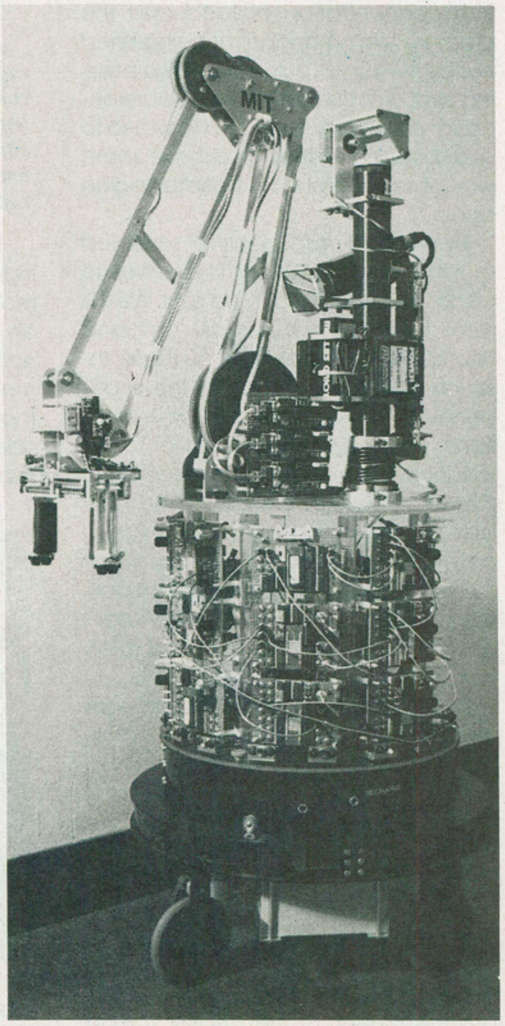
Popular Electronics®
AUGUST 1991

DESIGN your own ROBOT

Learn the fundamentals of brain activity and how to simulate animal behavior by building your own robot.

BY JONATHAN CONNELL



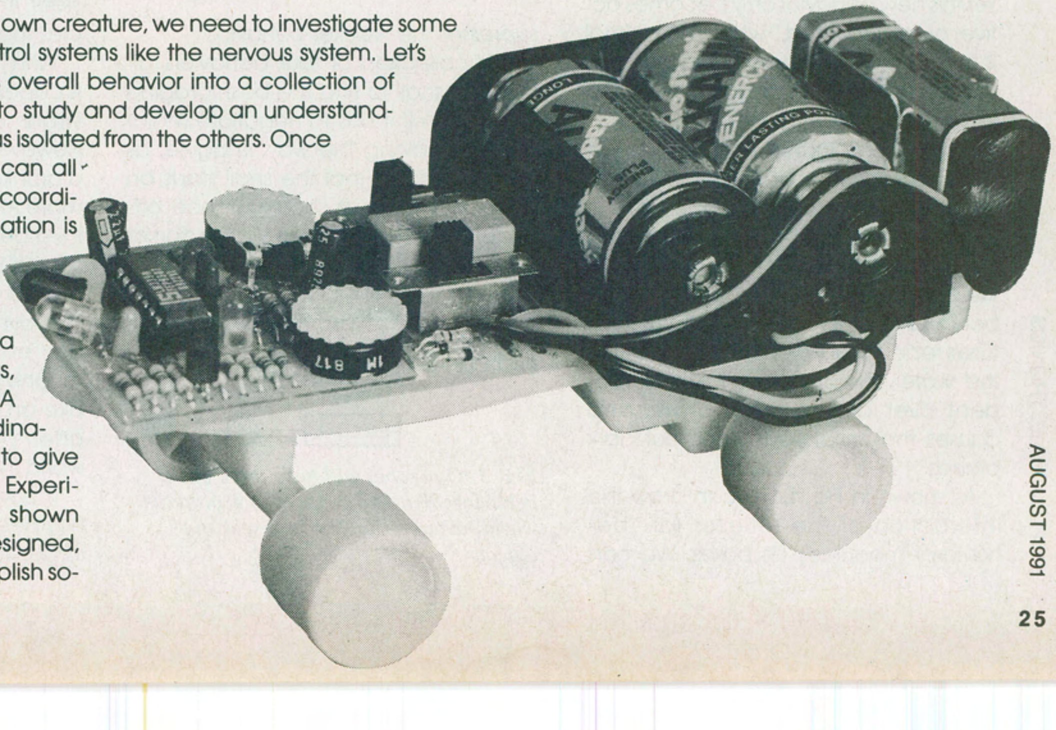
The human brain is very complex. It contains billions of neurons and has trillions of connections between them. Although, many areas have a uniform structure, there are several-hundred architecturally distinct regions. That makes simulating a human brain very difficult.

By contrast, there are many insects and marine animals that have far fewer neurons. Some of them have been studied in detail and scientists have a good idea of how the various components of their brains are connected together. There is also a rich collection of experimental data detailing what sort of behaviors are present in each animal and how various groups of neurons interact to perform the necessary computations. So, at least for now, it's more feasible for us to build robotic models of such simple creatures rather than humans. Also, since man evolved from simpler organisms, the knowledge gained from such an endeavor should ultimately lead us to a better understanding of our own minds.

In this article, we'll investigate the nature and capabilities of elementary, animal-like reflex systems. We'll also show you how to construct an inexpensive mobile robot based on simple animal behavior.

Behavior. Before designing our own creature, we need to investigate some of the principles of natural control systems like the nervous system. Let's start by breaking an organism's overall behavior into a collection of separate reflexes. This allows us to study and develop an understanding of each reflex as though it was isolated from the others. Once each reflex is understood they can all be put together in a model that coordinates their activity. This coordination is necessary to prevent potential conflicts between reflexes.

Each reflex can be modelled as a set of "if-then" rules—*if* a particular circumstance exists, *then* perform a specific action. A simple way to model the coordination of the animal's actions is to give each rule a different priority. Experimental robotics research has shown that such systems, if cleverly designed, are powerful enough to accomplish sophisticated tasks.



The functioning of a rule-based system is best illustrated by an example. Consider the coastal snail. This creature spends its life at the edge of the ocean, eating algae off rocks. The best place for this kind of snail is in a crack right above the waterline. There it can find a rich concentration of food and the creature is in no danger of being dried out by the sun or gulped-up by a passing bird. Unfortunately, the snails are occasionally swept away by a wave, so to avoid starvation they must have some way of seeking out this optimal region again.

Ethological studies have revealed that the snail has two primitive drives: to climb upward and to avoid light. We will refer to these reflexes as "up" and "dark." However, neither of these instincts completely controls the snail's behavior. In fact, there are some situations in which they play no part. For instance, if there is no difference in light intensity between directions, the dark behavior is quiescent and the snail crawls straight upward. Similarly, when the snail is on a more or less flat surface, the up drive is inactive and the snail's direction of travel is determined solely by the illumination gradient.

Overall, however, dark is the stronger reflex. If a very bright light source is present, the snail will crawl away from it even if this means going downward.

Surprisingly enough, if one turns the snail upside down, instead of avoiding light, it will now head toward bright areas. We can say that this is due to a third reflex, "bright," which provides the animal with an urge to seek out light. Since the bright reflex ends up controlling the motion of the animal, it must override the output of the dark module. Yet this new behavior only becomes active, or "potentiated," when the animal is inverted. When the snail is right-side up, the creature acts out one of the lower level behaviors.

There is one further twist to all this. It has been observed that the light-seeking behavior occurs only underwater. If the animal is in air, it will invariably seek out dark areas, even if it is upside down. We can model this by adding another behavior, called "crack," to the creature's repertoire. When the snail is out of the water, this behavior takes precedent over all the other drives and causes the creature to seek out dark places.

As shown in Fig. 1, we can draw the interaction of the reflexes (or "behavioral modules") as boxes. We can

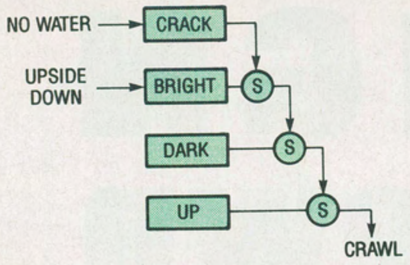


Fig. 1. The overall activity of a sea snail can be broken down into four natural tendencies or behaviors. In some situations, one tendency suppresses another. For example, bright behavior suppreses dark behavior.

indicate the priority of the behavioral modules through the use of circles with an "S" (which stands for suppressor node) in it. In the case of conflicting motion commands, the behavioral module which injects its signal into the side always wins and gets control of the snail's body. The dominant behavior

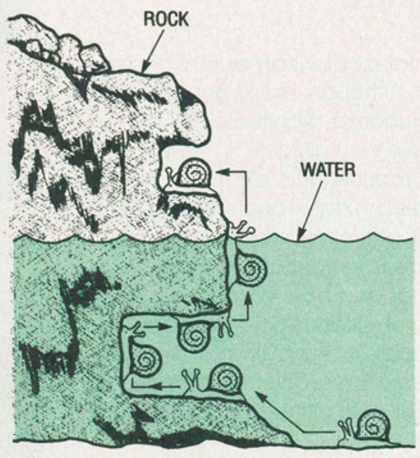


Fig. 2. The priorities placed on a snail's behavior cause it to climb up out of the water, seek a dark crack in the landscape and stay there.

supresses the weaker behavior.

This collection of four behaviors allows the snail to find the best foraging area, even if it has to negotiate major obstacles along the way. Imagine, as shown in Fig. 2, that the snail starts on the ocean floor, a short distance offshore. Since the rocks are slightly darker than the surrounding sand, it crawls along the bottom towards them. Then, when it reaches an outcropping, it starts climbing the face. However, every time it comes across a notch in the rock, it is drawn inward by the darkness. Upon reaching the end of the crack, the snail starts climbing the rear wall and eventually reaches the ceiling. Here, it becomes inverted and thus moves outward toward light again.

Having successfully overcome this impediment, the snail continues climbing toward the surface. When it reaches the edge of the water, it ascends still further until it comes across another crack. As before, the dark-seeking behavior will take over and directs the snail into any crack encountered. However, since it is now above water, the snail does not head upward or turn around when it reaches the back, but instead stays deep in the crack to search for food.

The Robot's Behavior. Using behavior-based control systems we can now design our own synthetic creatures. The one to be described here is called "*Muramator*," which is Latin for "wall lover." As its name suggests, the robot follows the edges of walls and furniture. To design such a robot, we start by breaking its desired "behavior" down into separate parts.

The most primitive behavior we'll call "explore." This module should constantly urge the robot to go forward. While that causes the robot to move around its environment, it also causes the robot to get stuck easily. To prevent this, we'll add another behavior, "avoid," which overrides the output of the explore circuitry. Avoid's job is to steer the robot away from any obstacles that might be encountered.

With just these two behaviors a robot is capable of wandering around its environment for long periods of time. However, it tends to bounce around like a drunken pool ball. To make the robot more responsive to its world, let's add a third behavior, "seek." This module can search for objects and guide the robot toward them.

A dynamic balance between seek and avoid will keep the robot running roughly parallel to the edges of objects. Like an ancient mariner, the robot will attempt to keep the shoreline of its world in sight at all times.

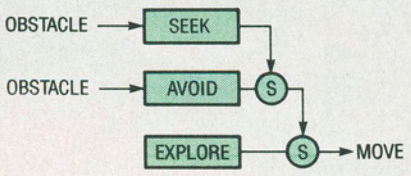


Fig. 3. Our robot will have three basic behaviors that will cause it to explore its world, avoid obstacles, and seek out objects.

The whole algorithm (shown in Fig. 3) has been successfully used by several larger robots. Our abstract specifications now need to be translated into real rules. To do this we need to know the actual perceptual and motion capabilities of our robot. To keep construction simple, for the body we have chosen a commercially available toy vehicle that is able to stop, go forward, or turn in place toward the left. No other actions are possible.

There are 3 different bodies (all available at Radio Shack) that will work with the Muramator circuit. The preferred body is a wire-controlled skate board. This configuration is called the "Whirligig" or WG model. Another option is to build the robot around a wire-controlled dinosaur (Radio Shack No. 60-2284), referred to as the "Dizzy Lizzy" (or DL) model. The DL model has a more appealing appearance, but the WG model has crisper performance. Still another option is the "Piro-jette" (or PJ) model, which uses a wire-controlled Stealth Fighter (Radio Shack No. 60-2305). However, the wings of that model have a tendency to get stuck on obstacles.

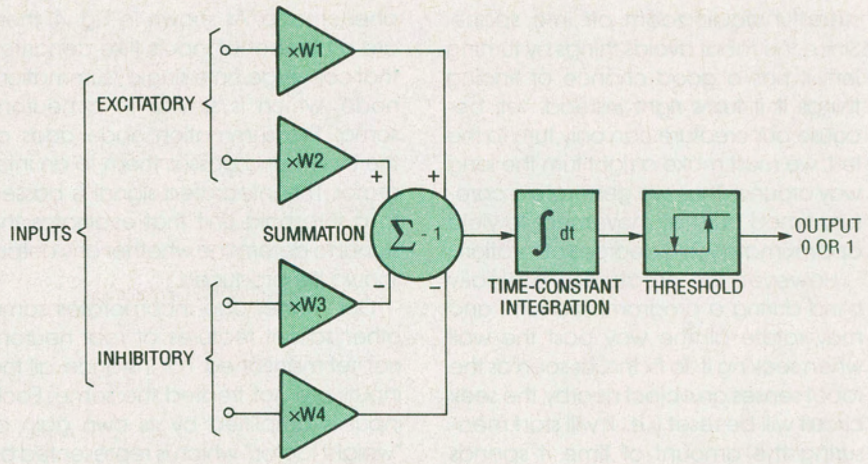


Fig. 4. A neuron receives input from other neurons and ascribes a certain amount of importance (weight) to each. Some weights are negative some positive. It then sums the overall value of the inputs, integrates them, and if the integrated value reaches a certain level the neuron outputs a one.

Since the Whirligig is the technically best model, we'll be covering that one here. You can get further information on using the other bodies from the kit supplier mentioned in the Parts List. Unfortunately, your choice of model might be constrained by which toys your local store has in stock, although they may be able to order the model you want. In any event, most Radio Shack stores stock most of the toys during the holiday season.

For sensing, we'll use a single infrared proximity detector. That device works by emitting a beam of light then looking for a bright reflection. The sensor is able to "see" objects in an almond-shaped region about 3-inches wide by 12-inches long.

The implementation of the first behavior, explore, is trivial: we just run the robot's motor. The next behavior, avoid, is also fairly simple. If the obstacle detector senses anything, we run the robot's motor in reverse. This causes the creature to turn away from the stimulus.

However, for this to work, the obstacle sensor must be oriented properly: On one hand, it is important that the robot have some forward vision to avoid ramming into objects directly in its path.

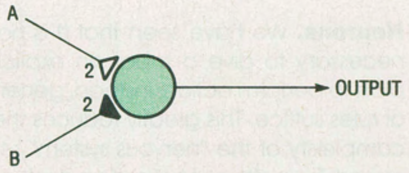


Fig. 5. This symbol of a neuron shows one excitatory input (A), one inhibitory input (B), the synapse (the circle), and the output. Both inputs have a weight of 2 as indicated.

However, if the sensor points straight forward, the robot is likely to side-swipe obstacles. Therefore, we compromise and aim the sensor about 30° to the right of the robot's midline. That naturally makes the robot more sensitive to obstacles on the right; a sensible choice since our robot avoids things by turning left.

The last behavior, seek, is more difficult to instill in the robot. Unfortunately, we can't directly determine where the wall is and how to steer the robot toward it. The robot only looks in one direction, and anytime it sees something, it is programmed to turn until the sensor reading disappears. However, if the proximity detector is active a large percentage of the time, we can assume that the robot is still near obstacles. Open spaces, on the other hand, are characterized by the absence of any sensor readings.

That forms the basis for our seeking strategy: When the robot has not seen anything for awhile, it spins around in an attempt to locate the edge of the world again. Notice that if the new seek behavior is omitted, the creature will not turn back toward the wall, but in-

PARTS LIST FOR THE MURAMATOR

SEMICONDUCTORS

U1—LM339 quad comparator integrated circuit
D1—1N4001 rectifying diode (not used in the WG model)
D2–D11—1N914 small-signal diode
Q1, Q2—MPS2907 PNP transistor
Q3—TIL414 IR phototransistor
LED1—SEP8703-1 infrared emitting diode
LED2, LED3—Light-emitting diode

RESISTORS

(All fixed resistors are 5%, ¼-watt units.)
R1–R3—1-megohm
R4—470,000-ohm
R5–R7—220,000-ohm
R8, R9—100,000-ohm
R10–R14—47,000-ohm
R15, R16—10,000-ohm
R17–R20—1000-ohm
R21—330-ohm
R22–R23—1-megohm potentiometer

CAPACITORS

C1—4.7-µF, 35-WVDC, electrolytic
C2—0.1-µF ceramic disc
C3, C4—.01-µF metalized film
C5—470-pF metalized film
C6—22-µF, 35-WVDC, electrolytic

ADDITIONAL PARTS AND MATERIALS.

K1—12-volt DPDT relay
S1—DPDT subminiature slide switch
S2—SPDT subminiature slide switch
B1—9-volt transistor-radio battery
B2, B3—1.5-volt C-cell battery
Printed-circuit board, wire-controlled skate board (Radio Shack No. 60-2298 or equivalent), 14-pin IC socket, 9-volt battery holder, 9-volt battery clip, 2 C-cell battery holder, wire, solder, etc.

A drilled and etched printed-circuit board with instructions is available for $25 (post paid) from Johuco Ltd., Box 390, Vernon, CT 06066. CT residents must add appropiate sales tax.

stead it would zoom off into space. Since the robot avoids things by turning left, it has a good chance of finding things if it turns right instead. Yet, because our creature can only turn to the left, we must make a right turn the long way around. Thus, we generate a carefully timed burst of movement to yield approximately 270 degrees of rotation.

However, the creature is basically blind during a programmed turn, and may rotate all the way past the wall when seeking it. To fix this, as soon as the robot senses an object nearby, the seek circuit will be reset (*i.e.*, it will start measuring the amount of time it spends near objects again). That not only causes the robot to stop turning, it also synchronizes the unit so that it correctly measures the time since the last obstacle sighting.

Neurons. We have seen that it is not necessary to give a robot an explicit plan to perform action; instead, general rules suffice. This greatly reduces the complexity of the "nervous system" required. Typically, a set of ordered reflexes is adequate for simple navigation, but what mechanisms make these reflexes possible? In animals the answer is, of course, neurons. So, as a guide to implementing our "creature" electronically, let us see how real neurons work.

Neurons communicate via electrical/chemical impulses. To transmit a signal, one neuron releases a puff of a specific chemical across a gap (called a "synapse") toward the next neuron. The next neuron absorbs the chemical into one of its inputs (which consist of tree-like structures called "dendrites"). This substance briefly opens a number of ion channels in the receiving neuron's cell membrane, and the resulting flow of charge carriers causes it to act like a miniature battery.

The combined charges are funnelled back to the body of the cell (known as the "soma"). If enough dendrites are activated, the neuron gradually becomes more and more electrically charged. When there is enough accumulated potential, the neuron spontaneously generates its own series of impulses which travel down an output fiber (the "axon"). Eventually this signal impinges on the inputs of succeeding neurons and a similar series of events takes place.

A Neuron Model. Our model of a neuron takes into account all of these phenomena. As shown in Fig. 4, there are a number of inputs (like dendrites) that converge on a single "summation" node, which is similar to a neuron's soma. The summation node adds all the inputs and passes them to an integrator. The integrated signal is passed to a threshold unit that evaluates the signal to determine whether any output should be produced.

Our model also incorporates some other salient features of real neurons not yet mentioned. For instance, all the inputs are not treated the same. Each input is amplified by its own gain or "weight factor," which is represented by the amplifier symbols shown. In this way, inputs with a large weight factor will influence the neuron more than inputs with a small weight factor. This reflects the anatomical fact that certain synapses in animals are more transmissive than others.

In addition, we also show inputs with negative weights (the lower two inputs). These correspond to "inhibitory connections" often observed between actual neurons. They can suppress the action of neurons.

Another characteristic of neurons that our model contains can be called "leakage." You can think of a neuron as a tub being filled with water. There are a number of hoses of different sizes feeding into the tub (the excitatory inputs) as well as a number of drain spouts (the inhibitory inputs). After the tub (neuron) has filled up, it fires. However, once "full," the neuron would fire at the slightest input so the tub has built-in leakage to gradually drain the remaining water.

To reproduce this effect, the summation node contains a "−1" term. This is essentially an inhibitory input that is always on. When none of the other inputs are active, the sum is now negative so the integrated excitation always decreases toward zero. However, we never let the value of the integral go negative; we restrict the integrated excitation to be between 0 and 1 at all times. This reflects the fact that we can not fill a tub above its rim or drain it below its bottom.

Interestingly, while real neurons also have limits to the voltages that they can produce, the voltage can drop below the usual "rest voltage." In animals, it is possible, and sometimes computationally useful, to discharge a cell below this neutral level. Such a condition makes it harder for later inputs to trigger the neuron.

The final twist to our neural model is the nature of the threshold circuit. We use a device known as a "Schmitt trigger" instead of a simple comparator. This device has two thresholds, a high one for rising signals and a lower one for falling signals. In our neuron model, the output switches on when the value of the integrated excitation reaches 1. The neuron will continue to output a 1 until the integrated output descends to 0 again.

This is much like the thermostat in a typical house. If you set the temperature to 70°F, the furnace will not turn on until it gets as cold as 68°F or so. Then it will proceed to warm the house until the temperature slightly exceeds 70°F, say up to 72°, before shutting off. Although this feature is not totally accurate from a neurological point of view, it's a convenient feature to have.

Supplementary Reading.

Vehicles, Valentino Braitenburg, MIT Press, 1986.

Minimalist Mobile Robotics, Jonathan Connell, Academic Press, 1990.

Mind Children, Hans Moravec, Harvard University Press, 1988.

The Study of Instinct, N. Tinbergen, Oxford University Press, 1951.

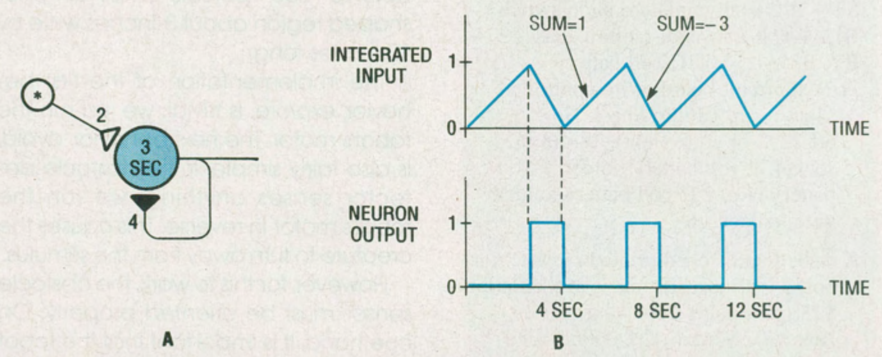


Fig. 6. The simple two-neuron system shown in A, can perform the operation of an oscillator. Its internal integration signal and final output are shown in B.

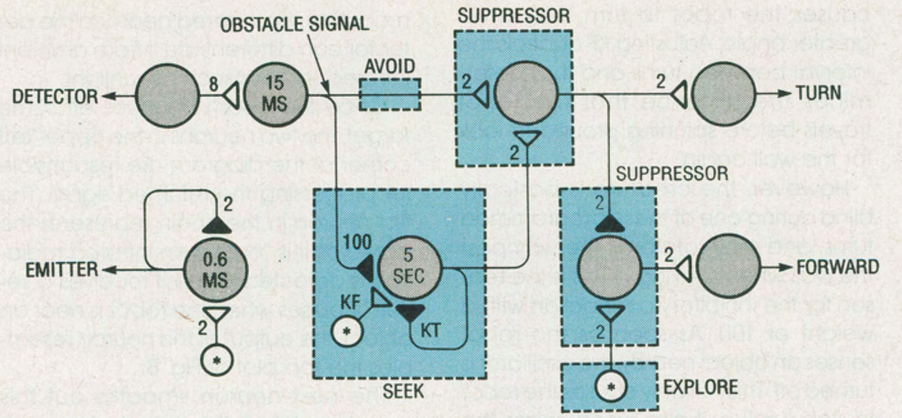


Fig. 7. The robot's 4-behavior activity can be emulated using 11 neurons. Can you find the two oscillators here? (Hint: look for feedback loops).

Neuron Examples. To see how such simulated neurons are used, consider the symbol shown in Fig. 5. Here, we depict the body of the neuron as a circle with an arrow coming out of it to represent the output. Input terminals are shaped like the bell of a trumpet and have their associated weight written next to them. White terminals are excitatory, whereas black ones are inhibitory. Since the structural details do not matter, we show all the inputs impinging directly on the cell body.

This neuron will only generate an output if input A is active and B is not. First, suppose neither A nor B is active. The input sum is:

$$0\times2 - 0\times3 - 1 = -1$$

(the last term comes from the leakage property of the model). This negative result causes the accumulated value inside the neuron (if any) to decay until the output switches to zero. Now suppose A comes on. The new sum is:

$$1\times2 - 0\times2 - 1 = +1$$

If we set the time constant of the neuron to be very short, the output will almost immediately switch to one. Finally, imagine that B comes on as well. The computed sum for this case is:

$$1\times2 - 1\times2 - 1 = -1$$

which forces the neuron to turn off.

A more complicated example is the oscillator shown in Fig. 6A. The central part of this configuration is similar to an AND-NOT gate. Here, the neuron itself supplies the inhibitory input, while another neuron provides excitation. The asterisk inside this auxiliary neuron indicates that it is of a special type that is always on. Assume that the central neuron's internal potential starts off at zero and that its output is off. The initial input sum is:

$$1\times2 - 0\times4 - 1 = +1$$

Thus, the integrated input starts climbing slowly as shown in the upper plot of Fig. 6B. The value written inside the neuron tells how long it takes for the neuron to fully charge with an input sum of 1. As can be seen, it takes 3 seconds for the integrated value to reach one.

Once the integral has reached the prescribed threshold level, the output of the neuron comes on. That is shown in the lower plot of Fig. 6B. However, this changes the overall sum sent to the integrator. It is now:

$$1\times2 - 1\times4 - 1 = -3$$

so the neuron's internal charge starts to decay. Yet, because of our special threshold stage, the neuron's output value remains at one until the integrator output again reaches 0. This happens one second after the output comes on. Once the output turns off, the whole cycle repeats.

As can be seen, the resulting output is a pulse train with a period of 4 seconds. The frequency of a cycle, as well as the width of the on and off portions, can be changed by adjusting the input weights because higher sums (whether positive or negative), alter the output in less time.

The Neural Network. The total collection of behaviors required for Muramator can be emulated by a network of 11 neurons as shown in Fig. 7. For motor control, the robot's brain has one neuron that drives the robot forward and another that causes it to turn. That makes the explore behavior easy to implement: a continuously active neuron provides input to the "forward" neuron. The activity of the avoid behavior is also simple: it just activates the "turn" neuron when the robot "sees" something. That is particularly easy since the appropriate obstacle-detection signal is directly available at the output of the neuron labeled "15 ms" (to be described later).

Notice that neither explore nor avoid is directly connected to the motor neurons. Instead, each sends its command via a single intermediate neuron. When designing large networks it is good practice to insert an "interneuron," such as this, to serve as an interface point where higher level signals can be injected. In this case, avoid indirectly suppresses the default forward drive by inhibiting the interneuron that would carry out the explore behavior. That blocks the forward behavior and allows the more important avoid module to substitute its own instructions. Tech-

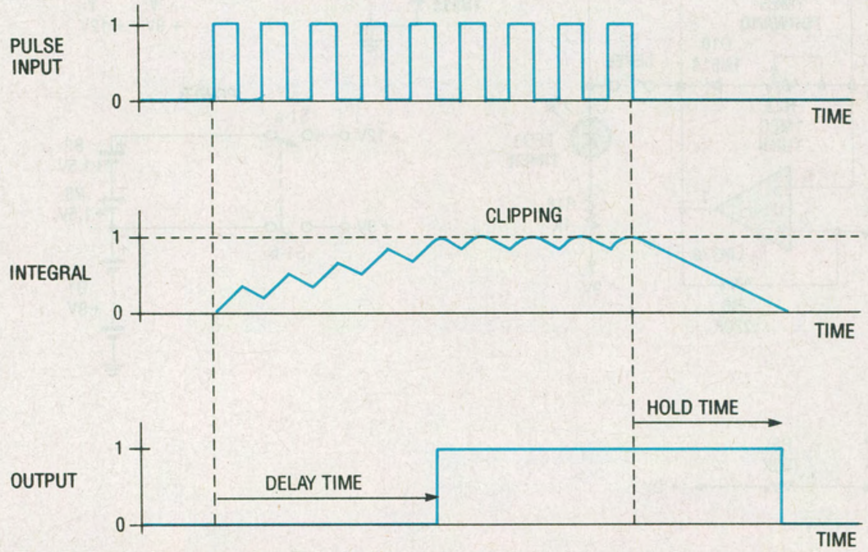


Fig. 8. Each detection of an infrared pulse (the top graph), causes the neuron to charge (middle graph). When the neuron reaches the threshold value, its output goes high and remains high till the neuron completely discharges (as shown by the bottom graph).

nically, the avoid circuit should really have two interneurons: one to generate the turn command and one to suppress the forward command. This arrangement would let us cascade suppressor nodes so that the most important behavior could suppress all the others with a single connection.

The third behavior, seek, is composed of two additional neurons. Neglecting the weight-100 input, we can see that the remaining structure is identical to the basic oscillator presented earlier. The output of this unit feeds directly into the avoid interneuron and thus causes the robot to turn. There is no interneuron involved in this pathway because there are no higher level behaviors that might need to suppress seek. An important feature of this module is that the weights of both the excitatory and inhibitory connections (KF and KT) of the oscillator can be varied. Adjusting the KT weight makes the output of the oscillator remain high longer and thus causes the robot to turn through a greater angle. Adjusting KF controls the interval between turns and thus determines the distance that the robot travels before spinning around to look for the wall again.

However, the creature is basically blind during one of these programmed turns, and may rotate all the way past the wall when seeking it! This is the reason for the inhibitory connection with a weight of 100. As soon as the robot senses an object nearby, the oscillator is turned off. This not only causes the robot to stop turning, it also synchronizes the unit so that it correctly measures the time since the last obstacle sighting.

The remaining four neurons form the proximity-detection subsystem. The neuron in the lower-left corner of the diagram is directly connected to the infrared emitter and generates the outgoing signal. That neuron functions as a simple oscillator and produces a symmetric square wave. It is necessary to modulate the infrared beam so the detector can differentiate it from ambient infrared sources such as sunlight.

Once the beam bounces off some target, the two neurons in the upper-left corner of the diagram are responsible for processing the returned signal. The first neuron in the chain represents the detector. It is "on" when infrared radiation is detected. Since it receives a series of pulses when the robot is near an object, the output of this neuron resembles the top plot of Fig. 8.

The next neuron smooths out this waveform. When the detector neuron is on, the input to the second neuron is:

$$1 \times 8 - 1 = +7$$

so the internal potential of this neuron increases. Between pulses the input sum is simply -1 and thus the potential slowly decays. The charging of the neuron is shown in the middle plot in Fig. 8. As can be seen, it takes several pulses to charge the neuron up to a high

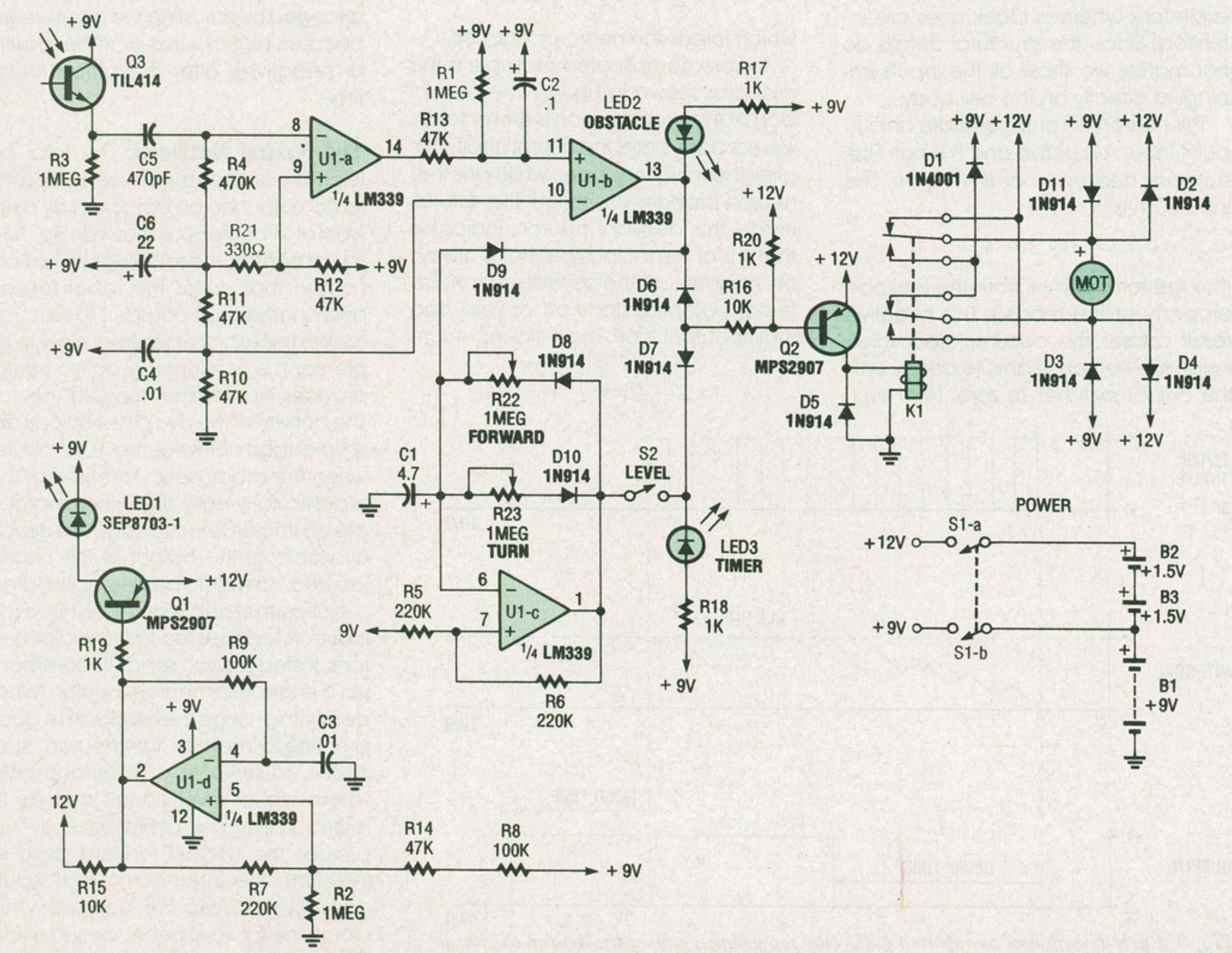


Fig. 9. All eleven of the robot's neurons are emulated by this circuit. Some neurons are based on comparators, while others are implemented with simple diodes.

enough level to generate an output, which is shown in the bottom plot.

This is useful for rejecting noise pulses. Similarly, the slow decay constant allows the system to "fly-wheel" through signal dropouts: the output of the neuron will remain on for several cycles after the stimulus vanishes. This same phenomenon can also be used to artificially increase the size of the avoidance turns the creature makes.

Circuitry. The final step is to compile all of this into circuitry. The actual schematic for the creature is shown in Fig. 9. In terms of actual circuitry, the 11 neurons in our model are implemented in a number of different ways. Some are modelled with voltage comparators, some with diode logic, and some with electro-mechanical devices. Keep this in mind as we describe the electronics for each of the creature's component behaviors.

Let us start with the explore behavior module. The explore reflex is incorporated directly into the relay circuitry (right side of diagram). Normally the robot's motor is connected so it runs forward. However, whenever the relay is energized by Q2, the voltage applied to the motor is reversed and the creature turns instead. An extra diode, D1, can be inserted to slow the creature down, if necessary. The diodes around the motor (D2–D4, and D11) and across the relay's coil (D5) serve no behavioral function, they just clamp inductive spikes to the power-supply rails.

Thus, the explore neuron, the first suppressor, and the turn and forward neurons are all emulated in this piece of circuitry. The transistor for the relay can be considered the equivalent of the second suppressor node in the neural diagram, and diodes D6 and D7 act as the two excitatory connections to this interneuron.

The circuitry for the LED oscillator (see the lower-left corner of Fig. 9) corresponds directly to the neural model. Here, the feedback from a comparator's output to its positive input provides the "hysteresis" needed by our dual-threshold neural model, while the resistor and capacitor on the negative input form the required integrator. The 10k resistor to +12 volts mimics the action of the necessary always-on neuron. This arrangement generates a squarewave that is amplified by another MPS2907 transistor to drive the IR LED. That unit radiates infrared energy (much like a TV remote control) into the

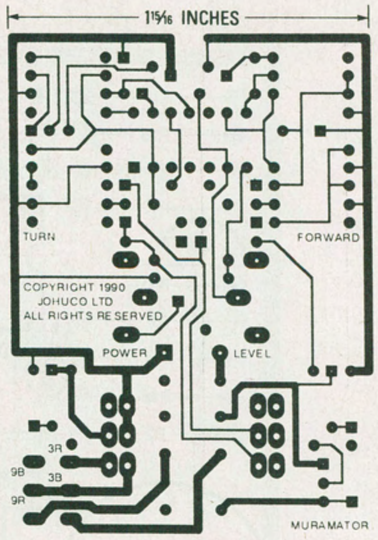


Fig. 10. This foil pattern is for the component side of the Muramator circuit board. The labels for the potentiometers and switches will help you use those controls.

environment, which then bounces off nearby objects.

The next behavior, avoid, switches the relay on using the signal from the infrared proximity detector. The circuitry for the detector is shown in the upper-left side of the diagram. The reflected IR signal biases a TIL414 phototransistor, which applies a small voltage across a 1-megohm resistor, R3. That signal is AC-coupled via a 470-pF capacitor into a simple threshold-detector formed by one section of the LM339 quad comparator. This is the "detector" neuron from the neural-network diagram. The capacitors on the voltage divider con-

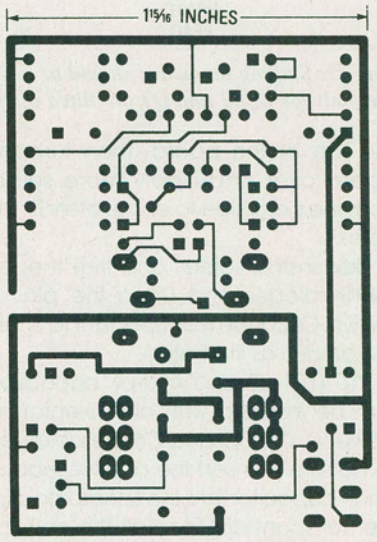


Fig. 11. This foil pattern for the solder side of the robot must be properly registered with the component-side pattern. Remember to connect all vias when the board is finished.

nected to the positive input of the comparator help stabilize the reference voltage against transients caused by switching the IR LED on and off, and operating the motor.

An inverted version of the obstacle-detection signal then enters a low-pass filter formed by R13 and C2. Since the LM339 has open-collector outputs, the decay rate of the voltage across C2 is governed solely by R1. It takes several pulses to discharge C2 sufficiently enough to generate an output. This is useful for rejecting noise pulses. As you'll see, the slow decay constant allows the obstacle-detection signal to remain on for a period of time after the stimulus vanishes. By substituting a larger value for C2, you can increase the size of the avoidance turns the "creature" makes.

The RC-filtered voltage enters U1-b, which also acts as a simple comparator. The threshold level for this unit is obtained from the same voltage divider used for the first stage. This comparator combined with the previously described low-pass filter corresponds to the second detection neuron. The total result of all this processing is a clean digital signal indicating when an obstacle is present. It is connected to a red LED that indicates when the robot is "seeing" something.

The last of the creature's behaviors, seek, is built from an oscillator similar to the one used to generate the IR beam. The oscillator has an active-low output: when pin 1 of U1-c is at zero volts, the robot should turn. Note that the oscillator's output is connected back to the integrating capacitor (C1) through two diodes. These diodes steer current through the potentiometers so that the top potentiometer controls the capacitor's charging time and the lower potentiometer controls the discharge rate.

Notice also that in the seek circuitry there is a diode (D9) descending from the obstacle-detection indicator, LED2. Since D9 has negligible resistance, it can "instantly" discharge capacitor C2 when the creature detects an obstacle. This connection models the weight-100 input in our neural design and is used to properly synchronize the oscillator as mentioned earlier.

The design also must have an arbiter that can combine the commands from each of the basic behaviors. To this end, both the seek oscillator and the avoid circuitry are connected to the relay's driver transistor through diodes D6 and D7. This arrangement is used to model the top "suppressor" neuron back in Fig. 7, as we mentioned earlier. Because both the obstacle detector and central oscillator have active-low outputs, these diodes form a logical OR gate. If the creature either sees something (LED2 turns on) or the turn timer kicks in (LED3 turns on), the relay will be activated and the creature will turn in place. The switch labelled "level" was installed to selectively disable the seek oscillator. That lets you investigate how the creature acts without this behavior. Of course the robot won't do anything at all unless its built, so let's get to that now.

Circuit Construction. Because of the complexity of the wiring, it is recommended that you use a printed-circuit board to wire the components together. For those of you that wish to make your own boards, the foil pattern for the component side of the board is shown in Fig. 10, and the foil-pattern for the other side is shown in Fig. 11. Since the board is double-sided, if you chose to make your own, you must solder tiny pieces of bus wire into the via holes and solder component leads to the pads on both sides of the board where applicable. Keep that in mind when soldering all components.

If you do not want to etch the circuit yourself, a drilled and etched plated-through printed-circuit board with instructions is available from the kit supplier mentioned in the Parts List.

With the board all ready to go, start by installing the 14-pin DIP socket at the location for U1 as shown in the parts-placement diagram of Fig. 12. Be sure to mount it and all the other components on the top side of the board (the side with the writing on it). Note that one end of the socket has a small notch. It should go toward the square pad on the printed-circuit board. Now flip the board over and solder its pins to the pads.

Now, guided by the parts-placement diagram, install the fixed resistors. Take each of the trimmer potentiometers and flatten the crinkles in its leads (if any) by squeezing the lead with needle-nose pliers. Now bend the two side leads of each potentiometer sharply downward away from the plastic adjustment dial. Next, gently bend the center lead of the potentiometer in the same direction. The bend should be made about halfway out from the body. Insert the potentiometers in the positions shown and solder them from

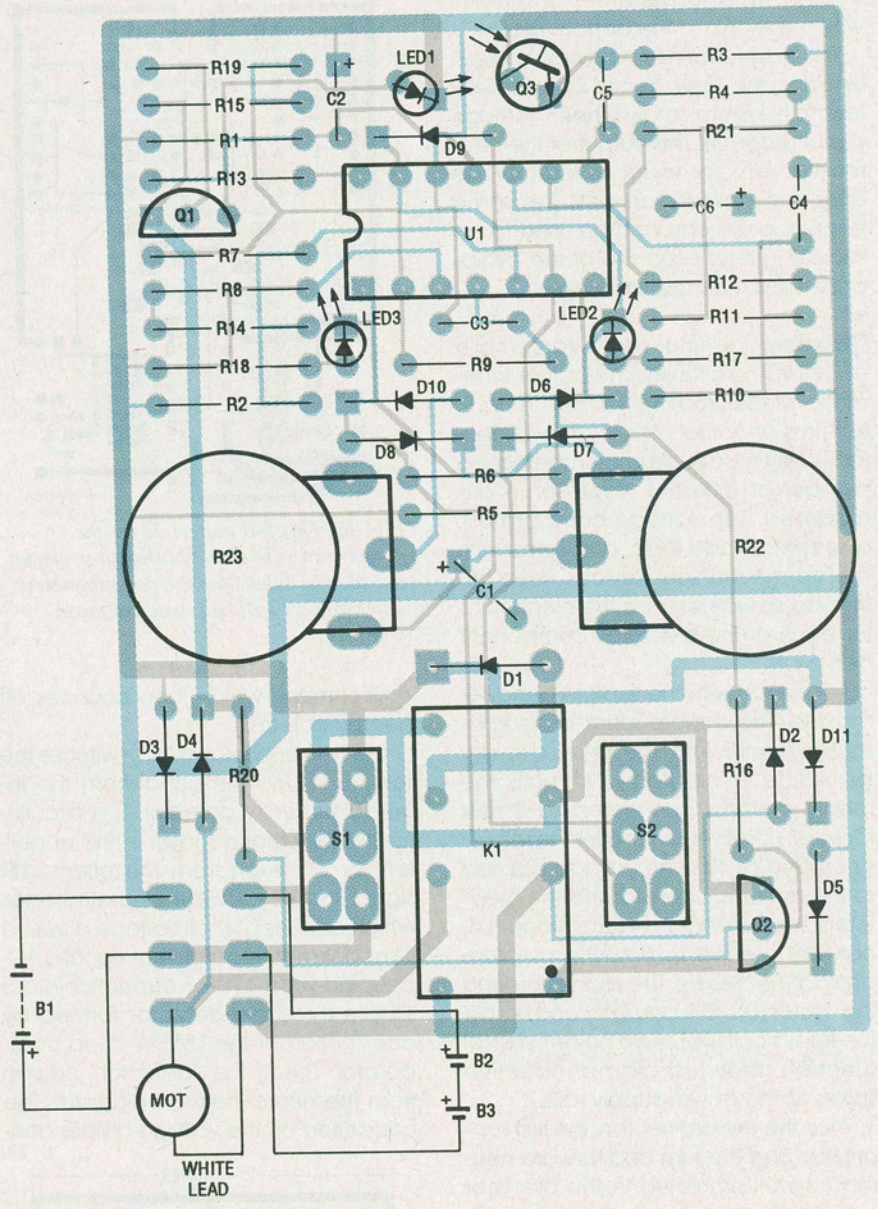


Fig. 12. Stuffing the board should be easy with this parts-placement diagram. Note that the pads for LED1 and Q3 are tilted at 30°.

the top of the board. Then turn the board over and apply more solder from the back side to completely fill the holes.

Next install, solder, and trim the ten 1N914 diodes and D1 in the places shown. Of course it is important to orient the diodes as illustrated.

The 0.01-µF and 470-pF capacitors can be installed with any orientation. However, C1, C2, and C6 are polarized so be sure to insert the positive lead of each capacitor into the square pad for the component. Once all the capacitors are installed, bend, solder, and trim their leads.

The orientation of the transistors is important. Insert them positioned as shown in the parts-placement diagram. Sometimes the transistors come in a metal case rather than plastic. For this type, insert the device so that the lead closest to the tab on the case goes into the square pad.

Next install and solder the DPDT relay. The two plain LED's go near the corners of the 14-pin socket and should be installed next with the proper orientation.

Now, install the switches on the top side of the board. Do not push them completely down to the board, instead leave a tiny air gap under each switch to prevent the case from shorting out any of the copper traces nearby.

Cut a 3/8-inch by 3/4-inch wide piece of black electrical tape and wrap it around the body of the TIL414 phototransistor. It should be used to cover only the sides of the component leaving the rounded end exposed. Install and bend the leads of the IR devices so that the body of each one lies along the surface of the board. Note that the orientation of the holes will place them at a 30° angle with the centerline of the board.

Next insert the LM339 integrated circuit into the socket with pin 1 oriented near the square solder pad used for the socket.

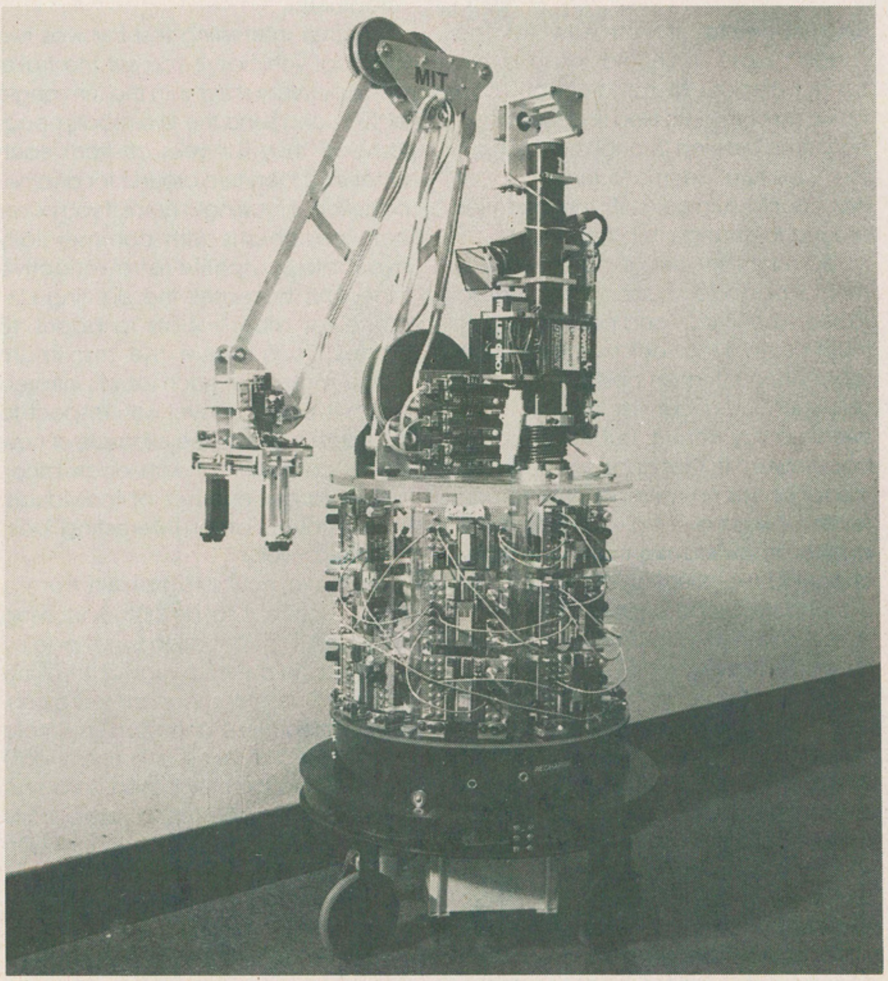
Take a 9-volt battery snap-on connector and cut its wires down to 4 inches. Strip and tin the ends and solder the wires into the lower-left corner of the board. The red wire should go into the hole marked "9R" and the black wire should go in the hole marked "9B."

If the C-cell holder you get has no leads of its own, you will have to add them. You can use a 3-inch section of the toy's wire-control cable for that. Strip and tin both ends of the two wires you use and attach them to the terminals on the holder.

If your battery holder has leads, then trim them to 3 inches, and strip and tin the ends. Whether initially present or not, attach the lead from the positive terminal to the pad marked "3R," and connect the negative lead to the pad labelled "3B."

Checking the Circuit Board. Now that all the components have been installed, double check to ensure that everything is in the right place and oriented in the correct direction. Look for the dark bands on the diodes, the flattened rims of the LED's, the markings on the capacitors, the notched end of the chip, and the flat sides of the transistors.

Now turn the board over and check all the solder joints. Except for 2 holes in the lower-left corner of the board, all holes should have something soldered into them. Check to make sure that all the leads are securely soldered in place. Also make sure you have used enough solder on each joint—you should not be able to see the edge of the hole underneath.


This robot was designed and built by the author using the principles discussed in this article. Named Herbert and seen on PBS's "Discover" program, its function is to collect empty beverage cans.

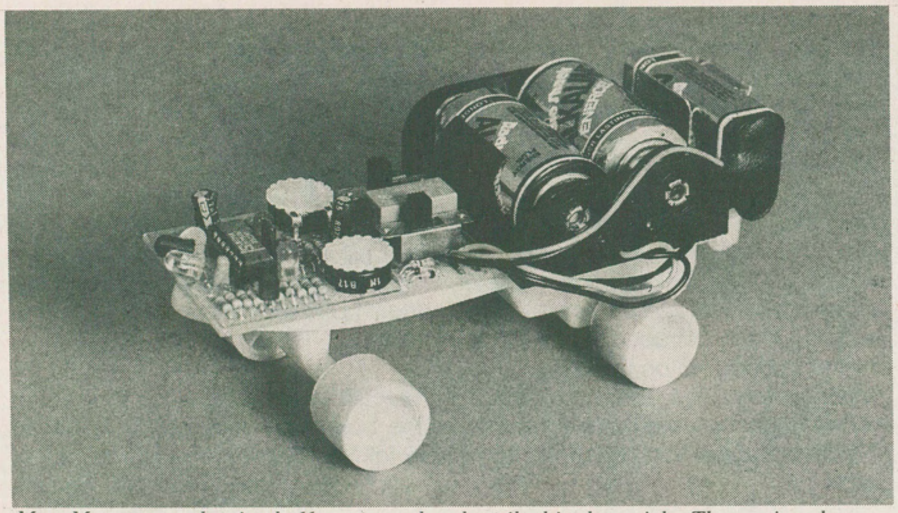
Also check the bottom of the board for solder bridges. There is only one place on the board where adjacent pads should be connected: beneath two terminals of the power switch. If any other two pins have a blob of solder between them, remove it by heating the bridge then tapping the board against the table. If you find any other suspicious-looking connections, heat them up to see if they are actually mistakes.

Electrical Tests. Now it is time to test the basic operation of the board so install the batteries. Start with the power switch down (away from the potentiometers), and the level switch up. In a dark room, you should see a dim orange glow inside the IR emitter. If you don't, check the orientation of the IR LED and the transistor at the top of the board.

Now put your hand in front of the IR components. The obstacle LED should come on and go off again when you move your hand away. If it doesn't, check to be sure the LM339 is inserted properly and that C2 is connected with the correct polarity. If the LED lights at the wrong time, check its orientation. You should also hear the relay click every time the LED changes state. If not, check the orientation of Q2 and D5; those are located in the lower-right corner of the board.

Next, adjust both potentiometers so that they are in the middle of their ranges (arrows point toward the left and right edges of the board) and then move the Level switch down. The timer LED should now blink periodically and the relay should click. If the light doesn't blink, check that the LM339 chip is inserted properly. If there are no clicks, check the orientation of the diodes near the center of the board. Next, test whether rotating the turn potentiometer adjusts how long the light stays on. Also test to find out whether adjusting the forward knob controls how long it is between blinks. If these controls seem reversed, check the orientation of the timer LED.

Finally, test the synchronization. When the timer LED is on, move your hand into the detector's field of view. The timer indicator should immediately go out and stay off until you move your hand away and the obstacle indication disappears. If not, check the orientation of the diodes near the center of the board and look for possible cracks in them.


Meet Muramator, the simple 11 neuron robot described in the article. The version shown here is built on the Radio Shack Skateboard chassis.

Final Assembly and Calibration. The circuitry as well as the appropriate batteries now need to be mounted on the robot's body. Start by removing the human figure from the top of the skateboard. It is held on by two screws through its feet. To get to the screws, turn the vehicle over then remove the 3 screws that hold the rear wheel assembly onto the board. After removing the figure, reassemble the vehicle being careful to keep the white plastic gears firmly seated in their slots.

Next cut the wire-control cable at about 6-inches from where it exits the back of the board. Strip and tin the ends and solder them into the circuit board. The one with the white streak should go to the hole near the center of the board while the other wire should go to the hole in the corner. Now use double-sided foam tape to mount the U-shaped metal battery clip to the rear of the board.

Affix the C-cell holder to the skateboard directly in front of the 9V battery clip using double-sided foam tape. As the last step, use an elastic band to secure the circuit board to the vehicle. The rubberband is only a temporary restraint; once you are sure that everything works properly, you can use a piece of double-sided foam tape instead.

Once the electronics are mounted on the robot, it is necessary to calibrate the "creature." Carefully align the two IR components so that they point exactly parallel to each other and about 30° off the centerline of the robot body. Then turn the robot on and place it on the floor. Check to see that the robot runs forward when it is in an open space. If the obstacle-detection indicator is always on when you put the robot on the floor, tilt the IR components slightly upwards until the light goes off. Next, hold the robot and move it to within 8-inches of an obstacle. Verify that the obstacle LED lights up and that the robot's motor runs in reverse.

Experiments. If you now let the "beast" loose. It should try to avoid things in its path. Muramator works best on wood or smooth tiled floors—it has a hard time plowing through carpeting. If the "creature" seems to lurch a lot or has trouble turning, shift the batteries around to change its balance.

You might try designing an experimental obstacle course for the vehicle to see what it can and cannot do. For instance, with the seek switch off if the robot encounters an obstacle that juts out, such as a convex corner, it will swerve away from it, but never come back toward it. That happens because the robot has no memory of which direction it was travelling so it cannot get back on track. You might also want to try changing the direction of the IR components, or purposely mis-align them, to see how these parameters alter the creature's behavior.

Switch in the second layer of control by sliding the Level switch on. Start with both potentiometers midway through their ranges. If you keep Muramator in a large open area, the timer LED should flash at regular intervals. When this light is on, the robot should perform some sort of spin. Once the circuitry appears to be working properly, adjust the Forward potentiometer until the robot goes about 8 inches between turns. Then adjust the Turn potentiometer so that the robot makes about ¾ of a revolution (270°) every time it turns. Muramator will now not only avoid objects, but turn around if it hasn't seen anything in a while. Together, these reflexes cause it to follow along walls and to circle any post-like object it sees.

Changing the values of the two potentiometers can produce noticeably different results. Note that if the turn timer is set for too long an interval, the robot will only regain the wall after a number of turns. If the obstacle had been a post rather than a corner, the robot might not have found it again at all. On the other hand, consider the scenario in which the travel distance is set too long; the robot may wander way off into the middle of the room when it encounters a corner. The first travel leg after an avoid turn is often longer than the succeeding ones. Thus, even though the robot heads back in the right direction, it will be too far away to see the wall again. Yet this can be a useful feature in a world with more than one robot because corners then become a natural meeting place for the "creatures."

Another interesting test involves two identical vehicles. If you set the travel distances very short with the turn angle at 180°, and send the two robots head-to-head, they will veer off from each other and then turn around for another pass. Robot jousting! It helps if you cover both opponents with commercially available Scotchlite retro-reflective tape. This increases the sighting distance for other robots to about 18 inches. To achieve the maximum range, make sure each robot's infrared beam is pointing level with respect to the floor. This is just one example of how robots can interact with each other. With a larger number of individuals, there maybe other interesting possibilities as well.

We have now progressed from a vague concept to an actual working robot. This transformation was made by applying the methodology of breaking an activity into component behaviors. We codified these behaviors as simple situation-action rules and finally cast the rules into circuitry based on a simple neural model. These same steps can be applied to other creatures with other behaviors. You might try designing some yourself, at least on paper. Incrementally extending this line of research to larger, more complex creatures is a promising path for developing a deeper understanding of how the human mind itself works. ■