wire

Figure 2.14: Use these or similar parts to construct TuteBot's brain. A good understanding of how the circuit functions will allow the builder to make substitutions. Radio Shack part numbers are given in parentheses. Where no part number is given, any component with the listed parameters can be used.

The basic idea of TuteBot's circuit is that the two front bump switches (marked $S1$ and $S2$ in Figure 2.13) which are wired in parallel, generate a signal that tells the robot to back up. This bump signal is sent to both halves of the circuit. The diodes $D1$ and $D2$ act to separate the circuit driving the left motor from the circuit driving the right motor so that they can have independent specifiable time constants for how long each wheel should back up. The time constants are implemented with resistor-capacitor (RC) circuits that hold a voltage for a given amount of time, depending on the values of the resistor and capacitor. The timing signals from these RC networks then direct the motors to reverse directions for the specified amount of time. Some driver circuitry to condition the signal to provide enough current to drive the motor has to be added at this point. This motor-driver circuitry is implemented with transistors and relays. A bank of resistors is added in series, with one motor to regulate its speed in comparison to the other motor.

There are two ways to proceed at this point. One is to go ahead and just build the circuit and not worry about understanding how it works. Simply build it, mount it on TuteBot's chassis, plug in the connectors, and start playing with various behaviors by tweaking potentiometers and adding resistors in series with the motors. The other way to proceed is to convince yourself that you understand every last detail of the circuit configuration before you start stripping wire.

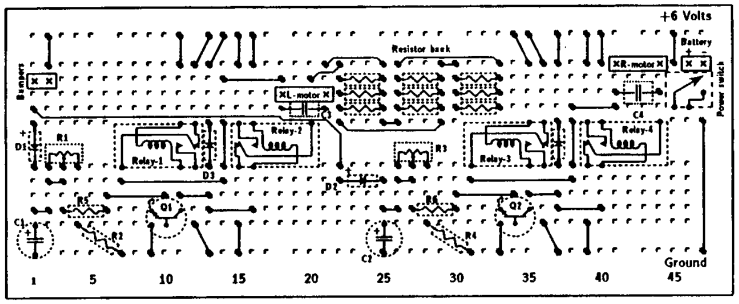

Figure 2.15: One possible layout of the TuteBot circuit. The horizontal row on top is connected to +6 V through the power switch from the battery case, and the bottom horizontal row is connected to ground. Four relays straddle the middle section, and other components and jumper wires are interspersed throughout the breadboard.

We recommend a quick skimming of the circuit description and then directly putting the circuit together. The parts list for the circuit is given in Figure 2.14. Because the purpose of this chapter is to overcome the inertia of getting started, an exact layout on a Radio Shack breadboard is given in Figure 2.15. Build the circuit just like this, and TuteBot should work. Later in this chapter, to achieve a better understanding, you can then go back through the circuit, observing voltage signals across various portions of the network with an oscilloscope to compare traces to graphs.

A finished breadboard is shown in Figure 2.16. One detail to note in assembling this circuit is that the relays may have leads too short to make good contact when inserted into the breadboard. First plugging the relay into a 14-pin socket, and then plugging the socket into the breadboard may solve this problem. Use care when installing the diodes and the electrolytic capacitors. These devices are polarized. If they are installed the wrong way, they may be damaged.

It is a good idea to test the circuit as you go. Build only half of it first, and check to see that it drives the motors as desired. With power applied and the motor not connected, check to see that pressing the bumper switches activates the relays. If the circuit is operating properly, a click will be heard. The bias resistors, $R5$ and $R6$, may need to be adjusted if relays or transistors other than the ones specified are used. If the relay does not operate, choose smaller resistors until it does (but don't go below about 100Ω.)

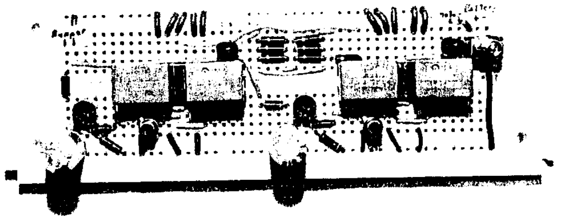

Figure 2.16: Details of the breadboard. Note that, in this example, the resistor bank contains seven resistors. Because the bank consists of a number of resistors connected in parallel, the motor can be slowed down by removing some resistors.

In general, it pays to be neat when breadboarding a circuit. Any time saved in quickly throwing together a sloppy circuit is usually more than wasted in debugging. Cut and strip wires to appropriate lengths so they lie flat on the breadboard. Buy lots of different colors of hookup wire, and stick to conventions for power and ground. If you use red for +6 V and black for ground, then it becomes easy to visually check your breadboard; namely, all wires connected to the top horizontal row should be red and all wires connected to the bottom horizontal row should be black.

Another important tip before turning on the power switch is to always "ohm out" power and ground—that is, check with an ohmmeter that power and ground have not been inadvertently connected on your breadboard. This prevents smoke from streaming out of your circuit. Never remove components with the power on. Power down first. If the circuit does not work, first check with a voltmeter that all points in the circuit that should be connected to power are actually at +6 V and that all points that should be at ground actually read 0 V. While this all sounds rather obvious, you would be surprised at how many problems are caught through these few simple tips.

2.2.3 Operation

For a more detailed exposition of the TuteBot circuit of Figure 2.13, we break the system into modules and explain each piece. The circuit is divided into two nearly identical halves. For simplicity, we describe only one half, the upper half, which controls the left motor.

As soon as power has been applied by closing the power switch, both motors will begin to turn forward and TuteBot will move straight ahead. If we look at the portion of the schematic showing the left motor's connection to its relays, we see that two lever arms can switch between normally open and normally closed connections. This type of relay topology is equivalent to a double-pole, double-throw (DPDT) relay, but actually, for TuteBot we use two single-pole, double-throw (SPDT) relays due to availability. We can see for the left motor that the normally closed connection applies 6 V across the motor. The motor should be installed on TuteBot so that this configuration initiates forward motion. The same is true for the right motor.

Again, looking at the left motor portion of the circuit, if TuteBot strikes an obstacle and either or both of switches $S1$ and $S2$ are closed, a current will flow through diode $D1$, charging capacitor $C1$. Simultaneously, current will flow through resistor $R5$ into the base of transistor $Q1$. The base current will cause $Q1$ to conduct—pulling current though the coils of the relays. When current is provided to the relays, they switch from the normally closed state to the normally open state. The motor terminal, previously connected to +6 V, is now connected to ground, and the other terminal, previously connected to ground is now connected to +6 V. This causes current to pass in the opposite direction through the motor, making it spin in reverse.

As the reversing motors cause TuteBot to back up, its bumper is no longer pressed against the obstacle and switches $S1$ and $S2$ are no longer closed. With the switches open, the RC circuit is no longer connected to +6 V. However, capacitor $C1$ continues to supply current for awhile to the base of the transistor and the motor continues its reverse rotation. The capacitor discharges at a rate controlled by resistors $R1$ and $R2$ (and $R5$ through the base-emitter junction of $Q1$). At some point, $Q1$ will cease conducting, the relays will open, and the motor will resume its forward rotation. Diodes $D1$ and $D2$ isolate the circuits so that the capacitors can discharge at the desired rates (so that current cannot drain off $C1$ and begin charging the right motor's RC circuit).

Figure 2.17 illustrates how the voltage across the left motor's RC network changes with time. With the switch closed, the battery charges the RC circuit (this voltage is taken as between point A and ground) up to V_o. When TuteBot backs away from the obstacle and the switch is opened, the voltage across the capacitor falls at a rate determined by the values of the resistor and the capacitor. To be precise, this relationship is $V = V_o e^{t/RC}$, where V_o is the power supply voltage. Figure 2.17b illustrates the RC network connected to the right motor. The smaller resistance in (a) causes the current to drain away more quickly, keeping the robot's left wheel in reverse

for a shorter time period than the right wheel. This causes the robot to turn to the right.

The left motor turns in reverse for a period of time, which is determined by the following factors:

- The size of capacitor $C1$.
- The value of bias resistor $R5$.
- The amplification factor of transistor $Q1$.
- The resistance of the potentiometer $R1$.
- The current level needed to activate relays 1 and 2.

A very brief motor reversal may be selected by setting the potentiometer to its smallest value. A reversal longer than the one available in the circuit as designed may most easily be achieved by increasing the values of $C1$, as it is actually the product of R and C, which sets the time constant.

We can see how the changing currents set up by the RC network are able to activate and deactivate the transistor $Q1$ by referring to Figure 2.18. Depending on the characteristics of the particular transistors and associated circuit components, a transistor can be used either as an amplifying device or as a switch.

The TuteBot circuit requires the transistor to act as a switch, as shown in (a). When base current is supplied, the switch closes and the load draws current because it is connected between power and ground. We will follow a very simple model of how a transistor switch operates: As long as the current flowing into the base of transistor $Q1$ is greater than or equal to $i_{b,sat}$, the switch will be on and current will flow through the load. When the transistor's base current falls below $i_{b,sat}$, the transistor will switch off and no current will flow through the load. A small base current is able to control whether or not a large load current is allowed to flow. In (b), we see that a base resistor is needed to set the base current for the transistor switch. The timing signals of the current flowing through the base resistor are shown in (c). For the duration of time that TuteBot was contacting the obstacle and the RC circuit was charged up to V_o, the base current was large enough that the transistor was completely on and saturated—that is, the collector current had reached its maximum possible level, $i_{c,sat}$.

As TuteBot backs up from the obstacle, the bumper switches open, and the voltage drains off the RC network, the current through the base becomes smaller. Eventually, it falls to $i_{b,sat}$, where the transistor begins to come out of saturation. The collector current falls to 0 and the load becomes open circuited. Actually, a small amount of current does continue

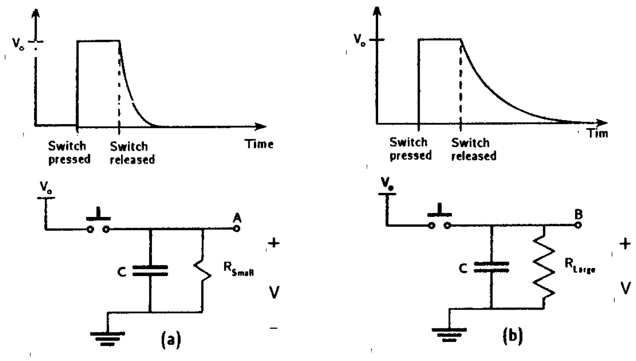

Figure 2.17: As long as the momentary contact switch is pressed, the voltage between point A and ground or point B and ground will be equal to V_o. When the switch is released, charge begins to drain from the capacitor through the resistor. The small resistance in (a) drains the capacitor more quickly than the large resistance in (b).

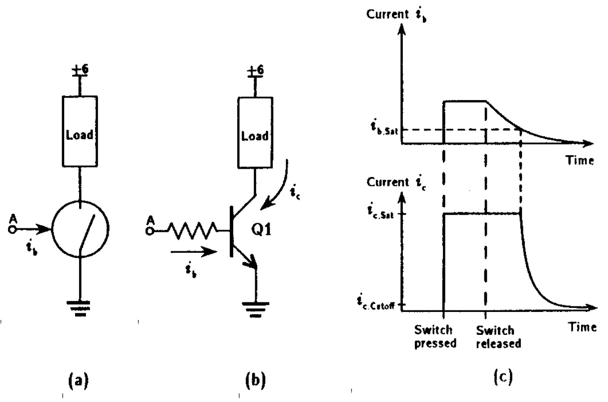

Figure 2.18: (a) A transistor is modeled as an ideal switch. (b) In reality the base current is set by the base resistor's value for a given voltage applied to terminal A. (c) The base current must be large enough to put the transistor into saturation (turning it fully on).

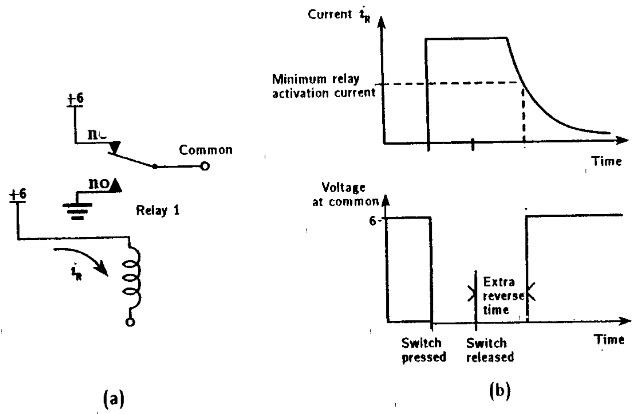

Figure 2.19: (a) The amount of current, i_R, flowing through the coil of the relay determines whether its common terminal is connected to its nc, normally closed, or its no, normally open terminal. When i_R falls below minimum activation current the state of the relay changes. (b) The "Extra reverse time" is the extra amount of time the motors run in reverse after the bumper switch has been released.

to flow for awhile, even when the transistor is off. The transition from on to off is not quite as sharp as with a real switch.

When the transistor switches on, it draws current, i_R, through the coil of the relay, as shown in Figure 2.19(a). Current through the coil creates a magnetic field, which forces the relay lever to move. The relay lever then switches the common connection (attached to one terminal of the motor) from the normally closed pin of the relay to the normally open pin. This happens on each of the two relays associated with both motors, reversing the polarity of the voltages applied across each. For all the time that $Q1$ is on, current is pulled through the relay, causing the motor to switch from forward motion to reverse motion.

The essential difference between the left and right motors is the relative times at which they turn off their reversing behaviors. In Figure 2.19(b), we can see the timing diagrams of the current through the relay and the resulting voltage applied between one motor terminal and ground.

First, as the transistor $Q1$ turns off, it causes load current to stop flowing. This takes some amount of time after the bump switch is released due to the time delay set up by the RC circuit. When the current through the relay falls to a level that can no longer sustain the necessary magnetic field to keep the lever attracted to the normally open pin, the relay switches back to its normally closed configuration. This occurs to both SPDT relays attached to each terminal of the left motor.

The lower graph in Figure 2.19(b) shows the resulting voltage change over time for one of the left motor's terminals. The other motor terminal, normally at 0 V, switches to +6 V when the bump switch hits an obstacle and reverts to 0 V again (after the time lag set up by the RC network) after the bumper is released.

A similar mechanism is implemented on the right motor, except that its potentiometer, $R3$, is tuned to give a different time delay than for the left motor. The robot can thus be programmed to turn more or less sharply by adjusting the potentiometer setting for each wheel.

Three other points are worth mentioning concerning the left motor circuit of Figure 2.13. The first is the appearance of diode $D3$ across the two SPDT relays. The reason for adding this device is that the diode protects the circuit from the large voltages that are induced by collapsing magnetic fields in the relay coils when the transistor turns off. If diode $D3$ were not there, the inductance of the coil would try to force the current flowing through it to keep flowing down through transistor $Q1$. Because $Q1$ has been opened, current through the coil results in an increase in voltage at the collector of $Q1$. If this voltage exceeds the maximum rating that the cutoff transistor can withstand, it will be damaged or blow up. The diode alleviates this problem by providing a return path for the coil current when the transistor turns off.

The second point to note in the final circuit is that the capacitor $C3$ has been placed across the terminals of the motor. This capacitor attenuates the voltage spikes produced by the motor. Typically, these capacitors are soldered directly to the motor terminals rather than placed back at the circuit board.

Finally, note that a resistor bank is connected in series between the relay and the right motor in the schematic. The purpose of this bank is to match speeds between the two motors. Determining which motor should be connected to the resistor bank must be done by experiment. Although the motors and geartrains are supposedly identical, in reality, they are not.

These differences manifest themselves as mismatches in the speed at which the wheels turn. To make the adjustment, first short out the resistor bank. Then turn TuteBot on, and allow it to roll across the floor. It will make a long arc in one direction or the other. If TuteBot turns to the left, then the right motor is turning faster; attach the right motor to the resistor bank. If the opposite occurs, attach the left motor. With n resistors wired in parallel, the total resistance, R_T, of the resistor bank increases as each resistor, R, is removed: $R_T = \frac{1}{n}R$. The more resistance we place in series

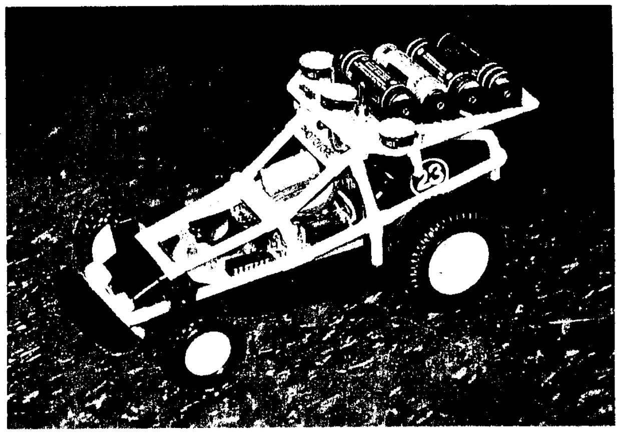

Figure 2.20: This MIT robot, known as Photovore, performs an interesting set of light seeking behaviors. It uses only analog circuitry to achieve its behaviors.

with the motor, the less current will flow and the slower the motor will turn. Add or remove resistors until both motors rotate at the same speed.

TuteBot is now complete and ready to go. Try running it in a few different environments. Try adding the wall-following behavior, discussed earlier, to bias the motor speeds so TuteBot travels forward in an arc by changing the resistor bank. If TuteBot goes too fast and falls apart when it crashes into things, electrical tape, double-sticky tape, and glue work wonders with LEGOs.

Have fun!

2.2.4 Exercise

When the wall-following behavior has been implemented, as described above, TuteBot will simply turn in circles if it is set in motion far from a wall. As an exercise, try to devise an additional behavior (possibly requiring another component or two) that will cause the robot to go straight until it encounters a wall and then begins to follow the wall.

2.3 References

The TuteBot exercise in this chapter was designed to be a simple example to get started. However, it might be the case that you feel more at home with a computer-controlled robot than with the analog electronics of TuteBot. If so, proceed to the next chapter, describing Rug Warrior's microcontroller brain. For background in electronics, the "bibles" for robot builders are Horowitz and Hill (1989) and the associated student manual (Hayes and Horowitz 1989), which give extensive practical information on analog electronics in very readable presentations. *The ARRL Handbook for the Radio Amateur* (Kleinschmidt 1990) is another very good source for beginners in electronics.

For articles and reports on simple robots and how to build things, a few pieces have trickled out of the MIT Mobile Robot Lab over the years. Connell (1988) describes Photovore, shown in Figure 2.20, a light-eating, dark-avoiding, relay-driven robot using three photoresistors and a Radio Shack toy car base. Photovore is also described in *The Olympic Robot Building Manual*, (Flynn et al. 1988), from which this book grew. A picture book of the resulting talent show robots is contained in Flynn (1989). Another minimalist mobile robot is described in the August 1991 issue of *Popular Electronics*, (Connell 1991). Kits and printed circuit boards for building your own version of Photovore can be purchased from Johuco, Ltd. See Appendix C for addresses and phone numbers in the list of manufacturers.