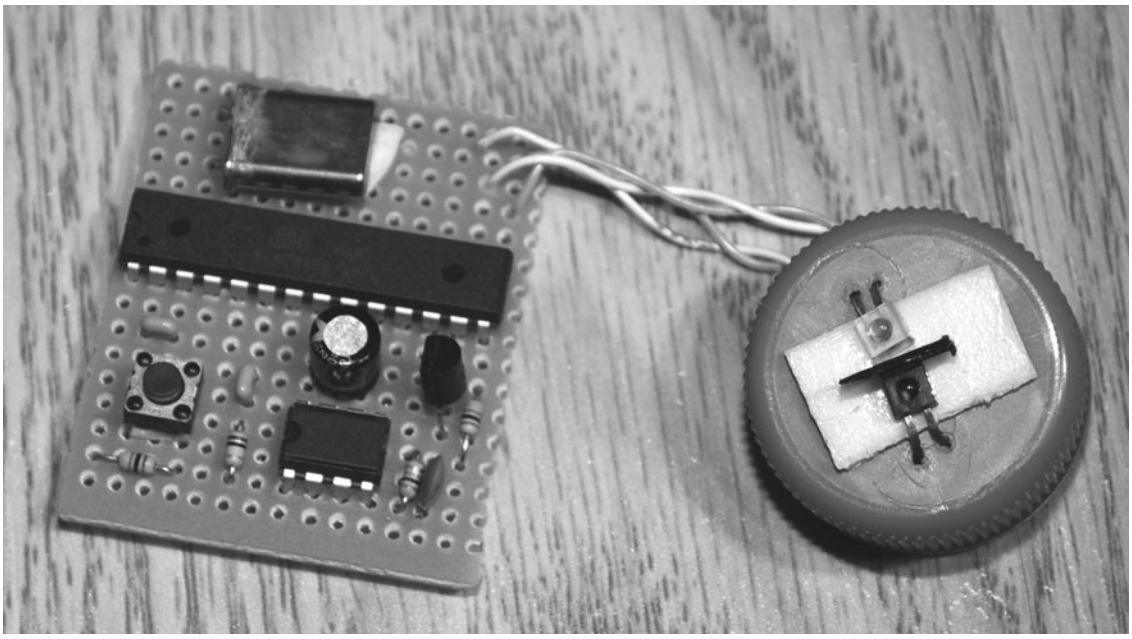


Figure 11-7 *Placing the circuit on a perf board.*

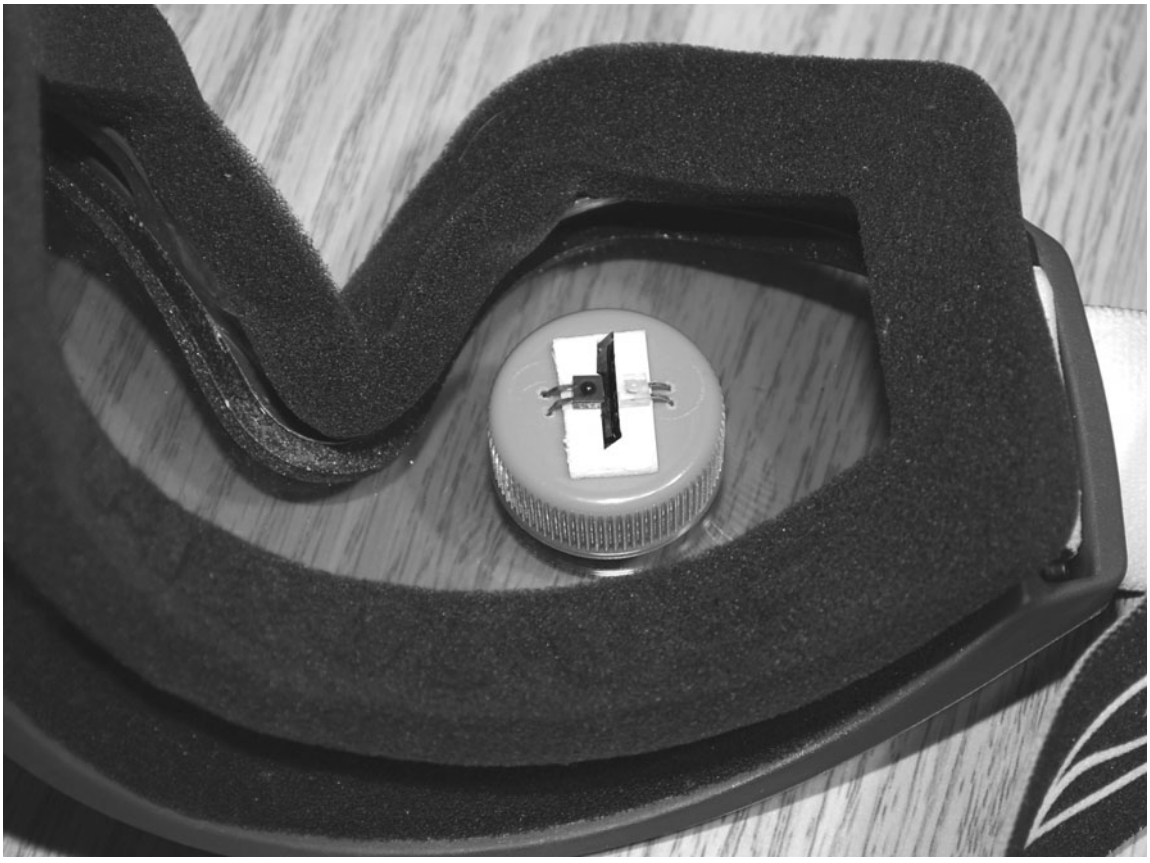


Figure 11-8 *Finding the optimal sensor position.*

directly to the inside of the lens, but it was a bit tall, and by cutting a hole in the lens, I had some control over the angle if I decided it needed to be adjusted later.

With the bottle-cap sensor installed in a hole through the ski mask lens (Figure 11-9), it was easy to set the distance as well as the angle so that an optimal response could be received from the infrared light reflecting off the user's eyelid. Again, this was done in a dark room while looking at the indicator LED on the breadboard circuit and moving my eyes back and forth until the LED gave the greatest response. If you find it annoying to look through one eye in the dark and tweak the variable resistor, then find a human helper to wear the mask as you fine-tune the system. Most likely the adjustment will work for all your test subjects because all human faces have the same basic geometry. Once you have the system mounted to your goggles or mask and working properly, you can go ahead and mount your components on a more permanent home for installation onto the mask or into some small box.

The completed optical-based REM-detection system is shown mounted to the ski mask in Figure 11-10. I split the batteries up to balance the weight on the mask and placed the circuit board in the center for easy access to the reset button in case of false triggering. The mask certainly was comfortable and worked well in all positions except for face down, of course. I decided to keep the lens clear rather than paint it black so that I could get up at night and move around without having to remove the mask and then go through the microcontroller test procedure each time. Adding the circuit board into a box would have made the unit more robust, but since a project is never complete, I decided to leave the board out for easy access when it came time to add or modify the circuit. Also notice the visible LED on the side opposite the sensor unit. Again, it's placed directly in front of

Figure 11-9 *Inserting the cap through the lens hole.*

Figure 11-10 *The completed infrared dream mask.*

my eye for optimal brightness. The indicator LED will be discussed in the last section of this chapter when dealing with the microcontroller and programming. You also might notice that there is no on/off switch on my dream mask. This is so because I usually recharge the batteries after each use and just let the unit run all night, which is usually how I use the system. If your plan is to wake an hour before your normal wakeup time and then use the mask, a power switch would be handy.

Once your dream mask has passed all your tests, it's time to become an oneironaut and make the journey to your other world. As you can see in Figure 11-11, the lucid-dream mask is more than just a powerful tool; it is also a cool Halloween prop or "steam punk" accessory! Besides alignment and initial testing (discussed in the last section of this chapter), the only other concern is sleeping with your mask strapped to your face. I found the ski mask comfortable, and since I normally sleep on my back, wearing it to bed was not a problem. Where I initially had some difficulties was being too excited to fall asleep easily owing to the possibility of an easy lucid-dream journey. It actually took a few nights of going to bed wearing the mask to end up with a good night's sleep because I ended up removing it to actually fall asleep. On the first full night with the mask, I woke right away when the LED started flashing—again too eager to see results. I ended up enhancing the microcontroller code to hold back detection until 6 hours after starting the unit (discussed later) so that I would not be awakened from an early REM stage. After several attempts and adjustments, I found that the mask does indeed fulfill its purpose as long as you can remember to "see the signs" in the dream world. These issues will be discussed at the end of Section Two when you complete the microcontroller part of this project.

Now let's look at an alternative to the infrared light-sensing system that uses an analog accelerometer IC to detect actual eyelid movements.

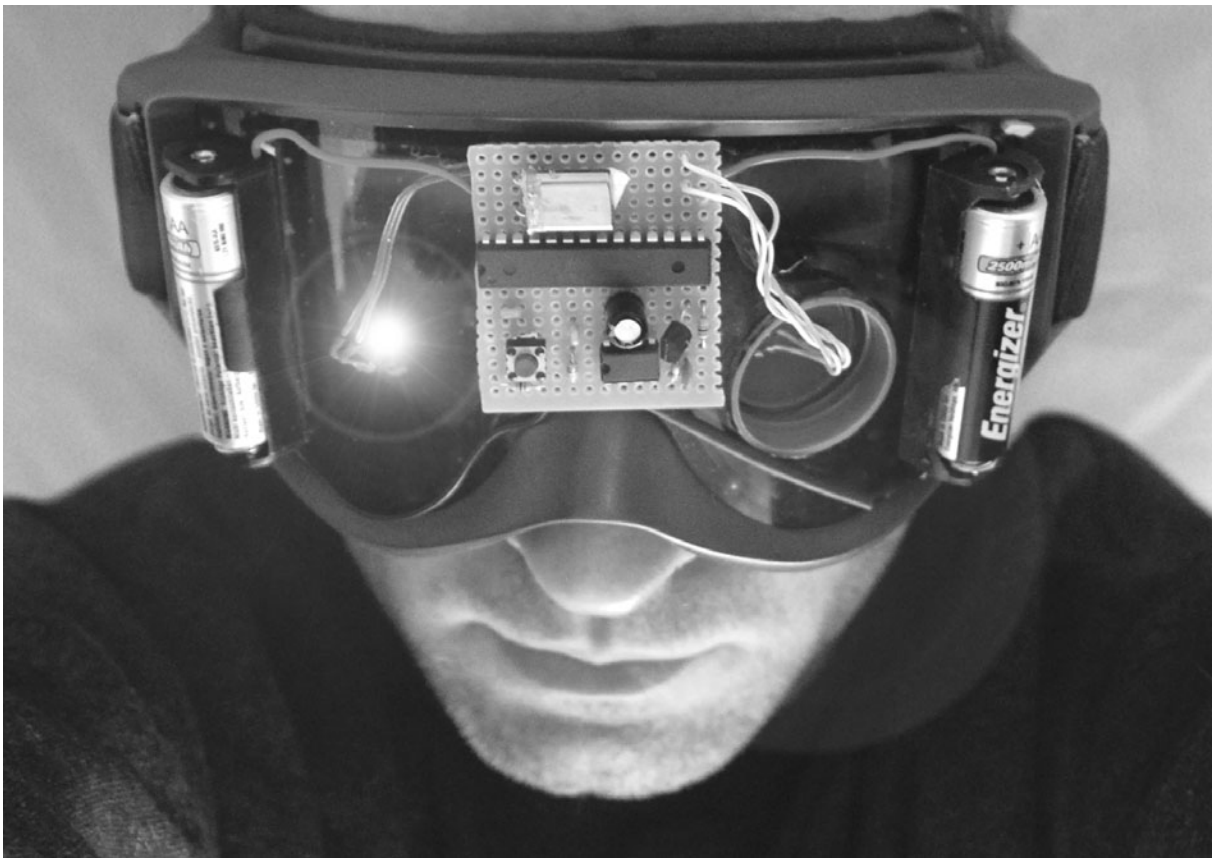


Figure 11-11 *Another oneironaut ready for takeoff!*

Project Twelve

Motion-Sensing Lucid-Dream Mask

This version of the lucid-dream mask project will replace the infrared light sensor with an actual motion sensor that can detect the smallest change in velocity. I came up with this system after finding the infrared version to be a bit quirky when it came to getting proper alignment. Even the commercially available units sometimes can be finicky when it comes to moving around in bed or having ambient light enter the phototransistor, so I wanted to try an alternative method of detecting eyelid movements. By using an inexpensive tin IC called an *accelerometer*, I was able to place the unit directly on my eyelid and feed its output directly into the microprocessor, bypassing all analog circuitry and making the unit completely impervious to ambient light and all electrical noise. It sounds uncomfortable to have something in contact with your eyelid, but the sensor is so small and lightweight that it feels no different from how the cloth on the sleep mask originally felt, so it is not an issue. The good news is that this system is far easier to set up and get working properly because the only data sent to the microcontroller are actual eyelid movements, and changes in the subject's position do not cause the system to fail owing to alignment changes.

An accelerometer is a tiny IC that can detect the slightest change in direction, and accelerometers are used in everything from self-balancing scooters (e.g., Segway) to impact sensors in vehicles to activate airbags. These amazing devices are so sensitive that I was able to connect one to a microcontroller and actually digitize my voice just by talking near the unit as it dangled from a wire on my breadboard! Yes, it actually converted the vibrations from the sound into digital data in real time. So you can imagine that detecting the movement on your eyelid would be a very easy chore for such a sensitive device. These little accelerometers are also very inexpensive now and can be purchased from many electronics distributors.

Figure 12-1 shows the accelerometer (the tiny block on the left) I decided to use for this project, the ADXL 202E, and my analog device. This $10 part actually has two accelerometers to detect *x*- and *y*-axis movement, but I will be using one channel because it was all that was needed. You can even find three-axis accelerometers for the same price, and if you get creative with your microcontroller code, you could filter out side-to-side movements, making your eyelid detector extremely robust to any stray head movements. The strip of even smaller accelerometers on the right in the figure holds three-axis units I plan to experiment with later as I get more complex with my microcontroller coding. I even have found

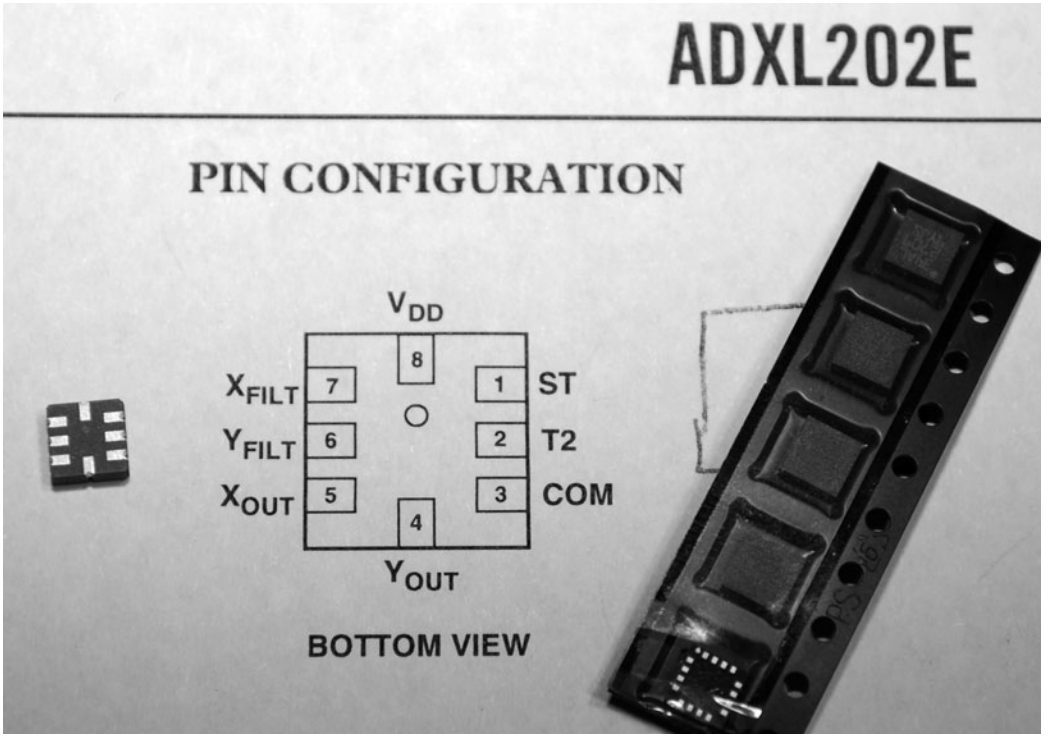


Figure 12-1 *A very tiny and sensitive accelerometer IC.*

these ICs to be sensitive enough to detect my pulse when placed in the optimal spot on my neck!

It really does not matter which brand or style of accelerometer you choose as long as it has a pulse-width-modulation (PWM) output so that it can be measured by your microcontroller as a digital signal. Pulse-width modulation is very simple: A series of pulses is sent out of the device at a constant rate, but the duration (duty cycle) of the high pulse is proportional to the detected motion of the device. For instance, if no motion is detected, the pulse may stay high for 1 ms and then drop back low for another millisecond. This would mean that pulses are sent every 2 ms (while they are using my device). Now, if motion is detected, the pulse may stay high for only half a millisecond and then drop back low after 1.5 ms, indicating that a small change in motion has been detected. Notice that the total pulse time is still exactly 2 ms, but the duty cycle or high time has been changed.

This PWM scheme is very easy to measure in a microcontroller because you simply wait for the rising edge of the pulse and then start a running counter until the output drops back down to zero. It becomes even easier in my system because I don't care about the actual value of the reading, just that it has changed from the last reading, so the code is very simple. Operation of the microcontroller code will be explained in the next section, so let's just figure out how to get the accelerometer set up to detect eyelid movements for now.

Although the ADXL is larger than many of the accelerometers available, it still seems dwarfed by the other semiconductors on my breadboard (Figure 12-2). I did not include a schematic because there really is not much to show; the accelerometer basically feeds its output directly to the microcontroller. The two resistors and the capacitor connected to the accelerometer are used to set the filter and response time to 2 ms as per the datasheet. Your accelerometer likely will have different methods for setup, but not to worry,

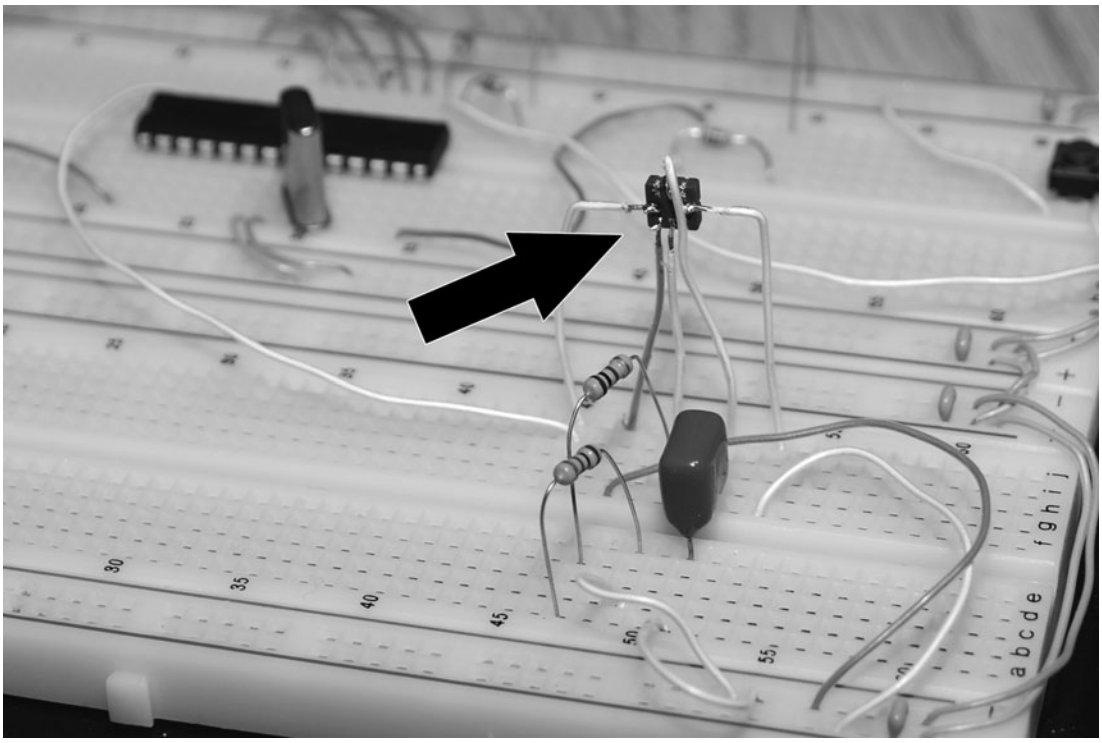


Figure 12-2 *Testing the accelerometer on a breadboard.*

because the microcontroller code is very forgiving; all it does is compare the PWM signal coming out of the accelerometer. Much faster or slower sampling rates will work just fine because it is change, not time, we are comparing.

It is best to start on a breadboard and include the programmed microcontroller so that you can make sure you have soldered the wires correctly on the tiny accelerometer package. If you have yet to work on the microcontroller part of this project, then you can just feed the output into an oscilloscope and watch the waveform change as you tap on the body of the accelerometer. Another trick I use to check if these things are functioning is to connect a piezo buzzer directly to the output pin and listen to the digital nose change as I tap on the device. An ugly hack, I know, but it does work! Once you have determined that there is indeed a signal spewing from the accelerometer, you can proceed to mount it on a small perf board along with the few needed semiconductors that make it function. The goal is to have only three wires coming from the accelerometer unit: power, ground, and output signal.

To keep the accelerometer unit as light as possible, I decided to cut a small bit of perf board to fit into a plastic bottle cap, as shown in Figure 12-3. In this way, I could add a bit of felt to the cap and just rest it on my eyelid for movement detection. This worked very well and did not even require a sleep mask when sleeping on my back because the cap just stayed there. There are many ways you could mount the accelerometer, including just a bit of heat shrink with the thin wires coming from the device. The wires should be thin and flexible as well because you do not want to hinder the movement of the accelerometer unit as your eyes begin to move back and forth. The other bottle cap shown in the figure will be used to carry the visible LED so that my sleep-mask installation has basically the same thing on both sides.

It took a bit of fine work to get the tiny accelerometer soldered down to the small

Figure 12-3 *Cutting a small perf board to fit the bottle cap.*

breadboard, as shown in Figure 12-4, but with a little patience, surface-mounted devices are not as bad to work with as you might think. Even after installing the two timing resistors and filter capacitor, there was plenty of room left over in the bottle cap. Maybe if you are a real wiz with a soldering gun, you could even include a surface-mounted microcontroller and 3-V button battery to make the entire device self-contained in a space no larger than a bottle cap! I opted for the easy way out and stayed with a dual inline package (DIP) microcontroller for now. Notice that there are only three tiny wires coming from the cap: power, ground, and output signal. Remember, you want the accelerometer to be as light as possible so that its range of movement is not restricted in any way and so that it does not need to put any pressure on your eyelid when in use.

The completed circuit, including the microcontroller, accelerometer unit, and visible LED, can be built as shown in Figure 12-5, ready for a new home in a sleep mask or to be used as is. If you sleep on your back and do not have a habit of tossing around all night, then you may not even need to install the components on a mask because the small accelerometer and LED units can just be placed directly over your eyelids if you place the circuit board remotely in a small

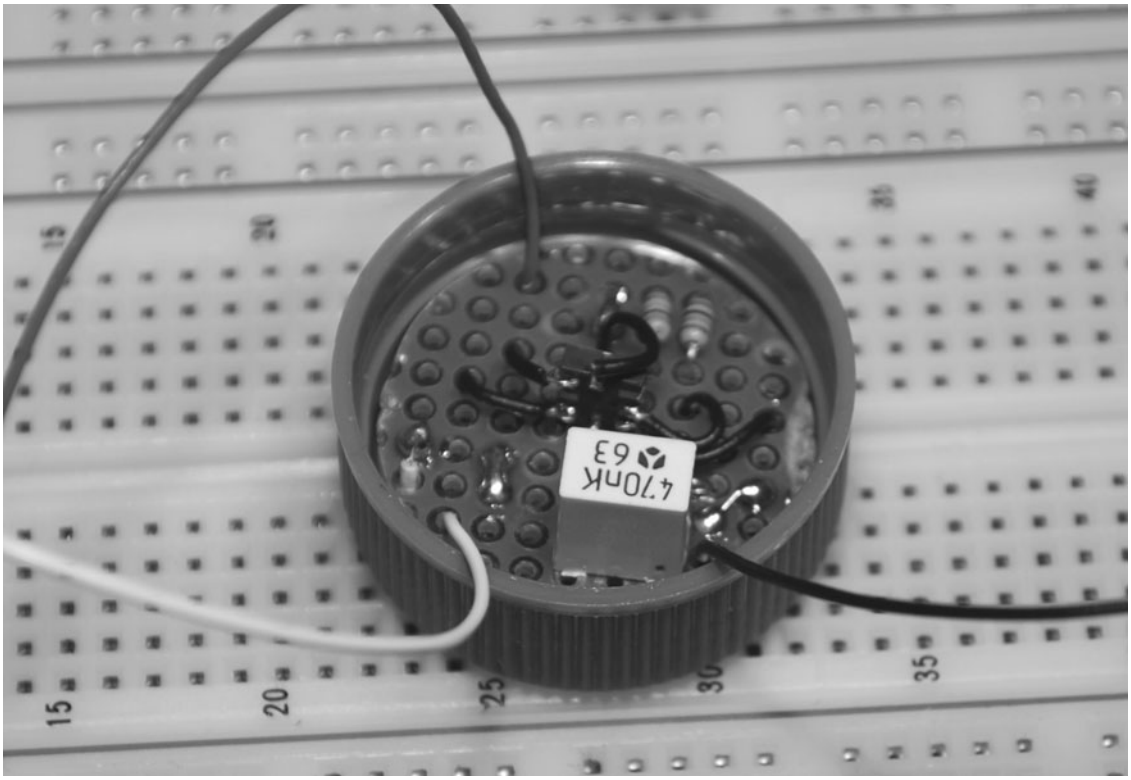


Figure 12-4 *Accelerometer and components in a bottle cap.*

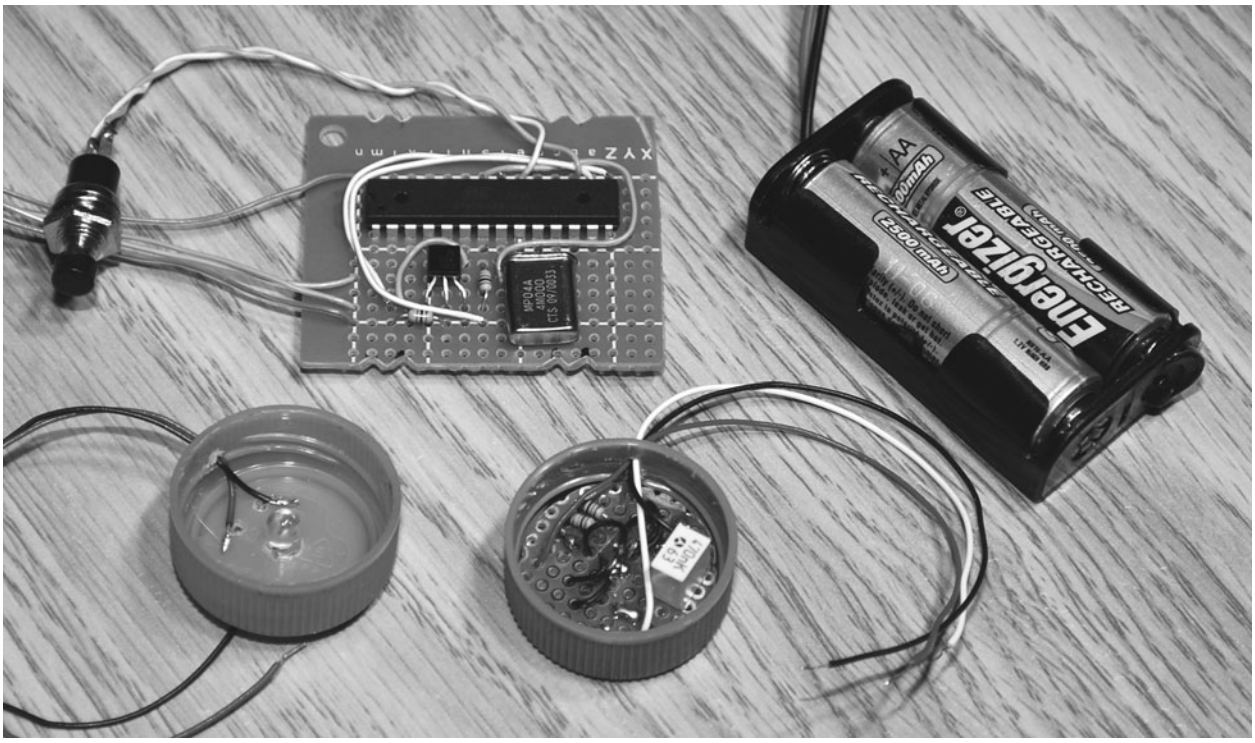


Figure 12-5 *Accelerometer, microcontroller, and visible LED.*

box. Since this version of the REM-detection system is completely oblivious to ambient light, you do not have to shield the sensor or worry about nearby electrical devices that might induce noise into the circuit. I actually prefer using the device remotely, with the two caps resting on my eyelids, but I will show how a cloth sleeping mask also can be used for those who may sleep sideways or move around a bit at night.

Even if you decide to use the device without a mask or a face strap, you will need to add the microcontroller, batteries, and test switch in some type of case, as shown in Figure 12-6. Besides the microcontroller and LED-driver transistor, there is not much else to the control board, so it should fit in a small box along with the two AA or AAA batteries, an on/off switch, and the test/reset pushbutton. Since operation of the unit only requires turning it on and then pressing the test button to ensure that the sensor is working, you will never have to look at the box, so it can be placed remotely so that it is easily accessible while you are relaxed and "wearing" the accelerometer and LED units.

If your battery pack and black box are small enough, you can mount the system directly to a cloth face mask to create a self-contained unit, as shown in Figure 12-7. The cap with the accelerometer is on the right side of the figure, and the visible LED is on the left. Notice that I am using the accelerometer cap top down and the LED cap open-end down. This makes the visible LED brighter and allows the accelerometer to move easily as my eyes are moving. I later added some felt pads to both caps, which helped to keep them in place when I was sleeping on my side using the mask setup. As with all home-built projects, a little trial and error (hacking) probably will be needed on your part to find out what works best for you.

The control box is shown on the front of the cloth sleeping mask in Figure 12-8, ready to be used for some serious lucid-dream training missions. I found this version of the REM-

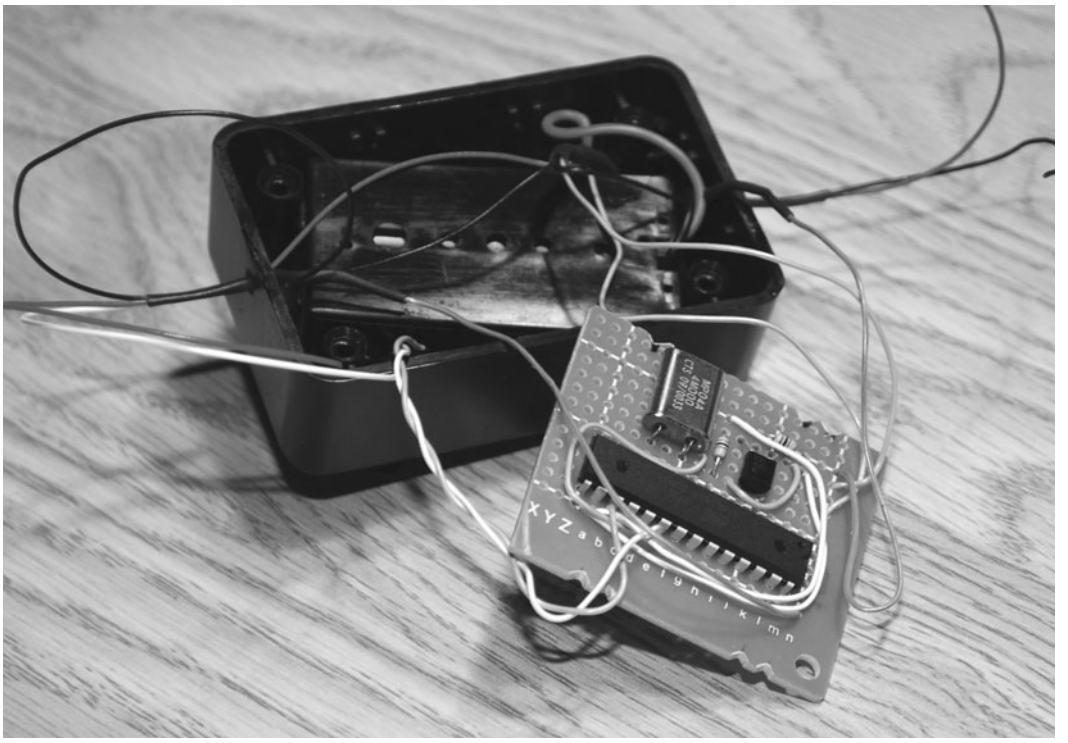


Figure 12-6 *Installing the microprocessor unit in a box.*

Figure 12-7 *Installing the hardware onto the sleep mask.*

Figure 12-8 *The controller can also be installed on the mask.*

detection system to be much more robust and easier to set up than the infrared version presented earlier, and it has been a great tool for further exploring the dreaming world. No matter which unit you decide to build (how about merging both of them together), you will need to move on to the next step and program the small microcontroller to sort out the raw data from the sensor and control the visible flashing LED. If you have never considered adding microcontrollers to your arsenal of electronics know-how, then trust me, it is worth it and will open up doors you never thought possible. Next we will build the lucid-dream mask controller.

This page intentionally left blank

Project Thirteen

Lucid-Dream-Mask Controller

Now that you have a working REM-detection system based on either the photo sensor or the accelerometer, you will need a way to create a response from the data received. Using some fancy analog circuitry and a few digital counters, you probably could rig up a system that counted pulses from the detected eyelid movements and then triggered a flashing LED, but this would become a very quirky system with a large number of components. The lucid-dream-mask controller needs to perform a number of different operations to make your lucid-dream mask functional and easy to use.

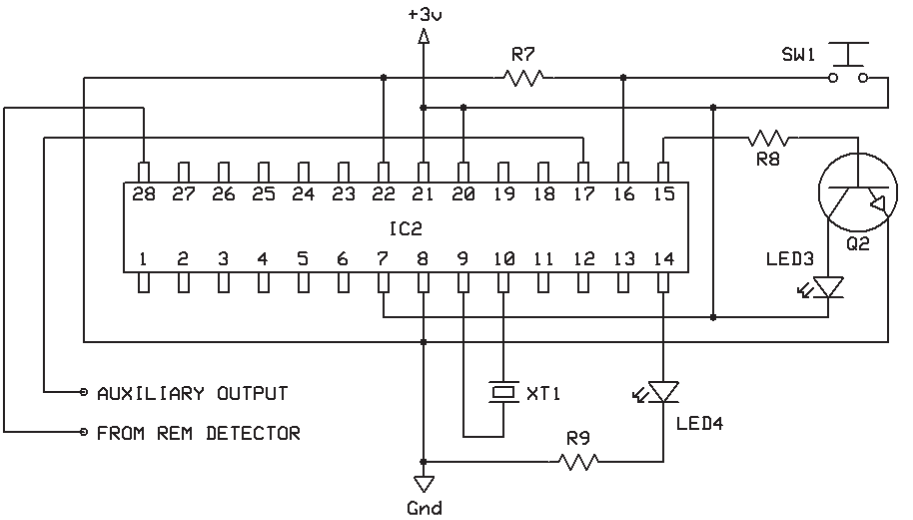
First, the controller needs to either detect the difference in the analog signal from the phototransistor, or it needs to measure the change in PWM from the accelerometer. This job would require a lot of finicky analog circuitry to get working properly, but any small 8-bit microcontroller can do this job and not even break a sweat. After the detection phase, the controller must begin to count changes in order to avoid triggering the visible LED each time the user moves his or her eyes. A few movements may not indicate REM sleep, so the controller needs to look for continuous eye movements, which is another simple task for any microcontroller. In my version, I just count 20 eye movements. Once the controller has decided that the user is in REM sleep, the visible LED must be flashed a number of times (100 in my case) so that the user can look for this signal in his or her dreams. Often the signal will be something related to flashing light in the dream, so it takes a bit of practice and conditioning to recognize the signal.

Another task for the controller is to create some type of test procedure so that the user can verify operation of the unit and to reset the counter if a false trigger happens owing to accidental waking eye movements or simply because of waking up from REM sleep. In the version of the Basic code presented here, the controller does both functions from a single pushbutton so that it is very easy to use the system in the dark. If a false detection happens, the user simply presses the button, and the controller resets the system to start over. If the user holds the button down for 5 seconds, the controller puts the system into test mode and will flash the visible LED instantly every time there has been sensor detection. In this way, the user can verify that the system is indeed responding to eye movements. To exit test mode, the user holds the button down for another 5 seconds, and the system once again resets and starts working in REM-detection mode.

The source code is extremely simple and was written in Basic so that it can be ported easily to any other microcontroller or any other language. Before we dig into the source code, let's look at how simple the schematic for the lucid-dream-mask controller really is.

Although the schematic shown in Figure 13-1 is amazingly simple, the unit has a great deal of functionality, even with the extremely simple basic program that lives inside the chip. The Atmega88 from Atmel is an inexpensive 8-bit microcontroller that offers 23 IO pins, 8K flushable program memory, 1K SRAM, 20-MHz operation, a bunch of onboard peripherals such as an analog-to-digital converter and serial port, and a cluster of hardware timers. Of course, this is pretty standard for many of the low-end, low-cost 8-bit microcontrollers offered by various other companies such as Microchip as well. Because the controller program is kept to a minimum and written in Basic, you could adapt easily it to just about any low-end 8-bit microcontroller or even a Basic Stamp module from Parallax, Inc.

If you have never heard of an 8-bit microcontroller before reading this book, then think of it as a blank IC that costs you only five bucks and can be programmed to do anything imaginable from emulating a simple logic circuit to playing video games on your TV. No kidding! I have made single microcontrollers play music, record sound, control huge robots, log computer keystrokes, and even emulate entire retro 8-bit computer systems. Since the magic happens in your code, it will feel like the entire world is under your control once you feel the magic of the microcontroller. There are thousands of various microcontrollers available, some of them having as few as 8 pins and a few kilobytes of program storage, whereas others have hundreds of pins and megabytes of onboard memory. Our little Atmega88 is considered a smaller microcontroller, but it is still overkill for the simple program presented here.



PARTS LIST FOR THE LUCID DREAM INDUCER
IC2 = ATMEGA88 MICROPROCESSOR
R7,R8,R9 = 1K
XT1 = 4 MHZ CRYSTAL
SW1 = NORMALLY OPEN PUSHBUTTON SWITCH
LED3 = HIGH BRIGHTNESS COLORED LED
LED4 = LOW BRIGHTNESS RED LED

Figure 13-1 *Lucid-dream-mask controller schematic.*

Each brand of microcontroller has its own unique instruction set of assembly language, and as you progress into the hobby, you will usually end up programming in assembly so that you can create the smallest, fastest, and most efficient code possible. Of course, high-level languages such as C and Basic are much easier to start with and are good for rapid prototyping because you usually can get your program working in a few hours unless it is a real monster. For this reason, I chose Basic because it is so easy to understand that you probably could make modifications even if you have never written a program in your life. Basic is also available for most of the major brands of microcontrollers, and the language is very similar among the different varieties.

Now back to the schematic. The Atmega88 runs fine on just 3 V and only needs an external crystal to start working once your program is compiled and loaded into the internal Flash memory. XT1 is a 4-MHz quartz-crystal resonator and was chosen only because it is a common value and I had many of them in my junk box. The Atmega88 actually offers an internal 8-MHz clock source, so the crystal is not even necessary, but I have a lot of source code designed to work at 4 MHz, and the internal clock is not always the most accurate clock source.

Pin 15 is configured as an output and will drive the transistor that switches on the visible LED (LED3) that will try to signal the dreamer to become lucid. You also can try using the resistor directly from the microcontroller's output pin through a 150-W resistor because it may not need to be all that bright since your eyes are already adjusted to the darkness. I found that the system worked best if the LED was so bright that it was almost too bright when trying out test mode. The test LED (LED4) and resistor R9 are used only while breadboarding the system, so they can be omitted from the final design. LED4 will flash every time a change in input data has been detected. Also, test mode does the same thing. Pin 16 is set to be an input and tied low so that the pushbutton switch (SW1) can be detected as the pin goes high. This button controls reset as well as test mode by holding it down for 5 seconds. Pin 17 has been set up for use as an auxiliary trigger in case you want to activate some other external device during REM detection. The audio dream director presented earlier in this section would work well here. And of course, pin 18 is the input pin that will read the PWM or voltage changes from the REM-detection system. When using the infrared phototransistor, the input is fed to the onboard analog-to-digital converter, and when using the accelerometer, the pin is just a digital input pin used to measure the pulse-width time. This is why two versions of the code are presented.

Programming a microcontroller is as easy as pressing the "program" button—well, as long as your program works, because we all know the rule "garbage in, garbage out" when it comes to programming. I work with both PicMicro and Atmel microcontrollers and consider both to be equal when taking into account such things as price, support, speed, and ease of use. Of course, PIC versus AVR is like Ford versus Chevy, so I don't plan to go there! Both platforms have good C and Basic compilers available, and the assembly-language instruction set is very easy to understand. In this project I chose the Atmega88 from Atmel because it was inexpensive, fast, had more than enough IO pins, and my programmer, the STK500 (right side of Figure 13-2), allows for programming directly in circuit or on the breadboard. My PIC programmer requires me to place the chip in its onboard socket, so this is less convenient when trying out a lot of small code changes.

No matter which microcontroller brand you wave the flag for, you will need a programmer that can support the device you plan to use. Most commercial programmers will support many or all of the devices in a certain microcontroller family, and this is true with both the STK500 and the PicStart Plus programmers that I often use. There are also many low-end or home-brew programmers available, but be aware that support

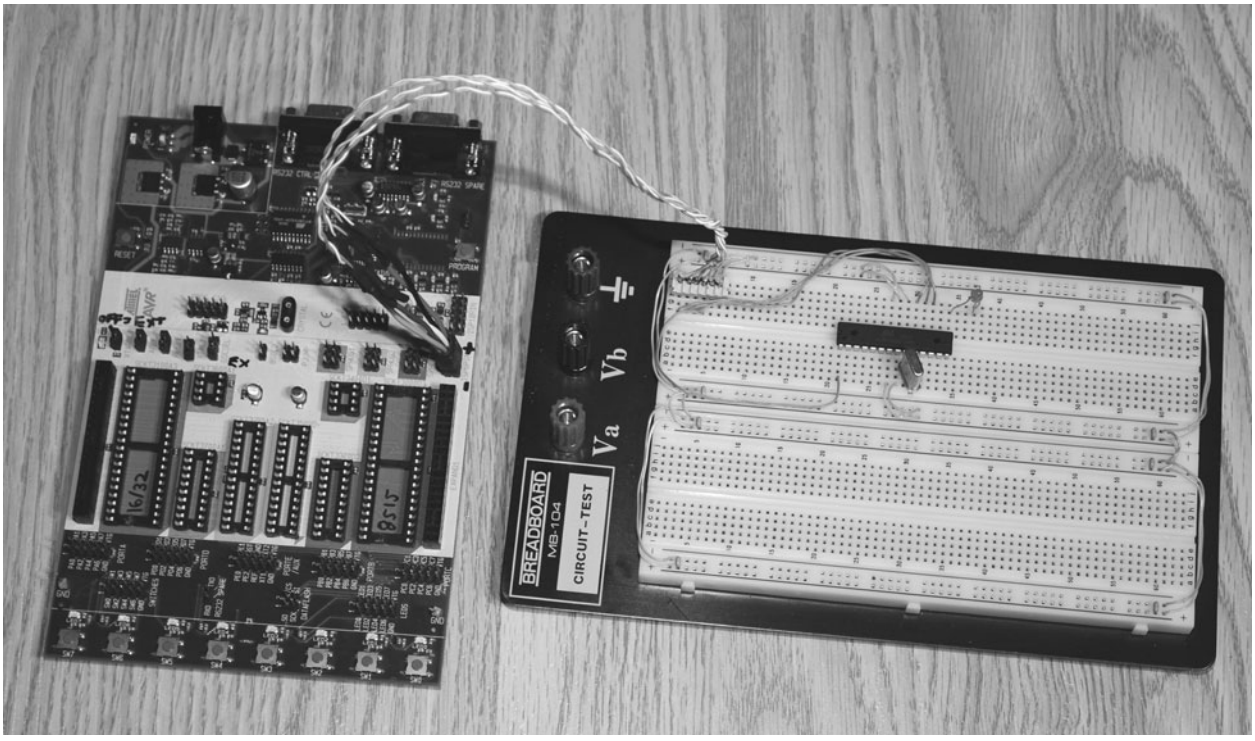


Figure 13-2 *Programming the AVR in circuit.*

may not be as good, and you may find that they may support only a limited number of microcontrollers. The STK500 shown in Figure 13-2 is probably my favorite programming board because it lets me program almost all the 8-bit AVR family, and I can do it right on the breadboard or directly in circuit. Being able to quickly alter lines of code and see the results right away, in my opinion, is a far more useful debugging process than using a software-based debugger.

Figure 13-3 shows the Atmega88 in my breadboard ready to be programmed. The four wires coming in from the left go back to the STK500 programming board, which is connected to the serial port on my computer. To program the chip, I just press the "compile and program" button on the Basic compiler, and away it goes into the Flash memory of the microcontroller. Once in the microcontroller's memory, a program is there forever, although you can make as many changes as you like. To program a PicMicro device, I have to remove the chip from my breadboard and place it in the programmer's onboard socket, a process that is fine as long as you don't plan on making a lot of changes to your code. Of course, there are in-circuit programmers available for PicMicro devices as well, but I have not used them myself.

Although I work mainly with assembly these days, I must admit that you can get a lot done in a very short time using Basic, and Bascom AVR is a really nicely done compiler with a rich set of commands and support for most 8-bit AVR devices. Figure 13-4 shows the Bascom AVR set to compile the lucid-dream-mask controller program in the Atmega88 device. Of course, when optimal speed and code size matter, assembly will kick Basic and C right out the door. For rapid prototyping and experimentation, it's hard to beat the ease and speed at which a program can be created using a high-level language. Since our dream-mask program is only reading a simple input and then running a counter, speed and code size are of no concern, so Basic is a good choice.

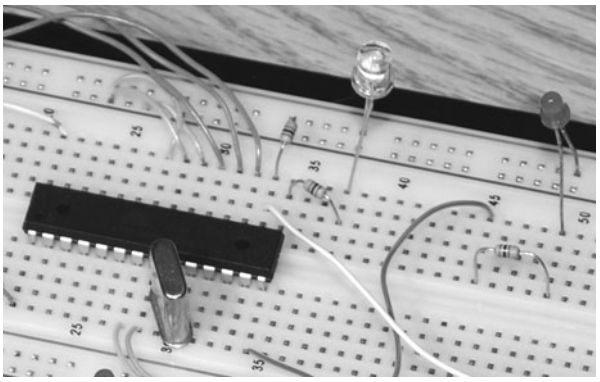


Figure 13-3 *Connecting the in-circuit programming lines.*

Now let's look through both versions of the program code so that you can see what happens inside the microcontroller. We will start with the infrared phototransistor–based version first because it was presented first in this section. Also, since much of the code is the same in both versions, this one will be explained in more detail.

Have a read through the complete Basic source code of Listing 13-1 provided in the appendix so that you can get an idea of what the code is doing. If you are an experienced programmer, then this trivial code is probably something you could write in 15 minutes from scratch, but if you have never written a program in your life, not to

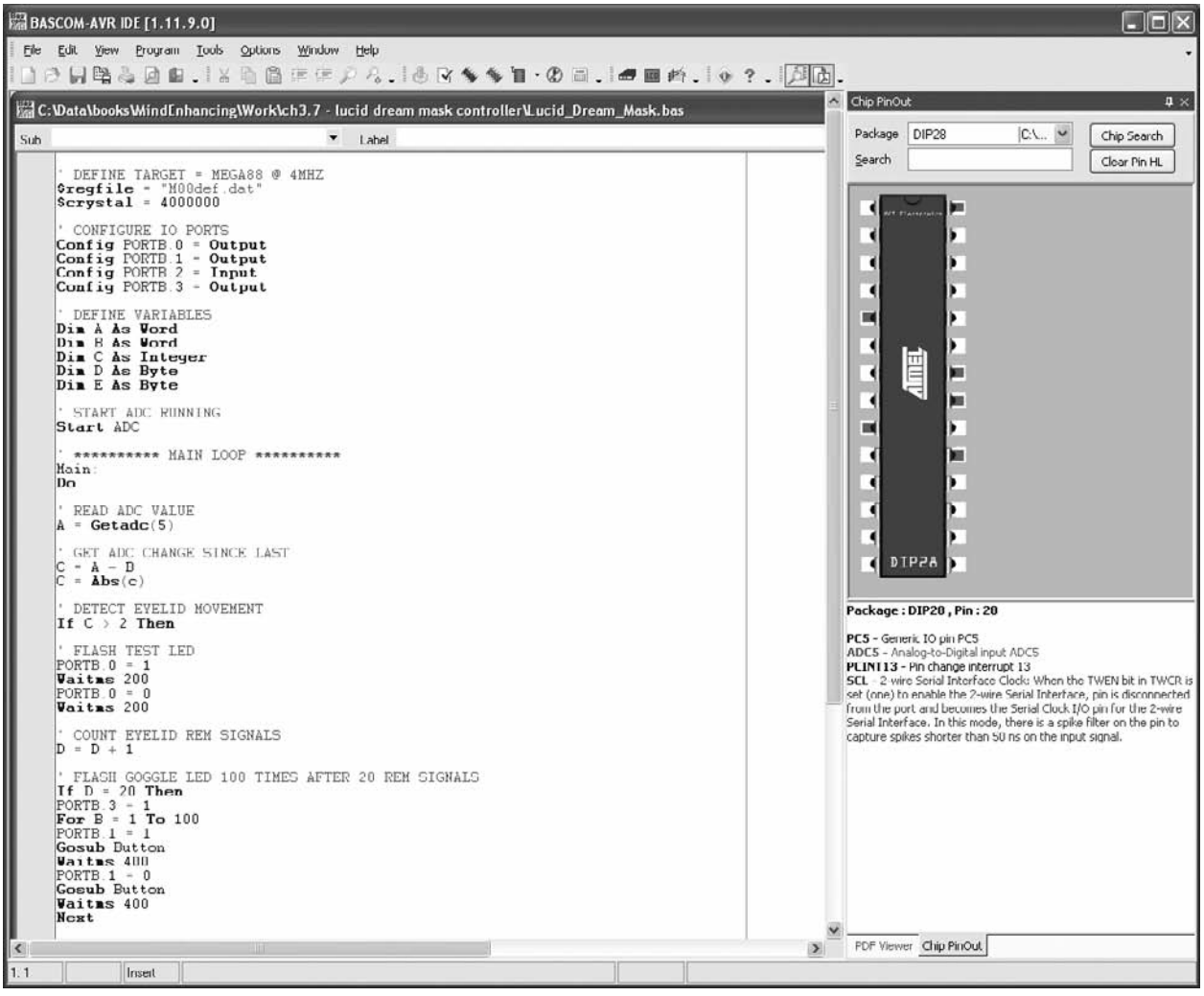


Figure 13-4 *Using Bascom AVR to compile my program code.*

worry because Basic is called that for a reason and is easy to understand. The lines in the program listing that start with an apostrophe are comments, and I will explain the code in blocks after each comment.

The code following this comment is required to tell Bascom that we are going to target the Atmega88 device and that our clock will be an external crystal resonator running at 4 MHz. Telling the compiler your clock speed becomes important when using commands that deal with timing-sensitive routines such as serial transmission or analog-to-digital readings. Defining the device also helps the compiler to generate user errors that have to do with IO pins. In this way, you can't accidentally try to toggle an IO pin that does not exist on the actual device.

Basic uses *variables*, which are letters or words used to hold values. I like to use single letters such as *A*, *B*, and *C* for simple programs like this one, but when you are working on a large, complex program, use of more descriptive variable names is recommended. TIMER2 and REDLED1, for example, are descriptive variable names that make a lot more sense in a huge block of code. Variables also are defined as the number of bits they are to contain, so in our code, "A" and "B" are 16-bit "words," which can contain a value of between 0 and 65535. Variable "C" is an integer that can range in value from –32768 to +32767. Variables "D" and "E" will only contain values from 0 to 255, so they are bytes. Although you could just define all variables using larger data types, this is a waste of memory space and will slow down your code.

The "Start Adc" command tells the compiler to include the code necessary to set up and initiate the onboard analog-to-digital converter on the Atmega88. This will allow us to read in an analog voltage and convert it into a value in order to test the state of the phototransistor on the dream mask.

Everything from here on is going to happen continually until the word "Loop" is reached, which causes program execution to start again where it first encountered the word "Do." This is called an *endless loop* because it never stops unless forced to by another command or an error.

This command reads the analog-to-digital converter (ADC) on pin 5 of the Atmega88 into variable "A," which is where the output from the phototransistor is connected. Since the ADC

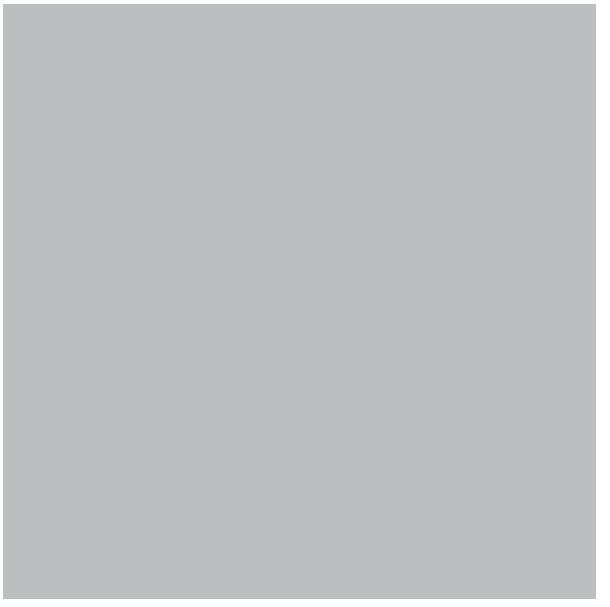
returns a 10-bit reading, values can range from 0 to 1024, which is why variable “A” needed to be a “Word,” not a byte.

These next two lines compare the current ADC reading “A” with the last known reading “B” by subtracting them. The “Abs(C)” command changes the value in “C” to an absolute nonnegative value, so we get the difference only as a whole number, not a negative number, if “B” is greater than “A.”

This line checks to see if the value in variable “C” is greater than 2. If it is, then all code following the “Then” statement will be executed. If it is not, then the program will skip ahead until if find a line the reads “End If.” In this way, the counter processing code to follow will not run unless the reading on the ADC has changed by a value of at least three out of a possible 1024 values. Thus, if you want to reduce the sensitivity of your dream mask, just increase this number.

If there is an LED connected to pin 14 (Portb.0) of the Atmega88, it will blink for 200 ms once a change of greater than 2 has been detected from the ADC. The “Waitms” command simply delays code execution for any number of milliseconds, so the LED goes on for 200 ms and then goes off and waits for another 200 ms. Writing a 1 to a port sends VCC to the port, whereas writing a 0 to the port clears it to ground.

“D” is a variable that will hold a counter that counts the number of times the ADC has detected a change. Since we don’t want the dream mask signaling us every time we twitch an eye, we use a counter to keep track of eyelid detections so that we can wait until they are happening nonstop.



This block of code will execute only if the counter variable “D” reaches 20 so that we can basically filter out any spurious eye movements.

Once “D” reaches 20 counts, pin 15 (Portb.3) is turned on and off 100 times at a rate of 800 ms. Pin 15 is connected to the base of the LED diver transistor, so this will switch it on or off. During each LED flash, the “Gosub Button” command jumps to the button-handling routine just to make sure that we are not trying to reset the device in case this was a false detection owing to eye twitching or an accidental wake-up from REM sleep. If you find that there are too many false detections, then just increase the number 20 to something higher. Also notice that “Portb.3=1” sets pin 17 to VCC, which is an auxiliary output in case you want to connect some other device to the unit.

After flashing the indicator LED in the dream mask 100 times, the program simply resets the variables, turns off the auxiliary pin, and then starts over again from the beginning. If you have another REM episode, the program will again detect it and flash the indicator LED in your dream mask. The two “End If” lines close off the previous two “Then” statements.

This line simply makes the variable “B” take on the last value of the ADC so that when a new value is read for “A,” it then can be compared with the old value to see if it is different.

Again, the program calls the button-handling routine in case the user is pressing the reset/test button. “Loop” then sends the program back to the “Do” statement so that the entire loop can continuously run.

This is the start of the button-press routine, and the first line checks to see if the button is actually pressed by reading pin 16 (Pinb.2); otherwise, it just returns back to the calling line.

If there was a button press, some of the variables are cleared, and “B” is set to 100, which will cause the LED flashing counter to end if it was running, once the button-handling routine is over.

Because things happen in megahertz in a microcontroller, we need to use counters and delays to compute seconds. Variable "E" increments as the user holds down the pushbutton, and then the code delays for 20 ms. If the user holds the button down long enough for "E" to count up to 200, then the program assumes test mode because about 5 seconds have elapsed. The routine then jumps to the "Setup" label in the code to run setup mode. If the button was not held for at least 5 seconds, then the routine just ends and returns back to where it was called.

This is the beginning of setup/test mode, and it is another endless loop.

Again, the program reads and compares the current ADC value "A" against the last known value "B" to see if there was a change of greater than 2 of 1024 values.

If an ADC comparison exceeds a value of 2, then this code block flashes the visible LED so that the user can see the results directly as he or she keeps his or her eyes closed and moves them back and forth. In this way, you can tell if the system is functioning properly or not in real time.

This block of code is much like the block of code that was used to enter setup mode. This time, if the user holds down the pushbutton for 5 seconds, the program exits setup mode and returns to the main loop to go back to normal operation.

Now you can see why Basic is great for rapid prototyping; the language is almost pure English and is so easy to work with. The complete accelerometer version of this code, shown in

Listing 13-2 and provided in the appendix, is very similar, and since there are only a few changes, I will just post the differences and explain them.

Since there are no analog voltages to deal with, the ADC is not used this time, so the obvious difference will be the exclusion of "Start Adc" in the code.

Instead of reading the ADC value, we read pin 5 as a digital input, but measure how long it takes for it to go from high (VCC) to low (GND) and then back to high again. The "Pulsein" command does this for us automatically. This length is stored in the variable "A" just like the ADC reading was in the preceding version of the code. Because of this, most of the code can be reused because we really don't care what the value is but only that it has changed since the last reading.

Of course, in setup mode, we also have to remove the ADC reading and replace it with the "Pulsesin" command, but other than that, the two versions of the code are exactly the same.

There is a lot of room for further improvements to this code and plenty of program space as well as IO pins left on that microcontroller, so get creative and add your own neat features and options. A much better REM-detection routine would include a timeout counter so that rather than just counting eyelid movements, the counter resets after a few seconds if eyelid movements stop. This would greatly reduce false detections from erroneous movements or eye twitches. You also could add a sound-alert system right into the microprocessor using one or more of the output pins to make beeps or complex noises or even play sound digitized from an external EEPROM memory. The possibilities for modifications and improvements are endless, but I only had so much space to add code to this book, so the "bells and whistles" are up to you.

There is also an option to use a microcontroller for this project that will allow you to collect the data into your PC for processing and analysis. There are many external ADC devices such as the one shown in Figure 13-5 that will read in an analog value and then transmit it to your PC through the serial or USB port.

The unit shown in Figure 13-5 is the Pico ADC from Picotech, and it is a very accurate and inexpensive ADC that can be used with the included data logger or in your own programs. Drivers for this ADC are even included for Visual Basic, so you can write a program almost exactly the same as the Atmega88 program to test your dream mask. Of course, with a computer, you have to run wires from the mask, but you open up a whole new bag of possibilities by using your PC to trigger your dreaming mind. You could add sound files or connect LEDs, buzzers, or even tactile feedback hardware to your body for some very interesting experiments. Of course, you will now be plugged into the ac system, so again, the risk of a lightning hit or power surge is always

Figure 13-5 *An external ADC for your PC.*

there. There are many rf modules available that can transmit and receive data for short distances, so you might want to consider that option if you decide to use your PC in your "sleep laboratory" at night.

Now you have a powerful tool to aid you in your lucid-dreaming journeys, so have fun experimenting with it, and remember that the dream mask alone may not be enough to induce a lucid dream. You need to practice proper dream-recall techniques and train yourself to do a reality test anytime you see a bright light or flashing light source in the real world. These could be car signal lights, a video game screen, room lights going on and off, and the like. Once you begin to automatically ask yourself, "Am I dreaming?" each time you see a bright or flashing light, you eventually will begin to do the same in REM sleep as the mask flashes its visible LED for you. The instant you realize that you are indeed dreaming, you can become an oneironaut and take control of your dream world!

Remember the waking reality tester presented earlier in this chapter? Well, why not remove the vibrating motor and modify it to flash a bright LED instead? The unit then could be placed near your workstation and help you get into the routine of seeing the flashing light signs, as well as remind you to question your reality at the same time.

There are many more techniques you can use to help your dreaming mind into a lucid state, so have fun, be patient, and dig around on the Internet for more information if you enjoy this subject; there is a lot of great information out there. Maybe I will see you in a dream someday. Oh wait, that's an entirely different "fringe science" for another time! In Section Three we will go on an inward journey using meditation, clairvoyance, hypnosis, and color-therapy devices.

This page intentionally left blank

Section Three

An Inward Journey

This section explores the amazing biological computer that we all carry around with us, our brain! Unlike the art of hacking a computer, you can't simply jack a few wires onto the CPU or alter the program code that makes us who we are, well at least not in any safe manner, so we will explore a few interesting methods that will allow us to "talk" to parts of the brain associated with varying levels of consciousness. Plugging a jack into the back of your spine in order to enter a large virtual world like Neo did in the Matrix is something that will certainly be a reality some day, but for now we must work with the simple, nonevasive tools we have available, and become pioneers in this interesting field. Who knows, maybe one day you will be the one to develop the technology that will allow us to project our consciousness into a virtual world! Of course, we have to start with the basics, so get out your soldering iron, and let's have some fun!

This page intentionally left blank

Project Fourteen

The Ganzfeld Effect

In the 1930s, Wolfgang Metzger, a German psychologist, found that if the eyes are deprived of any focal point or depth of field, the brain basically "disconnects" from the visual system, causing a drastic change in our mental state. The word *Ganzfeld* is German for "complete field" and has been the subject of much investigation into meditation, hypnosis, and even parapsychology, which includes telepathy. Some reports suggest that the use of a Ganzfeld effect device can bring about the same mental state as achieved by those who have practiced meditation for years.

The Ganzfeld effect is also the same effect reported by those who have experienced snow blindness, which often leads to strange hallucinations and a general altered state of consciousness. Some of the most profound research into the Ganzfeld effect was in the area of telepathy, where it has been shown that using a Ganzfeld device can greatly enhance the probability of a successful telepathic transmission. In these experiments, a "receiver" is often in a secluded room using a Ganzfeld device (such as the one that will be presented here) and is allowed to relax for some time so that sensory deprivation occurs. The "sender" then mentally focuses on some random image and attempts to send it telepathically to the receiver. Typically, the average "hit rate" of such an experiment would be around 25 percent, but by using the Ganzfeld device, this rate was shown to be over 35 percent much of the time, even by some researches who do not consider telepathy to be possible.

Since the Ganzfeld device is such a simple contraption to build, why not conduct your own telepathy experiments, or just use the device to send yourself into an altered state of consciousness somewhere between waking reality and the dream world? All you will need for parts are a few Ping-Pong balls and a handful of red light-emitting diodes (LEDs) to build this simple project.

The main component you will require is not a sophisticated IC or a hard-to-find semiconductor, but just a pair of ordinary white Ping-Pong balls. Most of the research into the Ganzfeld effect was done using Ping-Pong ball halves placed over the subject's eyes and a red-light source to create even lighting. The goal is to completely remove all points of reference from the subject's field of view so that the eyes cannot focus on anything, not even a change in brightness or color, which is why a light is usually used. In most of the experiments, red light is used, but since the Ping-Pong balls are flat white on the inside, you can illuminate them with any color light, or just find

Figure 14-1 *Ping-Pong balls with a logo on only one side.*

a comfortable chair outdoors and use sunlight for illumination.

As for the Ping-Pong balls, there is nothing special about them, but you will need to find a set with a logo on only one side, as shown in Figure 14-1, so that you can cut them in half and use the featureless halves.

There will be an equator along the circumference of the Ping-Pong ball dividing the logo side from the blank side. Cut a slit on the logo side as shown in Figure 14-2 using a steak knife or razor knife just large enough to insert the tip of a small pair of scissors. The smaller the scissors, the better because the plastic is very thin

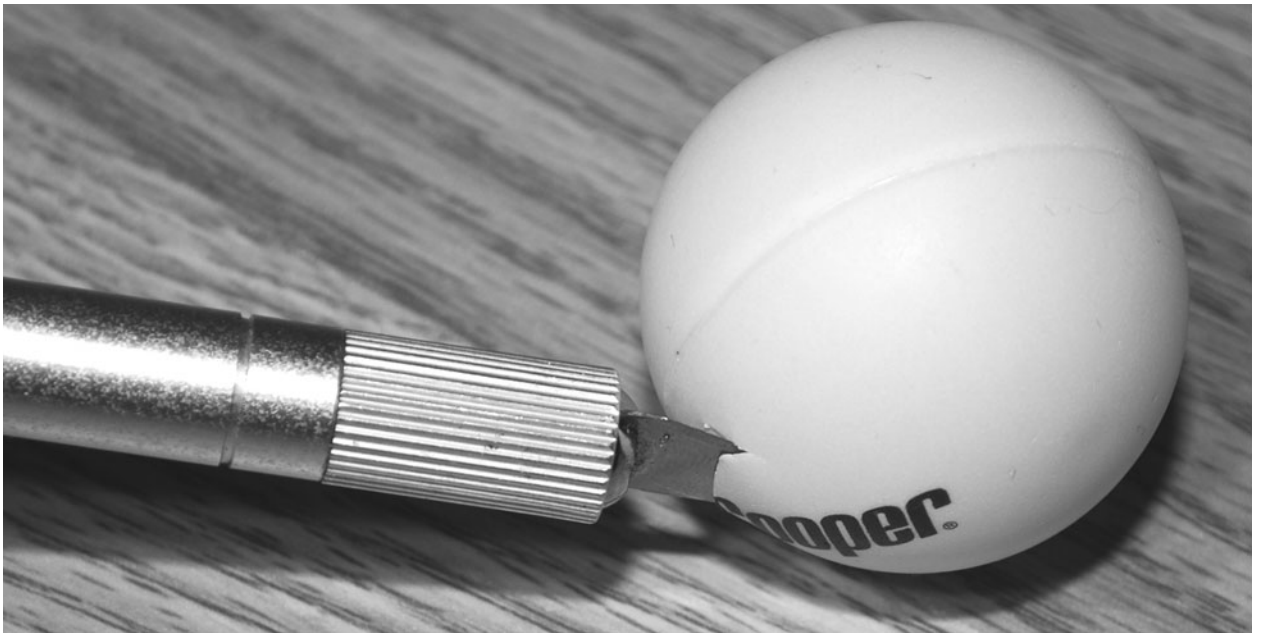


Figure 14-2 *Cut a slit on the logo side of the equator.*

and brittle, so you will want to make small cuts along the edge of the seam to cut the ball evenly.

Try to make small cuts using only the scissor's tip as you work your way around the equator, as shown in Figure 14-3. This will help to avoid the sharp sawblade edges that happen if the cut becomes misaligned. You also could cut around the ball with a Dremel tool if you have the tiny grinder disk attachment.

Try placing the ball halves over your eyes to see how they fit. I found there to be a tiny gap near the sides of my head where the edges did not conform to the shape of my face, and this would greatly reduce the effectiveness of the sensory-deprivation experiments. The goal is to see nothing that you can focus on, so you may have to cut a small slice out of each side of the ball halves as shown in Figure 14-4 to help keep out any stray light or viewable edges.

I cut a bit out of the nonusable ball half just to see what shape would work best for conforming to my face and then used a marker to trace it onto the good halves so that they could be cut. As shown in Figure 14-4, this required taking an arc about half an inch into each edge so that the ball halves made a better seal with my face. You may find that this is not necessary, but you shouldn't have to press down on the balls to create a good fit because this would hamper your ability to relax during the experimentation.

The schematic shown in Figure 14-5 is a guide that you can use to figure out how to run multiple LEDs from a single 9-V battery without the need of a regulator. All you have to do is divide your source voltage (9 V in this case) by the rated voltage of your LEDs (2.3 V in my case). Thus, since 9 divided by 2.3 is 3.9, I can safely run four LEDs in series, which will give each one 2.25 V. It's always better to be slightly lower than the rated maximum than slightly higher if you want your LEDs to enjoy a long, healthy life. Notice that for each series arm, I have four LEDs also wired in parallel. You can have as many LEDs as you want in parallel as long as your power source can handle the current draw. To sum up, then, series wiring divides the voltage by the number of LEDs in series, and parallel wiring increases current draw by the number of LEDs in parallel. In my configuration, I used four LEDs in series

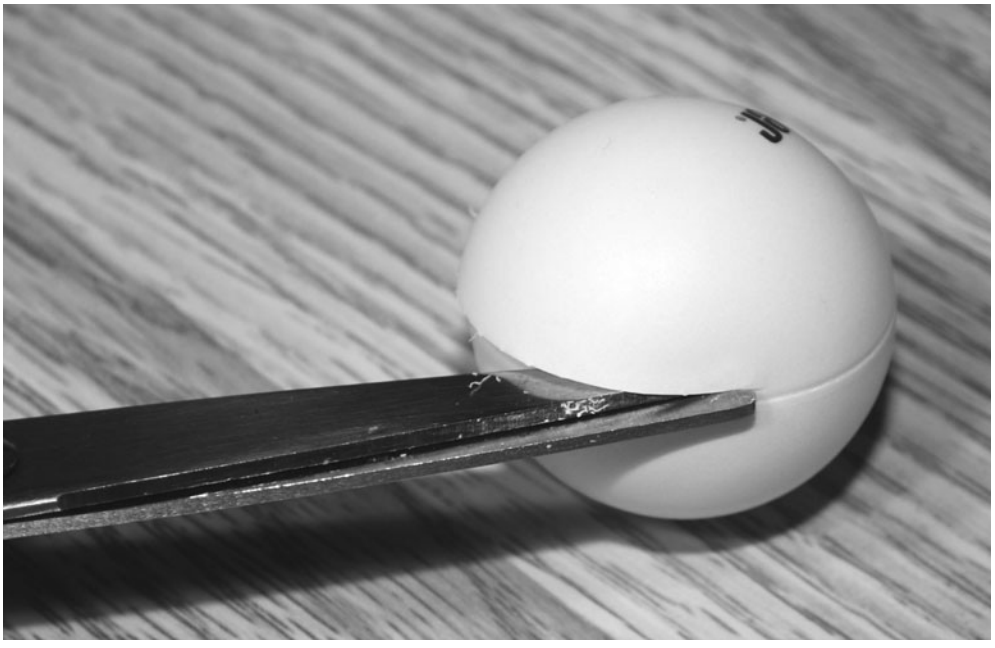


Figure 14-3 *Cutting the ball along the seam.*

Figure 14-4 *Shaping the ball halves to conform to your face.*

with four parallel legs so that there were eight LEDs for each eye—16 in total.

The series and parallel LED configurations are shown on the breadboard in Figure 14-6 as I tested the brightness using a 9-V battery. Remember, when adding LEDs in series, it's best to end up under the maximum voltage than slightly over because many LEDs cannot tolerate overvoltage. Also, you may need fewer than eight LEDs for each eye if you can find a more diffused lens style. The ones I used were the ultrabright type and were highly focused to a narrow field of view, which is why they required further diffusing to avoid hot spots.

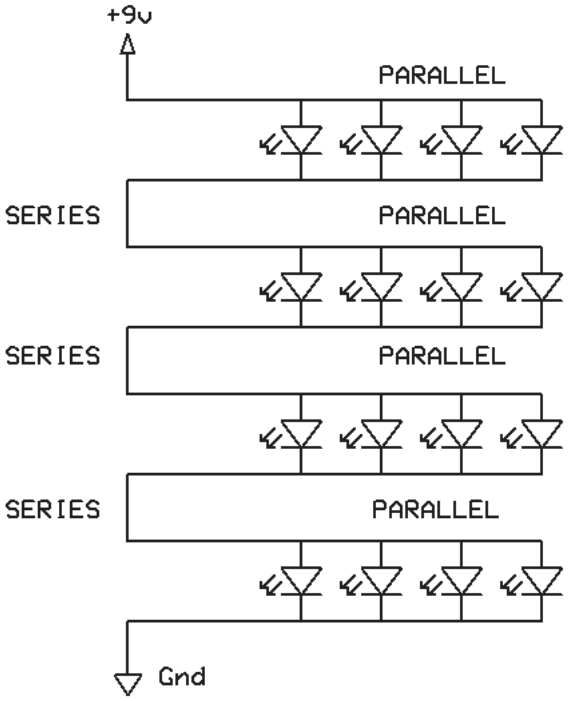


Figure 14-5 *Parallel and series LED wiring.*

If your LEDs are highly focused and very bright, then they may cast hot spots and shadows on your eyepieces when in use. The best way to find this out is to just light them up and place them about 4 to 6 in above your eyes while you wear the Ganzfeld eyepieces. You want a bright light source, but not one that will create spots that you can focus on. To filter out the hot spots, just cut up another pair of Ping-Pong balls and place them a few inches above the LED cluster to diffuse the light. Figure 14-7 shows the diffuser in action, and although the figure shows obvious light and dark areas, these were not visible while wearing the eyepieces and are mainly due to the way the camera took the photograph. White

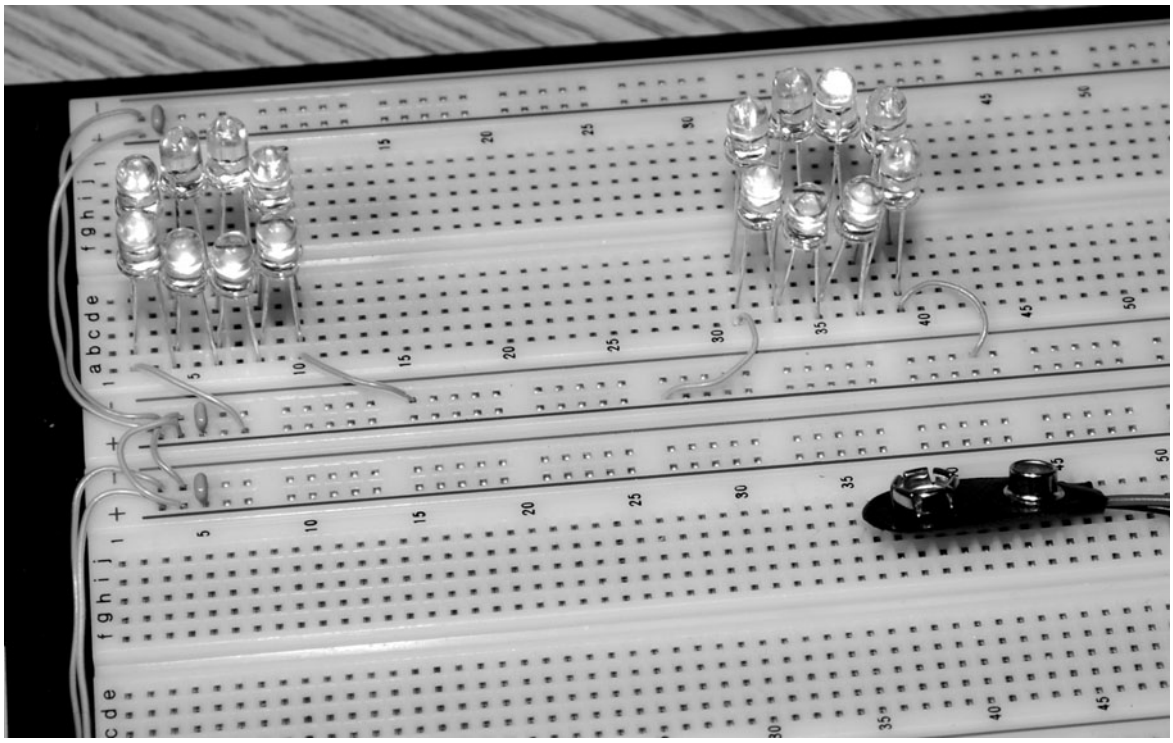


Figure 14-6 *Testing the parallel and series LED hookups.*

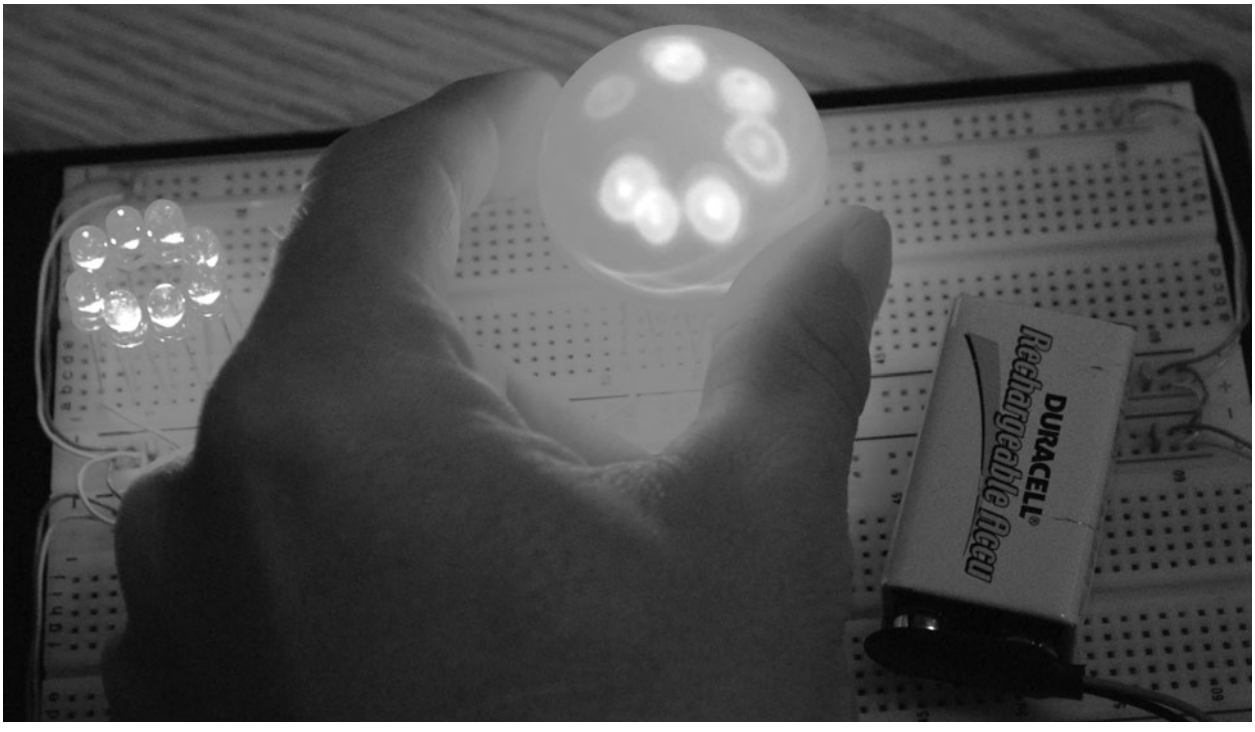


Figure 14-7 *Diffusing the LEDs to avoid hot spots.*

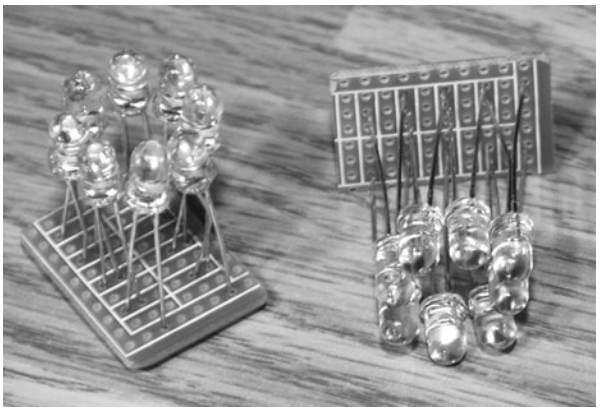


Figure 14-8 *Soldering the LED clusters to a perf board.*

Styrofoam and even a sheet of white paper also will work well as an LED diffuser.

Once you have figured out how many LEDs you will need, they can be mounted to a bit of perf board for later installation into a mounting tube of some type. Figure 14-8 shows my dual LED clusters after following the same wiring scheme used on the breadboard when testing them. In case you do not already know, LED polarity can be identified either by the longer lead (positive) or by the flat part on the housing (negative). This can come in handy if you have cut the leads short and find that one or more of your LEDs fail to light up when power is applied.

Since I wanted to be able to place the two LED lighting units over my head in various positions as I wore the Ganzfeld eyepieces, some type of housing was needed that could attach to a tripod for easy adjustment. I found a bit of PVC tubing, as shown in Figure 14-9, to fit the Ping-Pong ball halves perfectly, and two lengths were cut at the correct focal length for optimal light diffusion. At 5 in long, the LED light hits the inside of the Ping-Pong ball diffuser at an optimal angle so that no hot spots were present when pointing the light over the Ganzfeld eyepieces.

As shown in Figure 14-10, both the ball halves and the LED clusters were secured to the PVC

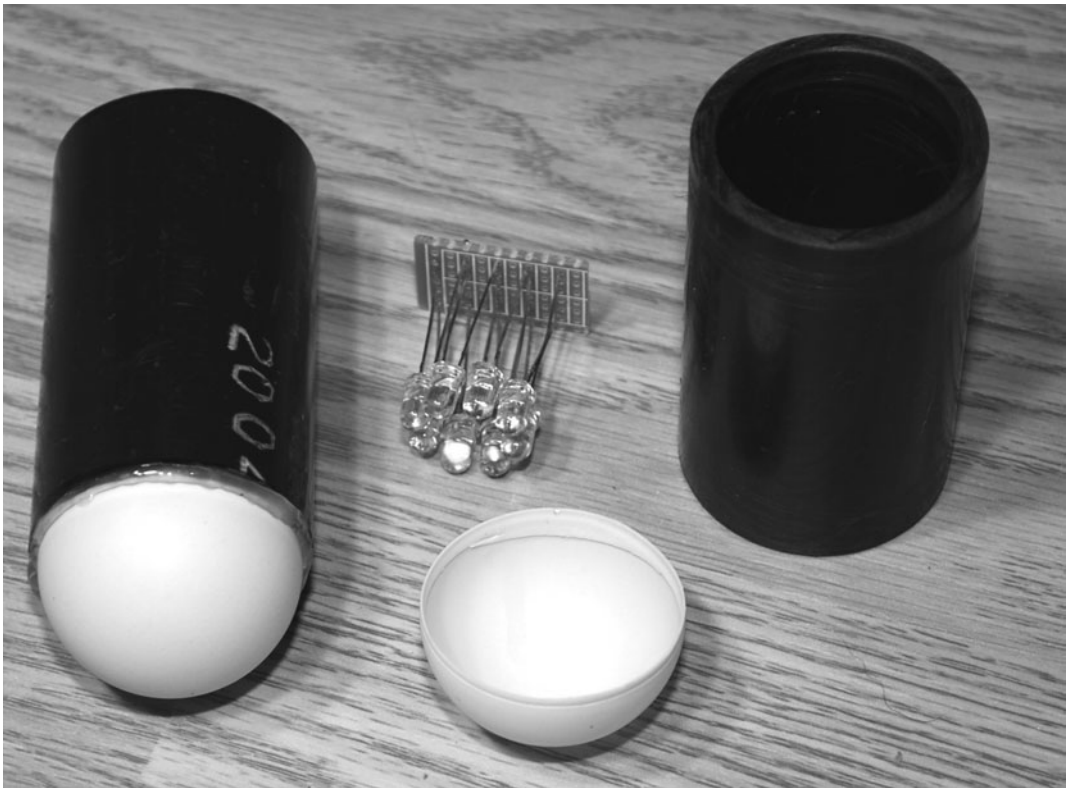


Figure 14-9 *Making housing for the components.*

Figure 14-10 *A hot-glue gun is an essential tool.*

tube using a hot-glue gun. A hot-glue gun is one of those "must have" tools for every electronics hobbyist and will create not only a secure bond but also one that can be broken if needed. Remember to drill any holes you need for wiring or other mounting hardware into the PVC tubing before you glue everything together. Besides PVC tubing, practically anything can be used to position the diffuser over the LED clusters. Cardboard tubes, plastic boxes, or even an old shampoo bottle washed out and dried thoroughly would work just fine for this job.

To make a simple mounting system that would attach to a tripod for easy adjustment, I cut a 2-ft length of PVC tubing as shown in Figure 14-11 so the two LED units could be mounted on either side. The diameter of the PVC tube was perfect to allow the distance between the two LED units to be the optimal distance for my eyes. Using a tripod to position the light source made it easy to find a comfortable place to use the Ganzfeld system without worrying about setting up the hardware. The tripod allows height adjustments from 2 ft to over 6 ft off the ground.

The completed light source is shown mounted to a camera tripod in Figure 14-12, ready to use with the Ganzfeld eyepieces. For extended periods of use, you may find it more convenient to use a 9-V direct-current (dc) adapter pack rather than the battery because multiple LEDs can become power-hungry, especially the high-brightness types. I found that a fully charged 9-V battery would run my lighting system for about an hour before it began to fade, so that was okay for short experiments where I did not plan to get into any really deep altered states. If you intend to really "go deep," then you most likely will have to switch to a dc adapter.

Considering that there is not much in the way of hardware to this project, it really does have some profound effects on a person's state of consciousness. Figure 14-13 is unfortunately lacking in color, so you cannot see that the red glow from the LED lighting system was very rich as I attempted to reach higher levels of consciousness. If you are interested in meditation or hypnosis, then the Ganzfeld device can be a powerful aid in reaching a trancelike state.

Figure 14-11 *Adding a support tube for tripod mounting.*

Figure 14-12 *The completed tripod-mounted light source.*

Figure 14-13 *Getting lost in a sensory-deprivation experiment.*

Maybe you want to experiment with telekinesis or PSI, attempting to recreate some of the well-known experiments done using hardware almost identical to what you have just built. How about trying some remote-viewing experiments? A *remote viewer* is a person under a trancelike state who can see a remote location as if viewing it through some type of invisible camera link. This may sound far out to you, but there must be something to it because the U.S. and Russian military have invested millions into remote-viewing research, often performing experiments with Ganzfeld devices. Further reading on the military's research can be found by searching for the StarGate Project on the Internet.

Well, I do hope that you enjoy your altered states, and if you discover some untapped power by accident, be sure to let me know. Oh, and don't use your new psychic tools and powers for anything too devious!

This page intentionally left blank

Project Fifteen

Alpha Meditation Goggles

Depending on our mental state, our brain is constantly oscillating at various frequency ranges. These oscillations, or *brain waves*, occur when numerous groups of neurons become coherent in their electrical activity. Alpha wave is the name given to the range of brain-wave oscillations that fall between 7 and 12 Hz (cycles per second). *Alpha waves* are also known as Berger's waves in memory of *Hans Berger*, who made the first known electroencephalographic (EEG) recording of a human subject.

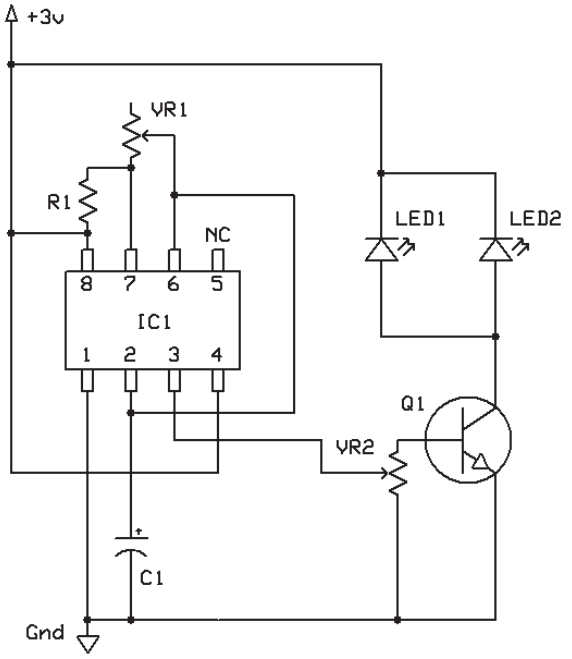
Alpha waves are present when we are in a relaxed state, often with our eyes closed, and bring about a heightened feeling of well-being. Interestingly, the earth's resonant frequency (also known as the *Schumann resonance*) happens to fall within the alpha-wave frequency range at 7.83 Hz. Some people have speculated that our minds would be "in tune" with the earth at this frequency.

The device presented here will let you cover the entire alpha-wave frequency range so that you can find your "magic" frequency when trying to relax or meditate. The purpose of the alpha meditation goggles is to coax your brain into synchronicity with the alpha-wave frequency of the flashing LEDs in order to reach a very relaxed state in a short time.

Please note that flashing lights (and video games) have been known to trigger epileptic seizures in a very small percentage of the population who have a rare form of epilepsy that seems to respond to external visual stimulus. Although extremely rare and only a risk to those who are known to have epilepsy or have not yet been diagnosed, it is worth noting this precaution before you build and use any device with bright flashing lights.

The schematic shown in Figure 15-1 will pulse the two or more LEDs that will be mounted to your goggles or mask. The 555 timer IC is working as an oscillator or pulse generator with its rate controllable via variable resistor VR1. You also can adjust the brightness of the LEDs by moving variable resistor VR2 to control the voltage to transistor Q1, which drives the LEDs. The working range of the oscillator can reach well below and well above the alpha-wave frequency, so you even could experiment with the other brain-wave groups as well, although alpha waves are the ones that are most useful in meditation and relaxation.

The schematic is very simple, but it is always a good idea to try things out on your solderless breadboard before warming up the soldering iron. Figure 15-2 shows the oscillator running on a breadboard and powering two LEDs. I found that



PARTS LIST FOR THE ALPHA MEDITATION GOGGLES
ICI = NE555 TIMER Q1 = 2N3904 OR SIMILAR NPN TRANSISTOR R1 = 1K C1 = 10UF LED1,LED2 = HIGH BRIGHTNESS LEDS VR1, VR2 = 10K VARIABLE RESISTOR

Figure 15-1 *The alpha meditation device schematic.*

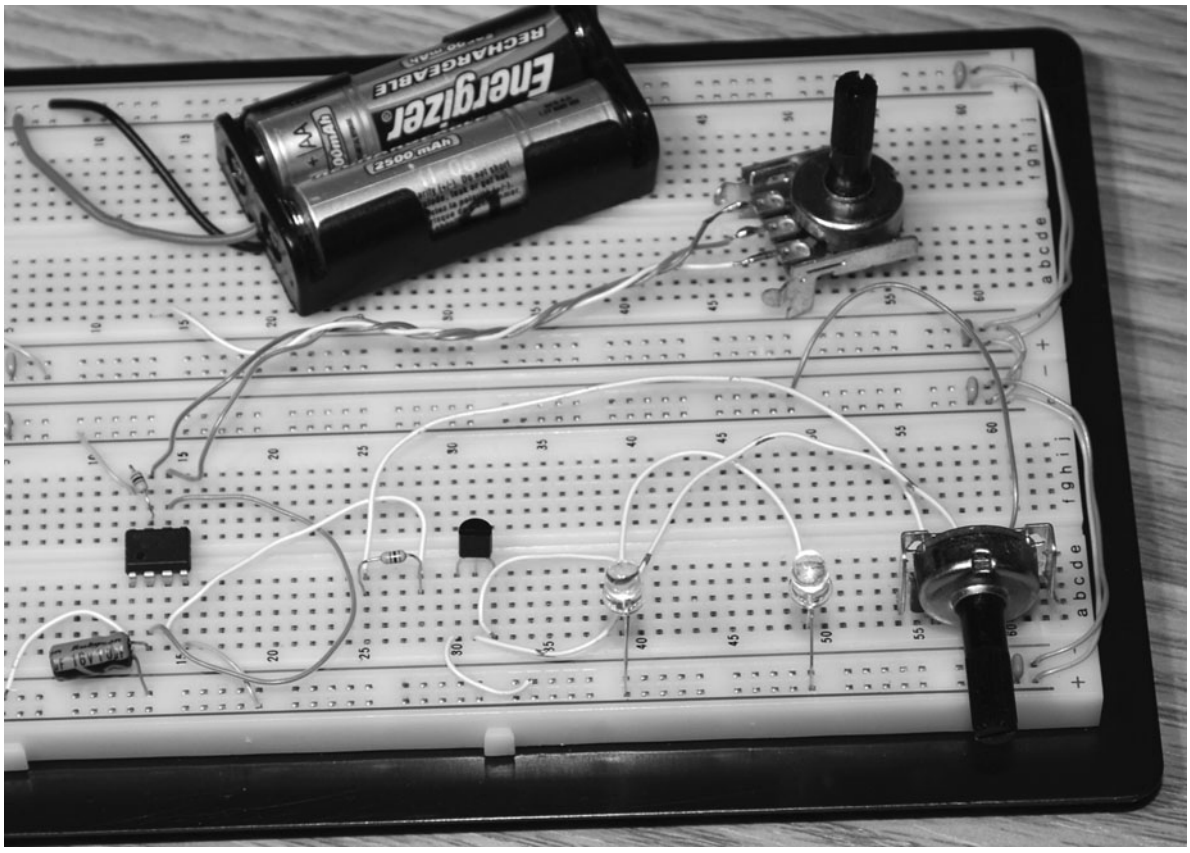


Figure 15-2 *Building the circuit on a breadboard.*

six LEDs (three for each eye) were best with the ones I was using because they could be set at a lower brightness level and cover more area that way. You will have to experiment with your LEDs, but do make sure that they are all the same make and model so that one side is not brighter than the other. The circuit is designed to run at 3 V to make battery power easy, but the 555 timer IC would be happy all the way up to 12 V. For operation above 3 V, you certainly would need to add resistors (150 Ω or more) in series with your LEDs in order to reduce the current.

Since the oscillator has no readout, you will need to use a frequency counter or oscilloscope to set the frequency into the alpha-wave range between 7 and 12 Hz at least once before use. You could just set your oscillator for 10 Hz, which is basically in the center range or experiment with the completed goggles and find your preferred frequency or the one that makes you feel most relaxed. For me, 10 Hz seemed to work just fine, but if you want the ability to adjust this at any time, then just make VR1 accessible as a control when you build the circuit into a cabinet. You also can feed an extra pair of wires out of pin 3 of the 555 timer and ground so that your frequency counter always can be connected when you want to know the actual working frequency. Figure 15-3 shows my circuit oscillating at 10 Hz, as reported by my basic multimeter.

There are many ways to get the light over your eyes, and using an old ski mask or a pair of swim goggles is a good method. The alpha meditation system is not intended for long-term use, but you still will want a setup that is comfortable and has proper air circulation, so the goggle straps are best replaced by loose elastic and should have small holes drilled in them to let the air flow freely. I decided on using the small swim goggles shown in Figure 15-4, and the straps were so long that they were easy to adjust loosely. You also may have to experiment with the type and number of LEDs that work best over your eyes so that you can achieve maximum light without having the light source be so bright that it is

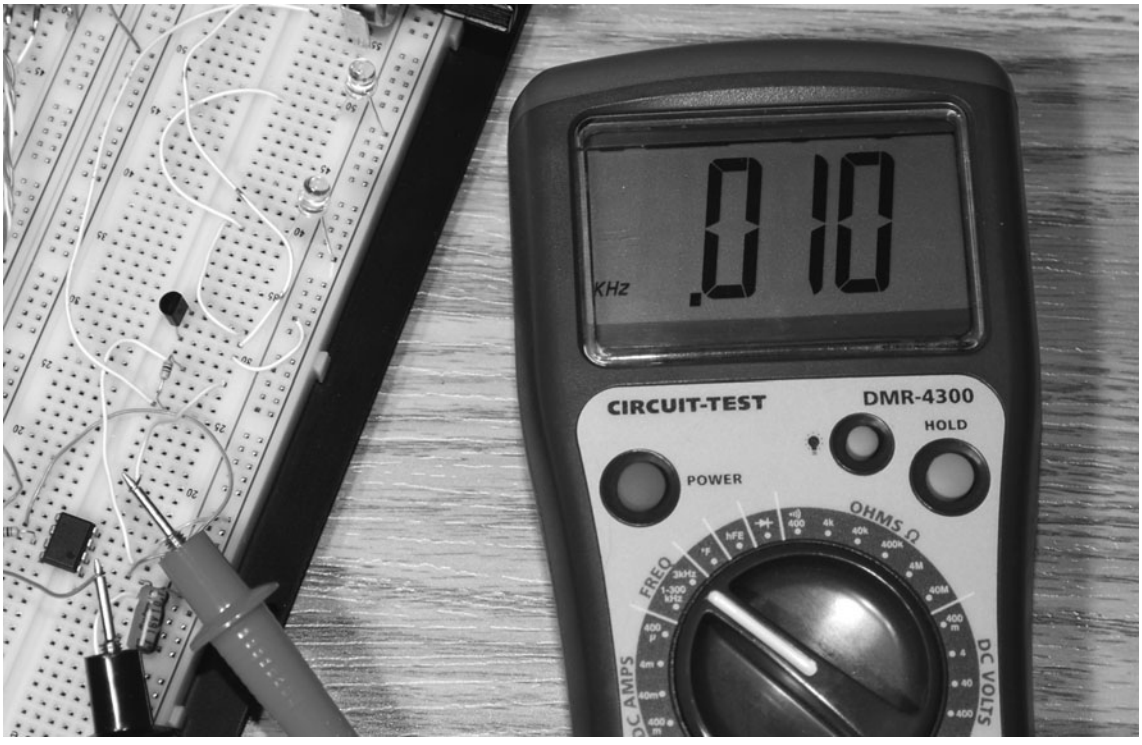


Figure 15-3 *Finding your magic frequency.*

Figure 15-4 *Choosing LEDs and a mounting system.*

uncomfortable. Your eyes should feel saturated with light, but not so much that it feels like a camera flash going off in front of your face. I found that three LEDs in each eye were good, and I still had room to turn up the brightness adjustment, if needed. A hot-glue gun is a handy tool that can be used to mount almost anything to anything else, so that is how I intended to secure the LEDs to the goggle lens.

The number of LEDs, brightness, and color all will have an effect on the way the light is delivered through your closed eyes and the lens (if there is one) on your mounting system. Red LEDs will transmit more light through your closed eyelids than any other color because your eyelids act like a red filter when closed. Any color will certainly work, but red seems least distracting and natural, and since color therapy is not part of this project, red seems like the most logical choice. Figure 15-5 shows the two small three-LED clusters made by soldering the three LEDs in parallel to a small bit of perf board. In this circuit, all LEDs are in parallel, and you can have as few as two or as many as your transistor can source current to. As you exceed your driver transistor's capacities, the LEDs will become less bright, and overheating of the transistor could occur. Of course, you could just use more transistors to drive more LEDs, but be aware that this also will shorten the life span of a battery charge.

Using the hot-glue gun, the LED clusters are affixed to the center of the goggle's lenses, as shown in Figure 15-6. Some goggles may not have their lens centers over your eyes' centers, so some testing may need to be done first. If you put on the goggles and look into a mirror, you can make a mark where the exact center for your eyes will be and use that as a guide. A nice side effect of using the hot glue is that it creates a bit of a diffuser to help spread out the light evenly when using high-brightness and highly focused LEDs. This diffusing effect can be exploited further by first melting a blob of glue over the lens before gluing the LED clusters down, further adding to the diffusing effect.

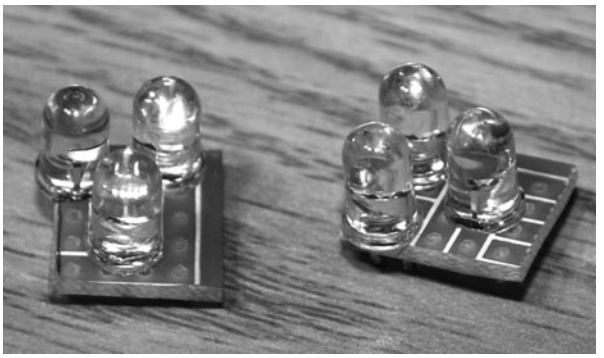


Figure 15-5 *Choosing LED color and quantity.*

I decided to add a jack to the goggles, as shown in Figure 15-7, so that they could be used for other experimentation and connected to any device capable of driving them. A ⅛-inch mono headphone-style jack simply was soldered right to the small perf board, and a matching male plug was added to the end of the wire coming from the oscillator unit.

The small circuit board, battery pack, switch, and variable resistor are mounted into a more permanent home, as shown in Figure 15-8. I decided to leave my system set for 10 Hz, so there is only one cabinet-mounted variable resistor, the one that controls LED brightness. A small trimpot variable resistor is added to the board and preset to the desired frequency before placing all the parts in the enclosure. If you plan to experiment with frequency adjustment, then just add a second cabinet-mounted variable resistor.

With your alpha meditation goggles ready for use (Figure 15-9), find a quiet and comfortable place to lie down and reach the height of Zen! Okay, it may take you a bit of practice and patience to reach that level of meditation, but synchronizing your brain waves into the alpha state will surely help you along the way. If you are planning to use the system in a dark room, then let your eyes adjust for about 15 minutes before using the goggles so that you can set the brightness for your already-adjusted eyes. The light should be bright enough to feel "mesmerizing" but not so bright that it feels uncomfortable in any way.

Looking like the cover of some cheesy sci-fi movie, Figure 15-10 demonstrates how our mind-hacking projects can be entertaining not only to the user but also to the onlookers! Flooded in a

Figure 15-6 *Mounting the LEDs to the goggles.*

Figure 15-7 *Wiring the LEDs for external access.*

Figure 15-8 *Putting together the oscillator unit.*

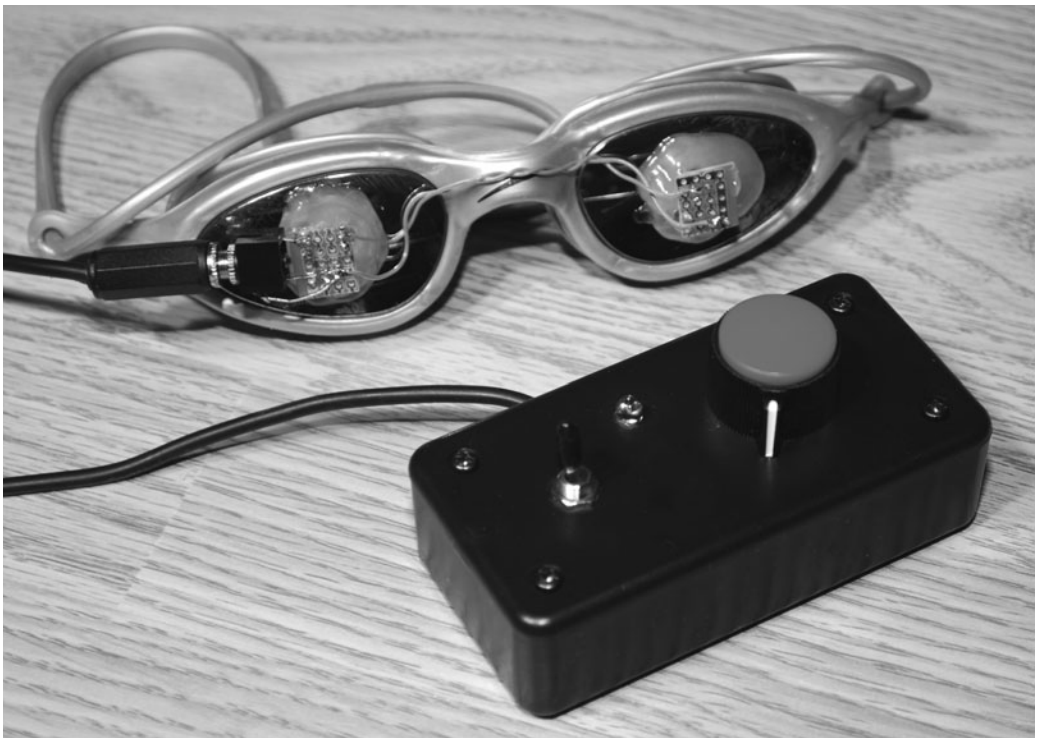


Figure 15-9 *The completed alpha meditation goggles.*

flickering pool of red light, I relax and drift into a state that feels much like a land between waking reality and sleep, using the device to power nap during the day. The alpha meditation goggles certainly do help when it comes to feeling relaxed, and there is a definite feeling of relaxation after a bit of use.

Some other experiments you might want to try with this device include the addition of sound to the output, using two oscillators to drive each eye independently, or the addition of some type of alternating color to the light source. Sound pulses can be added simply by placing a set of headphones onto the output of the LED driver transistor through a 1K resistor. Each time the LED pulses, a beat will be heard in the headphones. In the next project, we will transcend further to test our clairvoyance abilities.

Figure 15-10 *"Jacked" into a more relaxing state.*

This page intentionally left blank

Project Sixteen

Clairvoyance Tester

Some of us know at least one person who claims to have some clairvoyance or an uncanny ability to predict future events. Being of the "believe nothing without proof" discipline, I decided to build a simple box that would allow my all-powerful psychic contacts to prove their worth or, as it seems more often, disprove their abilities! The word *clairvoyance* comes from seventeenth-century French and means *clair* = "clear" and *voyance* = "visibility." Those who are clairvoyant are said to possess some extrasensory ability to see future events or visions before they actually occur. To date, there has been no undisputable proof that anyone possesses this ability, and the practice is not normally accepted by the general scientific community.

This device allows your subject to use his or her amazing powers to "visualize" a number between 0 and 9, enter it on an LED readout, and then press a button to generate a completely random result on another LED readout. If the subject gets a match, then a third LED readout adds a 1 to the current score. If the subject's score is already zero, then he or she just stays at zero score. Since there is a 1 in 10 chance that the subject's guess will be correct, it is unlikely that his or her score will be anything other than zero or stay above a few points for any length of time. If your subject is truly clairvoyant, then surely this basic test will be easy for him or her, and the score should remain well above 3 after a few minutes of testing. Of course, this is almost impossible, and anyone who can score anything above 1 on a regular basis is a truly amazing psychic with unreal abilities.

The schematic for the clairvoyance tester, shown in Figure 16-1, does appear to have a lot of connections, and this is mainly due to the triple seven-segment LED displays used to show the guess, the result, and the score readouts. There are not enough pins on the small Atmega88 microcontroller to connect all three LED displays at once, so the three transistors (Q1, Q2, and Q3) switch between the three displays at such a fast rate that they all seem to be on at the same time. This time-sharing trick is how most LED displays work when there are hundreds or thousands of LEDs to control and only a limited number of connecting wires. There really is no limit to how many seven-segment LEDs you can connect as long as you have sufficient drive current and microprocessor speed and can spare the extra IO pin for each common connection. Other than the LED-display circuitry, the only other part of this circuit is the 4-MHz crystal resonator and the two pushbutton switches that allow the subject to select a number and then run the randomizer.

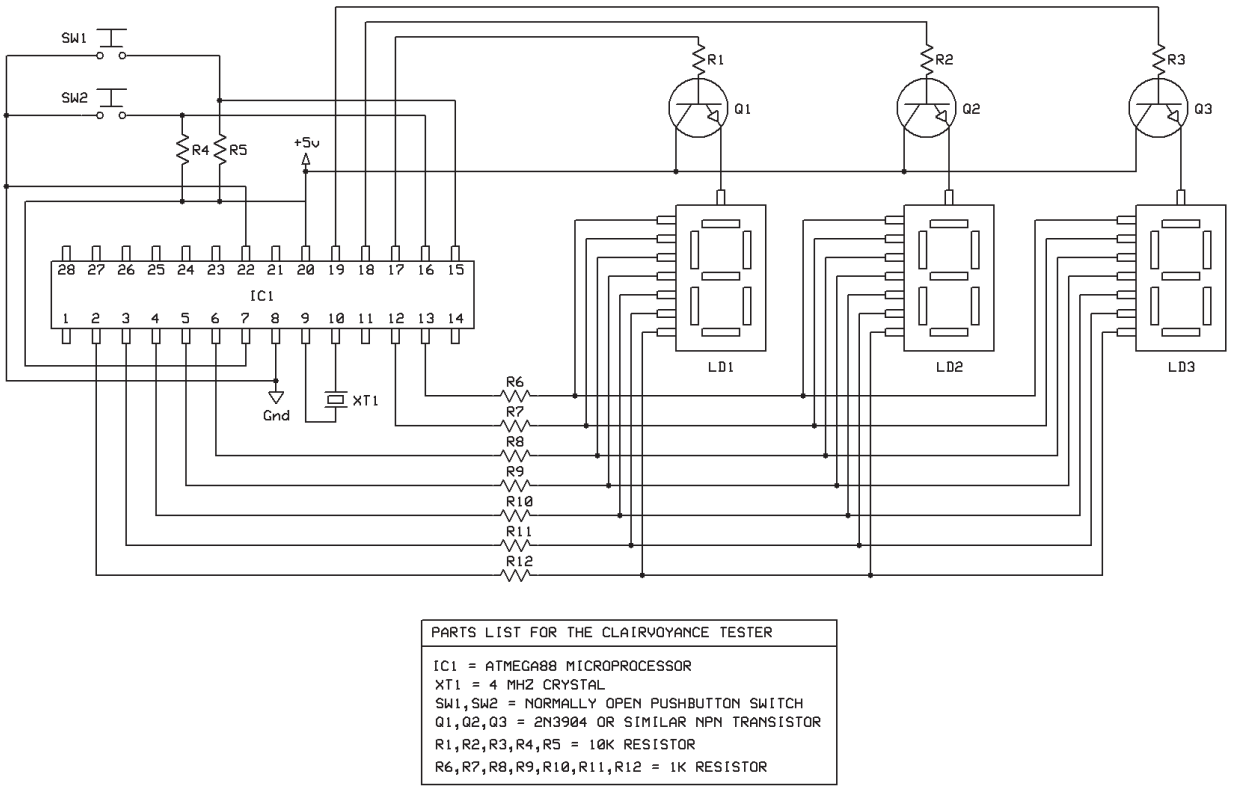


Figure 16-1 *The clairvoyance tester schematic.*

The seven-segment LED is a very common and inexpensive display that you are already familiar with because it is used in everything from your digital clock to the panel of your microwave oven. There are actually eight segments if you include the decimal point, but we don't use it in this application. These displays come as either stand-alone blocks or chained blocks containing more than a single digit. Some LED displays even have alphanumeric capabilities or multiple dots so that they can make any character imaginable. Figure 16-2 shows some of the LED displays that I found in my junk box when looking for three identical digits for this project. LED displays are either *common cathode* or *common anode*, which means that either the positive connections or the negative connections all go to a common point. It really does not matter which type you use as long as you install them in your circuit so that current is flowing in the proper direction. The ones I am using have part number HDSP-5501 and are basic green displays with a common anode, so the positive connections on all internal diodes are tied together.

To use a common-cathode LED display in this circuit, you would have to tie the driver transistor emitters to ground and connect the common cathode to the collector instead. Other than that, it is only for aesthetic purposes that all LED displays should be the same.

There are a lot more wires to connect in this project, so a breadboard prototype is important to ensure that the display shows the expected results, not some random hieroglyphics.

Figure 16-3 shows the schematic up and running with the triple-LED display reading "662," the results of over 2 minutes of clairvoyance testing. The first digit is my guess, and the second digit is the result of the random-number generator. The third digit is my score,

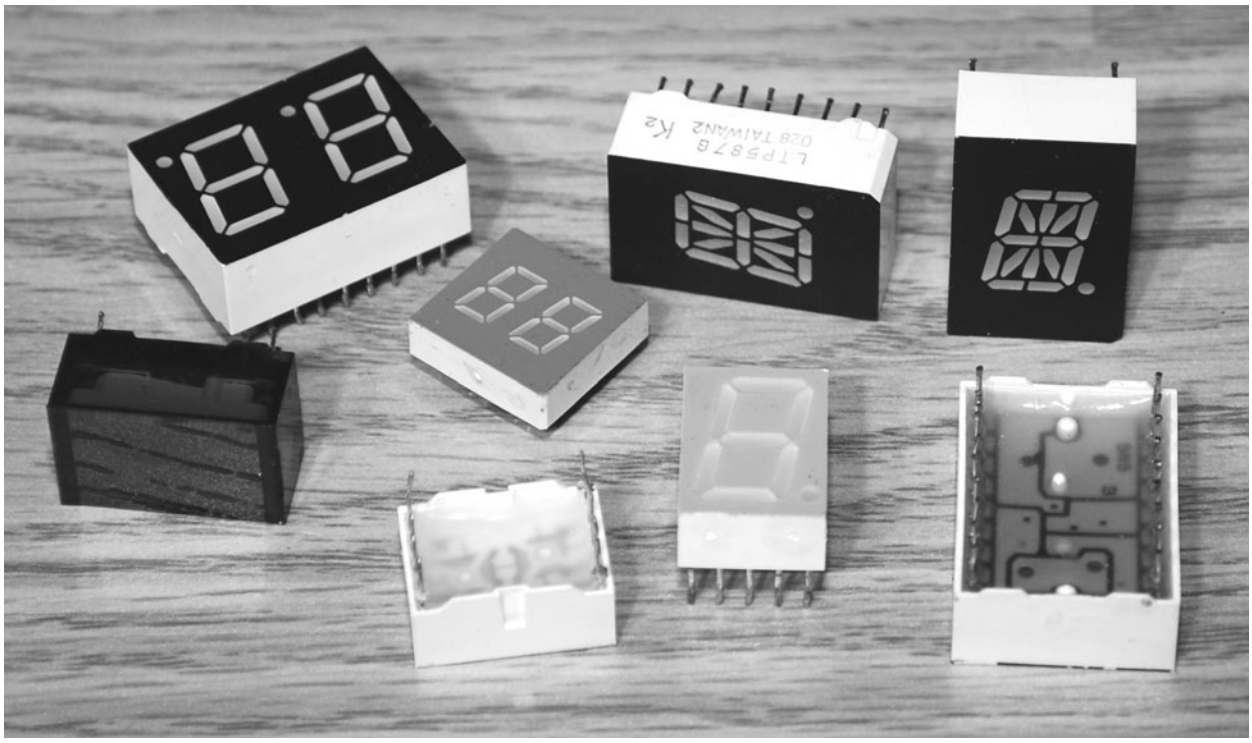


Figure 16-2 *Various segmented LED displays.*

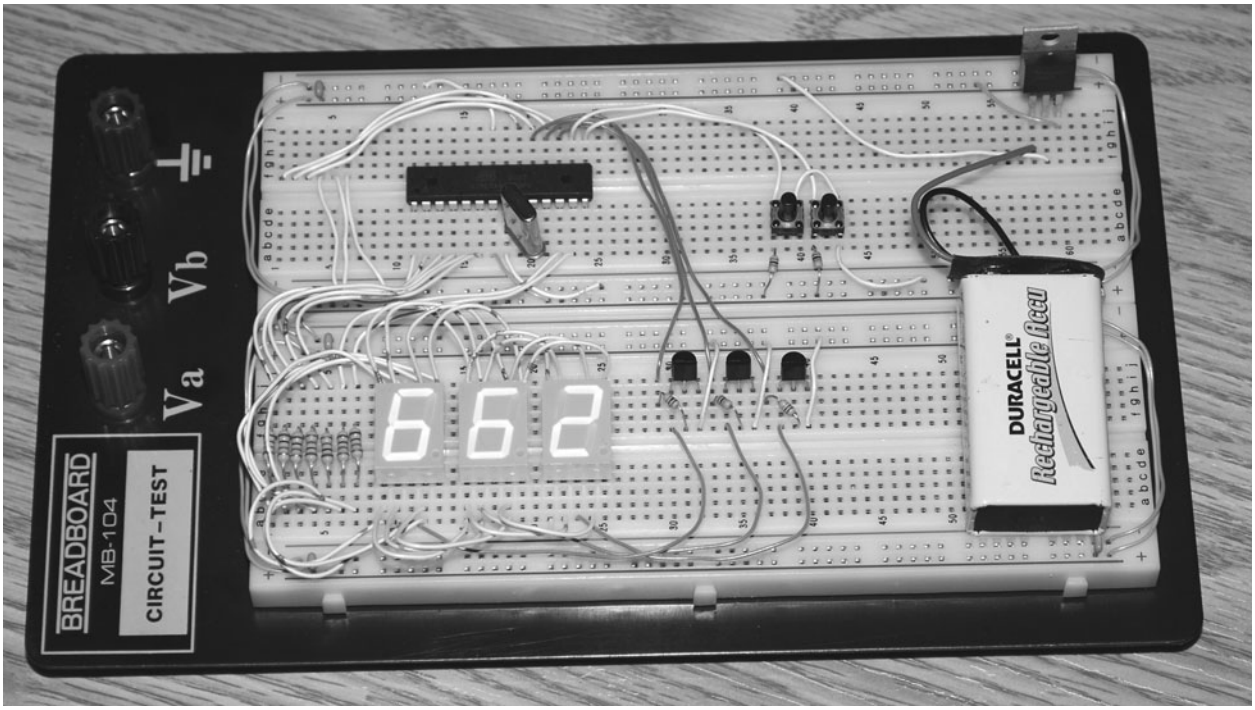


Figure 16-3 *Breadboarding the clairvoyance tester.*

which is usually 0 for the most part, so I thought I should grab a photograph of the unit. On the next two guesses, my score dropped back down to 0, so my belief that lotteries are a complete waste of time and money still holds strong. Notice the bank of seven current-limiting resistors to the left of the LED displays and the driver transistors to the right. Ninety percent of the wiring and hardware is just for the LED displays. I am also using a 7805 regulator to drop the 9-V battery down to 5 V, although the Atmega88 also would run fine from only 3 V supplied by a pair of AA batteries. As you lower the voltage, you may need to alter the value of the current-limiting resistors if your LED display is not bright enough. For 3-V operation, 500 Ω may be a better value than 1K.

If you have a bunch of LED displays and cannot locate a datasheet, then there is a simple way to figure out the connections to all pins. You will need a 5-V supply to your breadboard, a 1K resistor, and a few wires. Start by connecting the first pin to the resistor and then to VCC. Now take a wire from GND and try all other points to see if you get any segment to light up. If not, place the resistor to GND and the wire to VCC and then try again on all points. Keep moving along all the pins until you find the common pin and its polarity.

To keep the cabinet size to a minimum, I broke the circuit into two parts: the LED display perf board and the microcontroller perf board, as shown in Figure 16-4. Once all the wires were added, I was able to fold the two perf boards together like a stack to reduce the amount of space needed inside the small black box. The top of the box will have a square cut to fit the LED display by cutting over a line using a very sharp razor knife. A notching tool is also great for cutting out square holes for displays and other things in a plastic or thin-metal box. With a notching tool, you just drill a hole at each corner and then nibble your way across a line, taking one small bite at a time.

Figure 16-4 *Deciding on a cabinet and board layout.*

The completed clairvoyance tester is shown in Figure 16-5 displaying what is probably my highest score since building the unit. My original guess was a 7, and I just pressed the "randomize" button three times, and by chance, it resulted in a 7 each time. The odds of this happening are the same as matching any number three times in a row and quite rare. It would be the same as throwing a 10-sided die three times in a row and guessing the result each time before throwing. If a normal "nonclairvoyant" person were to use this device all day long, the score would remain at 0 most of the time, with the odd spike up to 3 or even 4 while "fluking out" once in a while. If you can find a person who keeps his or her score above 0 consistently, then please tell him or her to get ahold of me immediately because I will have a lot of work for him or her to do, such as choosing lottery numbers and reading the stock market. Anyone capable of beating the odds certainly will be living a life of luxury and fame!

Now let's look through the Basic source code that will be compiled into the Atmega88 microprocessor. There is also more information regarding microcontrollers and programming languages in Section Two with the project called "Lucid-Dream-Mask Controller" in case this is your first attempt at programming a microcontroller. I will step through each block of the program code, which is shown in Listing 16-1.

Have a read through the complete Basic source code of Listing 16-1 provided in the appendix so that you can get an idea of what the code is doing. If you are an experienced programmer, then this trivial code is probably something you could write in 15 minutes from scratch, but if you have never written a program in your life, not to worry because Basic is called that for a reason and is easy to understand. The lines in the program listing that start with an apostrophe are comments, and I will explain the code in blocks after each comment.

Figure 16-5 *Showing off my "impossible" score!*

The code following this comment is required to tell Bascom that we are going to target the Atmega88 device and that our clock will be an external crystal resonator running at 4 MHz. Telling the compiler your clock speed becomes important when using commands that deal with timing-sensitive routines such as serial transmission or analog-to-digital readings. Defining the device also helps the compiler to generate user errors that have to do with input/output (IO) pins. In this way, you can't accidentally try to toggle an IO pin that does not exist on the actual device. Critical timing is not an issue in this program, but speeds over 8 MHz may make it difficult to press the buttons without adding to the delays.

This block of code sets up the pins that will connect to the LED display (outputs) and to the two pushbuttons (inputs). You can change these around any way you like, but the way they are now makes wiring a lot easier because there is some order to the IO pins.

Basic uses *variables*, which are letters or words used to hold values. I like to use single letters such as *A*, *B*, and *C* for simple programs such as this one, but when you are working on a large, complex program, the use of more descriptive variable names is recommended. "TIMER2" and "REDLED1," for example, would be descriptive variable names that make a lot more sense in a huge block of code. Variables are also defined as the number of bits they are to contain, so in our code, "A" and "B" are 16-bit "Words," which can contain a value between 0 and 65535. Variable "C" is an integer that can range in value from –32768 to +32767. Variables "D" and "E" will only contain values from 0 to 255, so they are bytes. Although you could just define all variables using larger data types, this is a waste of memory space and will slow down your code. Also notice the odd variable name "___rseed"; this is a reserved name and has special meaning to Bascom. When you place a value in "___rseed," you mix up the random-number generator, which is important when dealing with software-generated random numbers. By *seeding* the random-number generator, you ensure that the

sequence will be random each time the program is run. Without this seeding, the same sequences of numbers would be generated each time you powered up the microcontroller. Although this random-number generation is still not purely random, it is certainly good enough for this application.

This block of code sets up an array of 10 values for the variable "LED." Although this may seem confusing at first, the values correspond to which of the seven segments will be lit to display a particular number. To make this a little more confusing, the value in parentheses is actually 1 higher than the represented numerical value, and to light a segment, we want a low bit, not a high bit. Thus the value of "Led(9) = 0" says that to display the decimal number "8," we want all bits to be off. This means that all segments light up, and an "8" will be displayed. It can be a bit of a chore computing these values by adding port bits, but after you have done it once for your segment, the hard work is completed.

Everything from here on will happen continually until the word "Loop" is reached, which causes program execution to start again where it first encountered the word "Do." This is called an *endless loop* because it never stops unless forced to by another command or an error.

Here, we simply check to see if the user has pressed the "Randomize" button, which starts the routine that flips through a bunch of numbers like a casino slot machine. Since the button inputs are tied to VCC through the 10K resistors, a value of 0 means that a button is depressed. If a button is pressed, the code will jump to the "Roller" subroutine, which is discussed later.

This code block checks to see if the user pressed the button that changes his or her "guess" number and then increments the variable "A," which is used to hold the number. If the value of "A" goes beyond 9, then it is reset to 0 because the display can only draw single digits. Since microcontrollers are working in millions of instructions per second, this routine also does something called switch debouncing so that the microsecond state between

switch contact and noncontact is not read as multiple presses. To debounce the switch, the code simply calls the "Ledshow" routine as long as the user is still holding the button down, which actually creates a bit of a delay and avoids false contact readings. Anyone working with microcontrollers and switches is definitely familiar with switch debouncing.

Here, the random-number generator gets another seed number, which is just a rolling counter, made of the word variable "___rseed." This ensures that there is no way you will ever learn to read a pattern in the random-number generator so that no one can trick this device.

Since the three LED displays need to be refreshed one at a time on the shared bus, this routine is called here each time the program loops, which is indefinitely. The "Ledshow" routine (discussed soon) ensures that something is always shown in the three LED displays.

The "Loop" command causes program execution to jump back to the main routine where the command "Do" was encountered. This is the end of the infinite loop, and all other commands beyond here must be called by either "Goto" or "Gosub" commands.

This is the routine that displays a digit on each of the three LED displays. A little trick called *persistence of vision* is used here to switch between displays so fast that your eyes think that they are all on at the same time. As you can see, only one bit of the three "Portb" pins is on at a single time, and then the variables "A," "B," and "C" are sent to the display for only 2 ms each. Because 2 ms is so fast, it appears that each display is on all the time, and the seven-segment lines can be shared, saving valuable IO overhead. Since Bascom array variables start at 1, not 0, the line "D = A + 1" adds one to the array pointer so that the value in "Led(d)" is the same as the decimal value we want. This conversion just makes it easier to understand the code, especially when trying to compute the segment bits from

scratch the first time. Once all three displays have been lighted for 2 ms each, this routine just "returns" to where it was originally called.

This is the routine that starts the slot-machine-style random-number generator that will display the random digit on the second LED display. The line "For E = 1 To 50" indicates that the random number will "roll" past 50 times, making a more interesting display, as if the device were actually rolling a die or spinning the mechanics in a slot machine. Each time, the "Ledshow" routine is called another 10 times just to slow the display of random numbers down enough to actually see them go by.

After 50 random numbers flip past on the second LED display, this next block of code then checks to see if the "guess" displayed on the first LED matches the latest random number displayed on the second LED. If there is a match ("If A = B"), the score variable "C" is incremented by one. If there is no match and the score is above zero, it is decremented by one. The line "If C = 255 Then C = 0" checks to make sure that the score variable never goes below zero because a value of 255 means that the byte variable has wrapped.

So this is how the Basic code works. A lot can be accomplished in a very few lines using Basic, and since microcontrollers work at nanosecond speeds, you have a lot of power at your fingertips.

Now you can weed out the false psychics from those who possess real clairvoyant powers! It has been suggested that under the effect of the Ganzfeld device presented earlier, extrasensory perception (ESP) and clairvoyance may increase for certain people, so this might be an interesting experiment to try. Have your subject guess a number, and then enter it on the clairvoyance tester for him or her, each time letting the machine keep score while the subject meditates. Maybe you want to train your ESP so that you can get the upper hand next time you play a dice game? All you have to do is alter the code slightly so that it displays only the digits 1 to 6, and your clairvoyance tester is now a dice-roll simulator. It even would make a great decision maker or electronic dice roller for game night!

Have fun weeding out your false wizards and psychics with this project. If you do find someone to be especially clairvoyant, then please tell them to get in contact with me for a job interview immediately. My phone number is . . . wait, let them guess it! Next, we will try our hand at hypnosis.

This page intentionally left blank

Project Seventeen

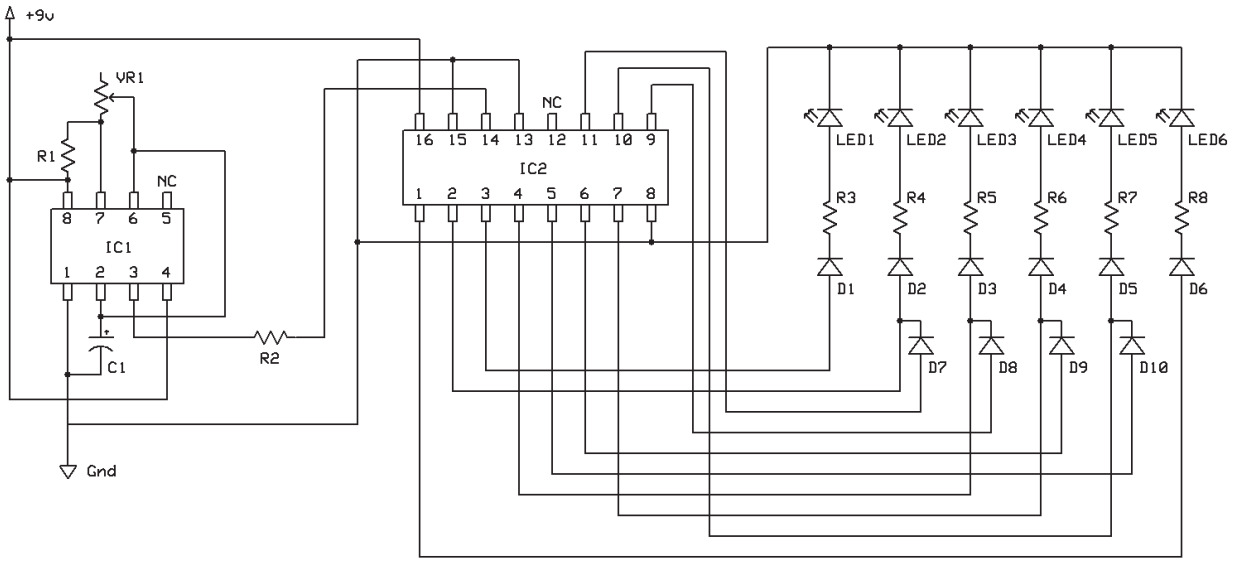
Visual Hypnosis Aid

When I began to research hypnosis in order to come up with a nice single-paragraph description of what it is all about, I discovered that there is currently a huge amount of controversy as to what the definition of hypnosis really is. Some people say that it is a trancelike sleep, whereas others say the exact opposite, claiming that it is more of a heightened conscious state. So rather than wave the flag and take sides, I will just concentrate on the methods and devices used to help a hypnotist bring his or her subject under hypnosis and present an easy-to-build visual hypnosis aid that you can use to learn the art yourself.

Obviously, hypnosis is a very powerful tool that can be used to help us with our inner problems, break us from long-time habits, delve into the subconscious, and unlock the mind in ways not possible without it. Hypnosis, like all skills, can be learned by anyone willing to read the literature, and you can even practice on yourself, exploring self-hypnosis. Of course, it is highly unlikely that you will have your friends under your complete control by speaking the "secret phrase," nor will you learn to mass-hypnotize a group of people into running around barking like dogs because this is the stuff of movies and rehearsed magic!

This device will replace the classic pendulum, or "swinging watch," that is so often shown in movie scenes involving hypnosis. Often taken to the extreme, the patient simply will fall into a trance as the doctor lets his or her watch swing back and forth in front of the patient's eyes. This scenario is highly unlikely, but the use of a fixation point to keep the mind focused during hypnosis is a viable method of induction. Instead of a swinging object, this device simulates a back-and-forth motion using six bright LEDs, allowing it to be used in a low-light room or in the dark.

As shown in the schematic (Figure 17-1), the six high-brightness LEDs are actually connected to the 10 output lines of the 74HC4017 decade counter (IC2). Since the goal is a back-and-forth motion much like a swinging pendulum, the sequence counts as follows: 1, 2, 3, 4, 5, 6, 5, 4, 3, 2 and then repeats, which is why we need 10 sequential lines and six LEDs. The 4017 is clocked by the 555 timer (IC1), which can have its rate controlled by adjusting the variable resistor (VR1). The range of speed is quite large, so you can create a nice, slow back-and-forth light sequence or have the LEDs moving so fast that they almost seem to flicker. The 10 diodes connected to the outputs of the 4017 block



PARTS LIST FOR THE VISUAL HYPNOSIS AID
IC1 = NE555 TIMER
IC2 = 74HC4017 DECADE COUNTER
C1 = 1UF
VR1 = 10K VARIABLE RESISTOR
R1,R2 = 1K
R3,R4,R5,R6,R7,R8 = 1K
D5 - D10 = 1N4003 OR SIMILAR DIODE
LED1 - LED6 = HIGH BRIGHTNESS LEDS

Figure 17-1 *Visual hypnosis aid schematic.*

current from feeding back into the output pins when counting back over the last four digits to reverse the sequence. The resistors connected in series with the LEDs are there to limit the current draw and can be replaced with a lower value if you want your LEDs to give off more light. There is a limit to how much current the 4017 can source and how much your LEDs are going to tolerate, so the values presented here are a good start. If, for some reason, you want extremely bright LEDs, you will have to drive them with a transistor rather than taking current directly from the outputs of the decade counter.

The visual hypnosis aid device is shown working on the solderless breadboard in Figure 17-2. The circuit is quite simple to build and tolerant of supply voltages from 3 to 12 V, although you may have to adjust the current-limiting resistors to set the optimal brightness for your LEDs. The high-brightness type of LEDs work best for this device, and they will allow you to project the light onto a wall or screen for easier viewing, as will be shown later. Color is totally up to you, and there are many ultrabright LEDs available in all colors and white. A single 9-V battery is used to supply power because this was the most convenient setup.

The circuit was transferred from the breadboard to a small perf board, as shown in Figure 17-3, after deciding on what type of LEDs would work best. This LED sequencer is a cool gadget that can be used for all kinds of display purposes, such as electronic props, computer case mods, or anything that would look better with a little light show. If you want to use more that six LEDs, you can double them up in parallel, but in the end, you still will only have 10 steps and may end up needing a driver transistor for each step to handle the extra current.

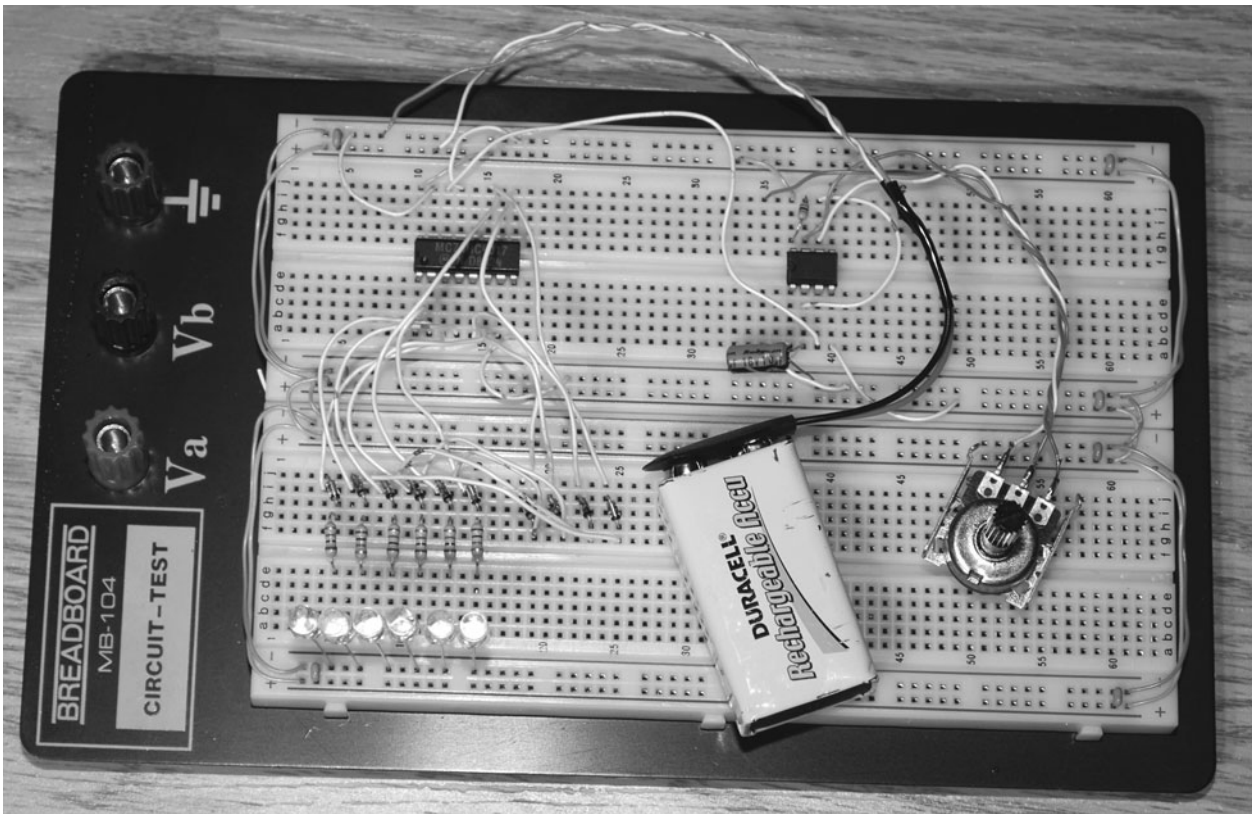


Figure 17-2 *Breadboarding the circuit to test the LEDs.*

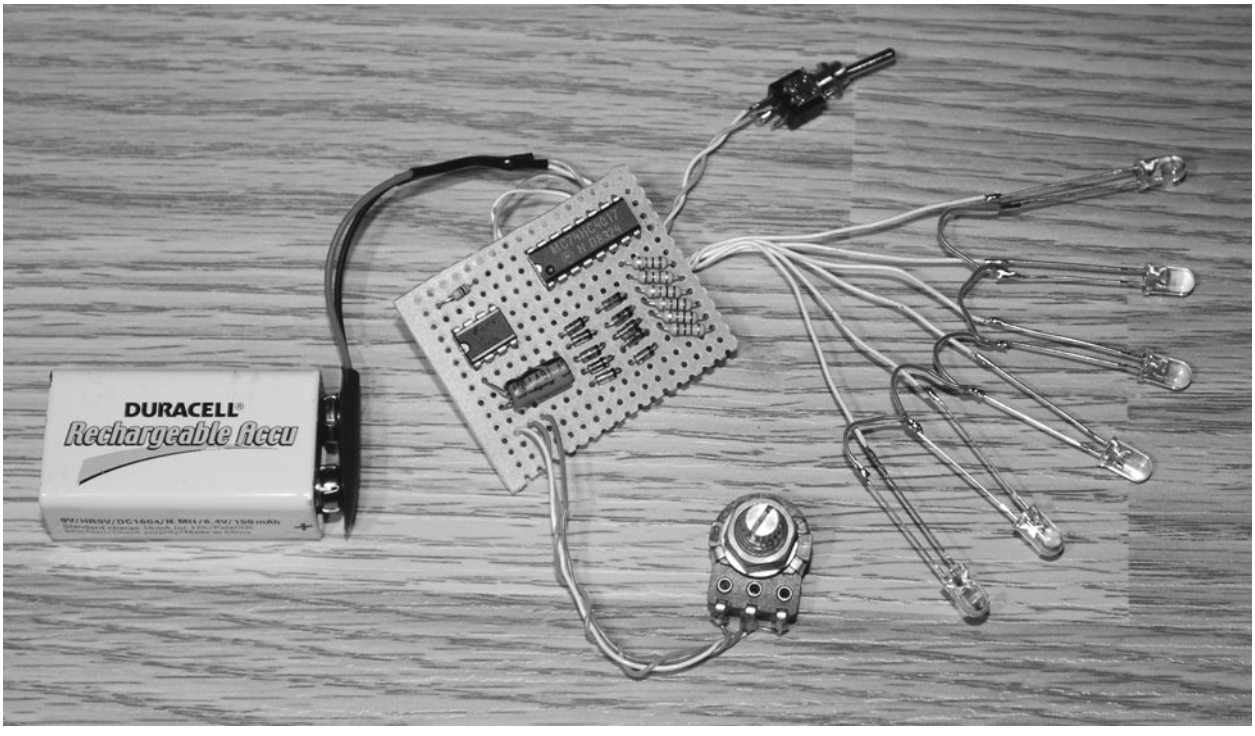


Figure 17-3 *The LED sequencer circuit ready to run.*

There are numerous configurations this device can take, and it really depends on how you want to deliver the light to your subject's eyes. The distance between each LED and the focal range of the LEDs will affect how far away the unit can be from your subject. I found that a spacing of no more than 1 in between the LEDs was fine, and anything beyond that began to ruin the smooth back-and-forth effect of the unit. To place the LEDs at a greater distance apart so that the device will work when it is placed farther from your subject, you will need some type of diffusing lens to spread out and smooth the beam from the high-brightness LEDs. Figure 17-4 shows the all-in-one configuration I used for my visual hypnosis aid. The spacing between LEDs is about 1 in.

Although the single boxed unit worked well in a dark room if placed about 2 ft from my eyes, I wanted the ability to experiment on more than one subject at a time, so I came up with a simple method that would expand the size of the display by projecting the light onto a diffuser screen. This setup allowed viewing of the hypnosis aid from a much greater distance and by many people at the same time. Depending on the brightness and focal length of your LEDs, you may be able to project them several inches or even several feet. Try aiming your device at a light-colored wall in a dark room to see what you get. My LEDs had the ability to project their light on a wall almost 4 ft away, so my goal of a 10-in screen certainly was within reach. Figure 17-5 shows how I made a basic frame for a white-paper screen by bending a coat hanger with a pair of pliers.

To project the light evenly to the screen in front of the LEDs, I had to drill the mounting holes a bit larger and then angle them slightly in opposite directions, starting from the center two LEDs. A hot-glue gun was used to hold the LEDs in place so that they would beam outward, as shown in Figure 17-6, while projecting to the paper diffuser screen. I also mounted the cabinet on a tripod so that it could be positioned easily in a room at any height and angle for easy viewing.

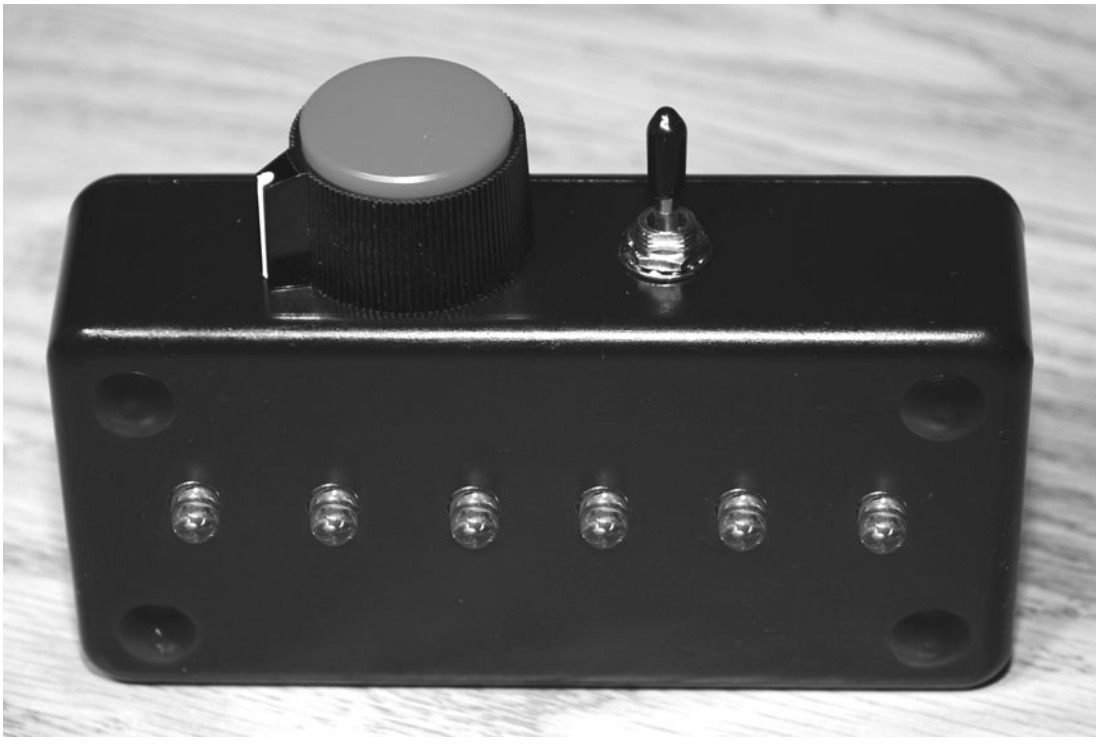


Figure 17-4 *Completed visual hypnosis aid unit.*

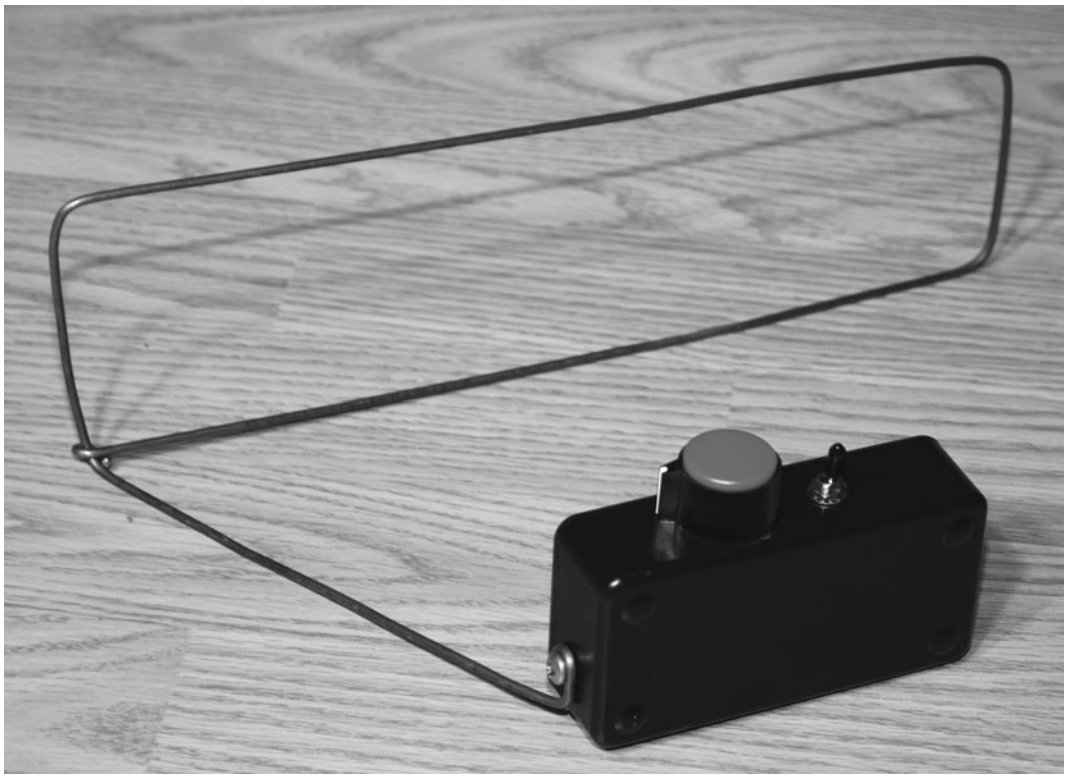


Figure 17-5 *A simple frame for a diffuser screen.*

Figure 17-6 *Diffuser screen lit by the LEDs.*

Now all you have to do is a little reading on hypnosis techniques and find someone willing to fall under your newfound powers. Hypnosis is really all about trust and belief, so don't expect much of a result from those who think they cannot by hypnotized because chances are they can't. You can always dig into self-hypnosis and never have to look for a willing participant! Some experiments you might want to try include setting the sequencer to a frequency in the alpha-wave band, changing the LED colors, or even taking a line out of one of the 4017 pins and feeding an external device to produce synchronized sound or more light. Have fun with this device, and if you manage to acquire hypnotic powers like Reveen, remember to be nice to your "zombies"!

Project Eighteen

Color-Therapy Device

Color therapy, also known as *chromatherapy* and *colorology*, is the process of balancing one's body and mind using color and light. As with many alternative medicines, color therapy is believed to be viable by some and is cast into the pit of pseudoscience by others. Color therapy has very old roots that go back thousands of years to the ancient Egyptians, who used giant solariums and colored-glass filter lenses to change the color of sunlight. On a more modern note, in the early 1920s, Swiss psychotherapist Max Luscher developed a well-known test using color that measures a person's psychophysical state based on the subject's color choices.

The power for color to evoke certain moods and responses is certainly undeniable, and advertisers have known this for decades. Have you ever noticed how many restaurant chains choose orange and yellow? How about financial institutes or other organizations that want your trust? They prefer to draw their presence in blue or magenta. There are reasons for many of the common color choices. The following is just a small sampling of colors and their accepted meanings.

Red says, "Look out!" Obviously, it screams out danger and commands attention. War, strength, love, desire, and passion are just some of the emotions that red represents. For this reason, red is the perfect color for warning signs, emergency vehicles, or anything else that needs to command attention. Did you know that people who own red sports cars pay more for their insurance? They are considered thrill seekers because of their color choices.

Orange is a happy color. Being a combination of red and yellow, it stands out, but in a way that does not say, "Look out!" Happiness, stimulation, and contentment are a few of the emotions associated with orange. Orange is the perfect color for a fast-food chain because it implies instant happiness. Orange also has been known to evoke feelings of hunger because it is also associated with citrus and good food.

Yellow is the color of the sun, so it has a warming effect and evokes feelings of happiness and joy. Energy, well-being, cheerfulness, and intellect are some concepts that work well with the color yellow.

Blue is the color of the sky and often is associated with trust, stability, loyalty, confidence, and strength. Blue is used to represent financial institutions or any organization that wants to convey its trustworthiness. Blue is also related to tranquility, peace, and serenity. Blue is also considered a masculine color, especially the darker shades. Blue also has the opposite effect that orange does when it comes to our appetites,

so there are not many fast-food chains using blue as the main color in their logos.

There are so many more examples of the link between color and emotion that they probably would double the size of this book. The real question is, Can color be used as a tool to heal the mind and change our mental state?

Using the simple device presented here, you can make your own decisions about color therapy by performing your own experimentation. Using only two LEDs and a few variable resistors, you will be able to generate any color that nature can generate in a controlled fashion.

An RGB LED is really just three separate LEDs in one small package. By placing the tiny light-emitting chips on one small base, you actually can mix the three colors together to produce any possible color because an RGB LED contains a red, green, and blue LED. By adding a variable resistor to each color, you can vary the intensity of each primary color and create millions of different colors exactly the same way your computer monitor and television can. You can use an actual RGB LED or just find a set of red, green, and blue LEDs with the same style of lens and use them to mix colors. Figure 18-1 shows some of the possibilities I found in my junk box, along with a $2 color candle that actually contained an RGB LED inside. The discount-store color candle actually was the same price as the bare RGB LED and even came with a battery, so it pays to scrounge around the dollar stores for parts once and in a while.

An RGB LED looks like any ordinary LED except that it will have four leads, not just two. There will be a common cathode or anode as well as the three separate color leads. LEDs that have three leads are only two-color LEDS. The inside of the dollar-store candle is shown in Figure 18-2, which reveals a typical clear-lens RGB LED that is often sold for more than a dollar at an electronics supply store. If you are shopping around for new RGB LEDS, try to find one with an opaque casing rather than the clear type because the clear LEDs need some kind of

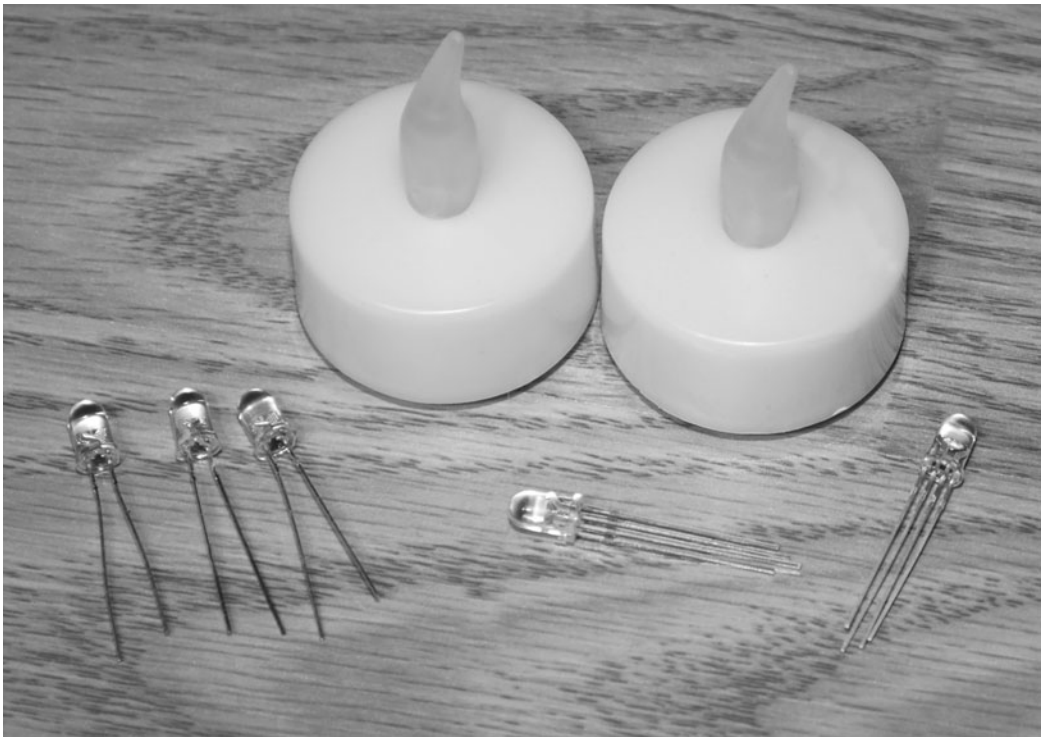


Figure 18-1 *A few different types of RGB LEDs.*

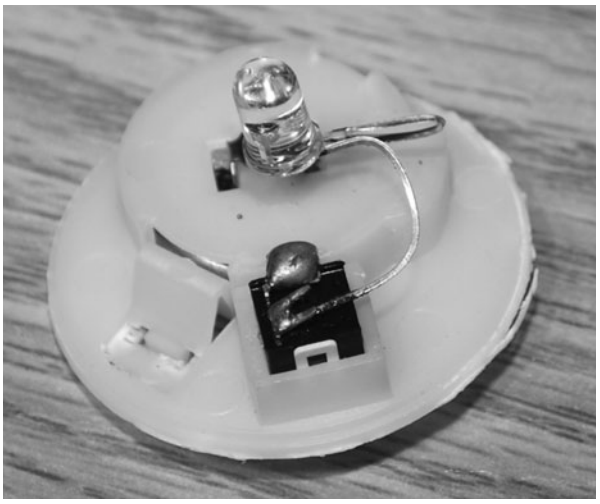


Figure 18-2 *Inside the dollar-store candle.*

diffuser to better mix the colors. Since there are actually three separate LEDs inside an RGB LED, there will be three distinct beams emitted when all the LEDs are lighted. Diffusing the three LEDs makes the color mixing look much more convincing.

Another quirky thing about RGB LEDs is that each color will have completely different current and voltage requirements. Often the red LED will have a much lower rated voltage and be made of a different material than the green and blue LEDs. Since I decided to use the new store-bought RGB LEDs (bottom right of Figure 18-1), I had a proper datasheet, which helped when choosing the current-limiting resistors. In my device, the red LED was rated for a maximum of 2.5 V, whereas the green and blue LEDs had a maximum rating of 3.5 V. Testing this with a single 1.5-V battery, it was obvious that the red LED was much too bright at the same voltage as the green and blue LEDs. For this reason, the current-limiting resistor on the red LED shown in the schematic (Figure 18-3) is three times the value of its green and blue counterparts. By tuning the current-limiting resistors to each LED, you then can control the brightness using three identical variable resistors so that they operate in a linear fashion when making up the endless color possibilities. If all variable resistors are set to the same rotation, you will have some value of white. Changing them to random positions will create every color possible.

The three current-limiting resistors shown in the schematic may not be the best values for your RGB LED, so you may need to experiment a bit to get white out of your LEDs with all variable resistors turned up full. To play it safe, start with 1K current-limiting resistors, and then look at the datasheet for your LED. Red likely will have a much lower operating voltage, so you can add resistors in series like I did until all three colors seem to be the same brightness level (Figure 18-4).

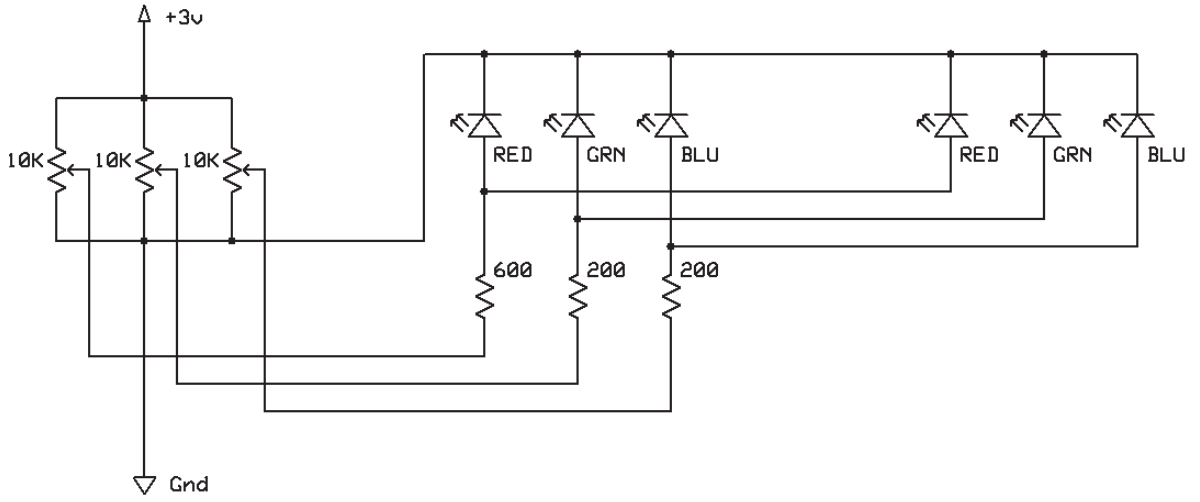


Figure 18-3 *Connecting the LEDs to a single power source.*

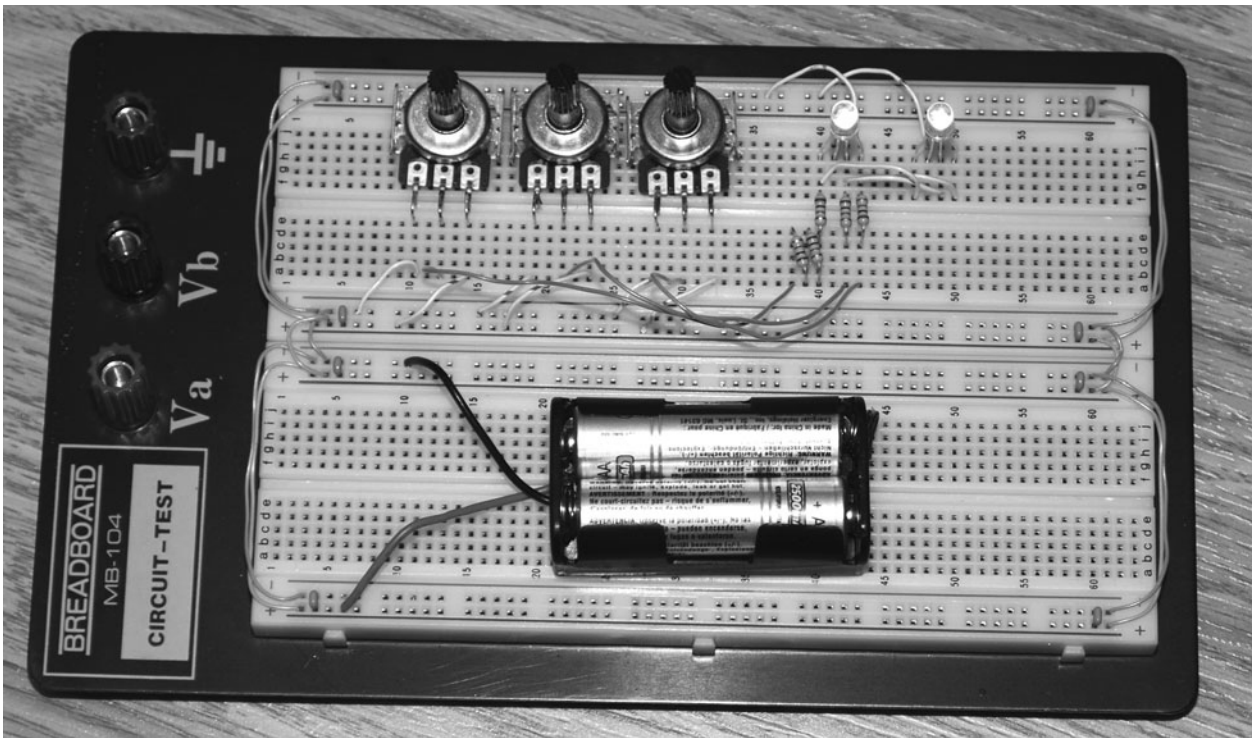


Figure 18-4 *Testing the brightness of the LEDs on a breadboard.*

Place a piece of paper over the LED to create a simple diffusion screen when checking the levels so that you can see a better color mix. If all colors are outputting the same level of brightness, you should see white.

Figure 18-5 shows the three small LEDs contained under the clear LED casing, which is why three separate beams will be emitted from the device. I am not sure why they are made like this because it really defeats the entire purpose of having an RGB LED, but there are ways we can solve the problem and get a much better color mix. A simple way to diffuse your LED is to just sand down the casing until it is so rough that you can't see through it anymore. If you are lucky enough to find an RGB LED in a milky white casing, then you will not have this problem, although these seem hard to find for some strange reason. The good news is that you can make a perfect diffuser for your eyes using some Ping-Pong balls, as was done earlier in this section.

By placing half of a Ping-Pong ball or a piece of white Styrofoam over the LED, you can really diffuse the three beams a lot better. Without the diffusing screen, it is almost impossible to see that the colors are mixing, but by placing the Ping-Pong ball half over the LED (Figure 18-6), a much better mixing of colors can be had. When the camera took the photograph, it was still easy to see the three separate beams, although the color mixing was quite good when viewed by eye. For even more diffusion, some white foam or paper slices can be placed inside the Ping-Pong ball half to further scatter the light, although this will cost you in brightness.

If you plan to deliver the LED light to your eyes using a mask or some type of goggles, then a perfect diffuser tube can be made out of a pair of film canisters and a pair of cut-in-half Ping-Pong balls. Place the Ping-Pong ball into the open end of the film canister, as shown in Figure 18-7, so that the nonlogo side will be used. Trace around the top of the canister so that you can cut out a segment of the ball that will fit perfectly back into

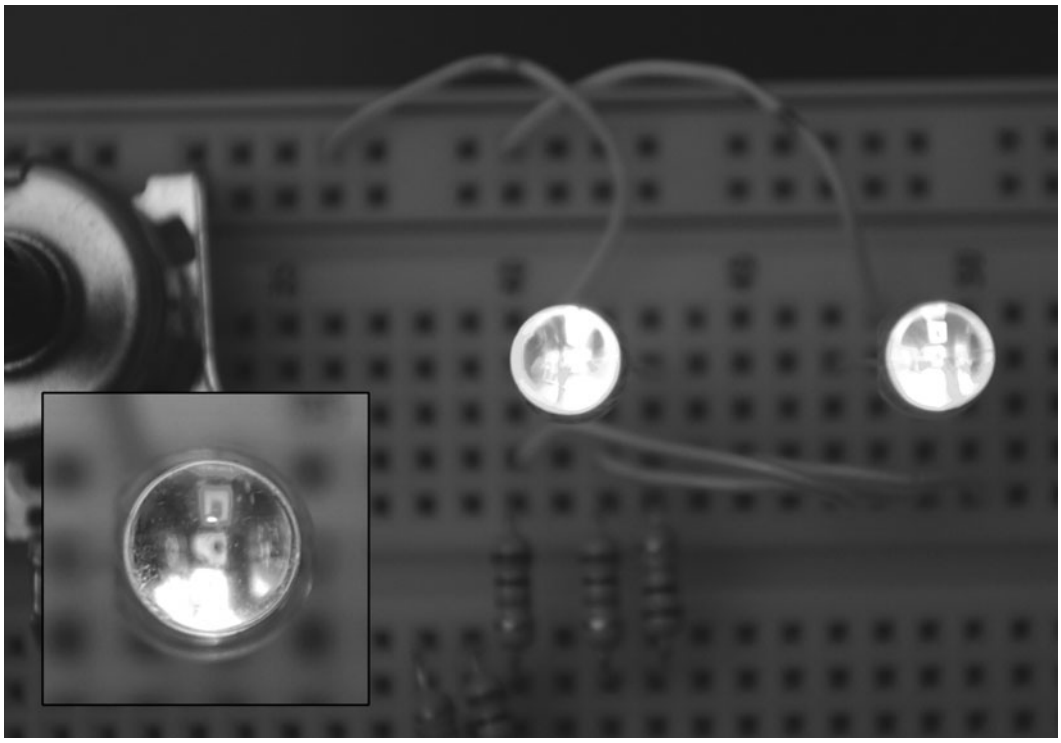


Figure 18-5 *You can see the three small LEDs inside.*

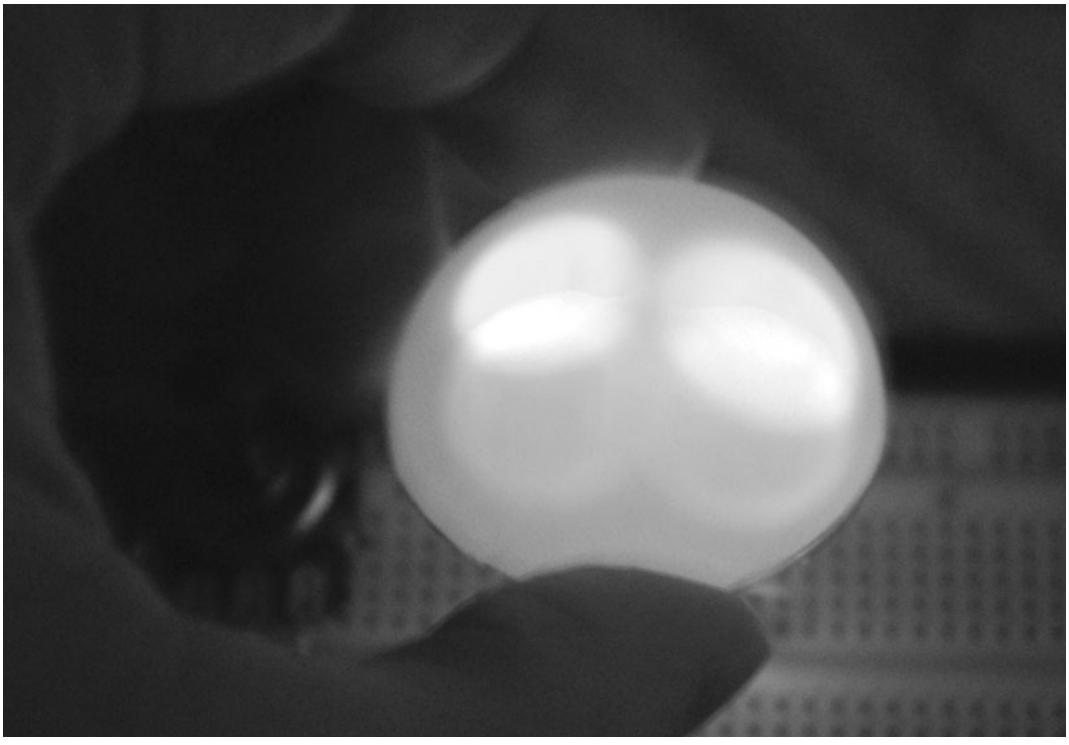


Figure 18-6 *Better diffusion of the three colored beams.*

Figure 18-7 *Making a diffuser tube for the goggles.*

the canister. A pair of scissors with a sharp tip is best for cutting the balls to avoid cracking the brittle shell as you cut along your line.

The two Ping-Pong ball segments are affixed to the open ends of the film canisters by hot gluing them around the edges, as shown in Figure 18-8. If you find that your LED works best with a bit of white Styrofoam in front of the ball half, then don't forget to stuff it into the canister before you glue in the ball segments.

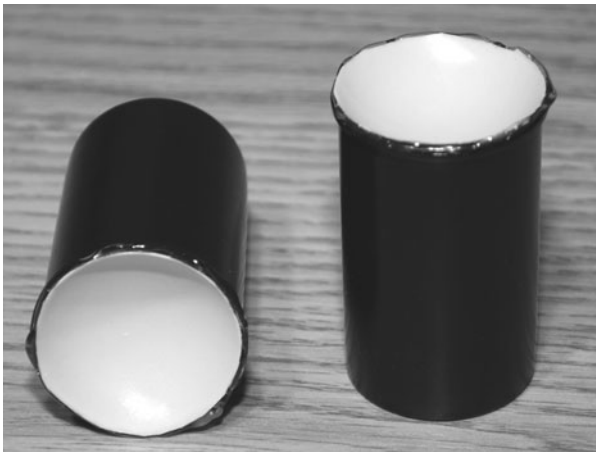


Figure 18-8 *Gluing the ball halves to the canisters.*

Since you won't be wearing the goggles for extended periods, a pair of inexpensive swim goggles will work just fine for this project once you loosen the straps so that they are not too tight. If you can't find clear plastic goggles, then you will have to use a Dremel tool or soldering iron to cut or melt out the front lens so that the color of the light is not affected. My lenses were clear and I also painted the inside using a black marker to block out any ambient light that may enter from the sides. Figure 18-9 shows the diffuser tubes glued to the front of the goggles, as well as the RGB LED stuck into a hole at the other end. This configuration worked very well because the length of the film canister was almost perfect for the focal length of the LEDs. The three beams came very close together over the Ping-Pong ball surface, which made for a good mixing of colors.

The RGB LEDs also were hot glued to the film canisters once the leads were trimmed and the wires installed. If you plan to use the goggles on another person or with some other type of LED driver, then you also might consider using a removable plug to connect the goggles to the control box. I just used a 6-ft length of four-

Figure 18-9 *Diffuser tubes glued to the goggles.*

conductor wire to connect between the box and the goggles because I was only using the color-therapy device on myself. Eventually, you run out of victims, I mean subjects, as your friends get to know your strange "mad scientist" side!

The control box is just your three variable resistors, current-limiting resistors, and battery pack placed in a box, as shown in Figure 18-11. Although you probably don't have to label the controls, it does help when you are not the subject and are experimenting with your subject's

Figure 18-10 *The completed color-therapy goggles.*

Figure 18-11 *The completed color-therapy control box.*

reactions to the various colors. Since the circuit will slowly drain the batteries even if the LEDs are turned right down, an on/off switch will be necessary. With a pair of AA batteries, this device will run for many days because the LEDs use very little current. Another modification you might want to consider is an input on the side of the control box to connect some external device to the LEDs. Some of the devices presented earlier, such as the alpha meditation goggles and the visual hypnosis aid, could be modified to drive the color-therapy goggles with little work.

Operating the color-therapy device is easy. Simply strap the goggles to your head (Figure 18-12), and play with the dials until you feel great. You need to have your eyes open for the color device to work because your eyelids are red and will only pass red light. I actually found that a few minutes of bright yellow each day did wonders to help beat those winter blues I often get when it's –40°F outside and dark before suppertime. Even if you don't believe in the whole balanced mind and body aspect of chromatherapy, there is no doubt that colored light will affect your moods, and this device has some real merits. I tried the orange-light experiment to see if it would affect my hunger, but I cannot honestly say if it worked because the thought of eating was already in my mind when I dialed up the orange color.

For a nonbiased experiment on the various colors and their known responses, try the device on a willing subject who has no clue about which colors are for which emotions. If you search for "color therapy" or "chromatherapy" on the Internet, you will find many sites that link a multitude of colors to various emotional responses. Have fun, and enjoy the rainbow!

Figure 18-12 *Ah, yellow . . . a cure for the winter blues!*

Project Nineteen

Synchro Brain Machine

This interesting device is based on research done by Professor Heinrich Wilhelm Dove in 1839. Dove discovered an effect called *binaural beats* in which the brain will hear a "beat" when presented with two slightly different independent audio frequencies into each ear. This effect is similar to what is heard when one tunes up an instrument while using another sound source as a reference. That unmistakable "wong, wong, wong, wong" sound as a guitar player tunes up a string is another example of the "beat" that can be heard as two sound waves go in and out of phase with each other. In the brain, something more interesting happens if we are fed the two slightly differing frequencies into each ear. It is called *brain-wave entrainment*.

Brain-wave entrainment also was the focus of a previous project in this chapter, the alpha meditation goggles, although it is said that by using binaural beats, the brain entrainment effect is much stronger. The general theory is that the resulting frequency of the binaural beat will coax the two brain hemispheres to become synchronized over time. By setting the oscillators so that their phase difference is somewhere between 10 and 15 Hz, your brain will subconsciously hear the alpha frequency and eventually fall into sync with it. The actual frequency of the oscillators is much higher than the resulting binaural beat, often in the range of hundreds of hertz.

For example, if you wanted to entrain your brain to the frequency of 10 Hz, you could set the right-ear frequency to 315 Hz and the left-ear frequency to 325 Hz, resulting in a binaural-beat frequency of 10 Hz. The various frequencies of the brain are named delta, theta, alpha, beta, and gamma. They have the frequencies and effects on consciousness as follows.

Delta waves are frequencies less than 4 Hz. This means that there are only four cycles per second, so this is a very low frequency. When the mind is in the deepest stage of sleep, where there are no dreams and the body is fully paralyzed, delta waves are prominent. It is probably not much use trying to entrain your brain to this frequency because you would have to traverse the other states of consciousness to reach this level.

Theta waves occur between 4 and 7 Hz, overlapping into the lower end of alpha waves. When your brain is running at the theta frequency, you are likely either dreaming or in a state of very deep mediation. If you entrain your brain to the theta state, the result likely will be that you fall asleep.

Alpha waves fall between 7 and 15 Hz. Entraining the brain to the alpha frequency is

usually the goal of entrainment because it provides a person with pure relaxation without falling asleep. At the lower end of the alpha scale, a person is just starting to cross the line between deep relaxation and sleep, so this is an interesting frequency to experiment with. At the higher end of the alpha range, a person is fully conscious but feeling very relaxed, so reaching this state a few times a day can be a real benefit to the mind.

Beta waves occur between 15 and 40 Hz. This is the state of mind we are in while working or thinking. Your brain probably is working on some part of the beta frequency range right now as you read this book, and it is the most common state of mind during the day. Trying to entrain the brain to this frequency might not be much use because you are probably already working on beta waves if you are getting ideas about doing these experiments!

Gamma waves are the frequencies that occur above 40 Hz. If your brain is in the gamma state, then you are really working your gray matter because this state of mind includes fear, difficult problem solving, anxiety, and many other higher mental activities. You might want to entrain your brain to the gamma frequency when trying to work out a very complex problem or right before some type of test that will push your mental abilities to their limits. The gamma state is certainly not a relaxing state, but it does have its uses.

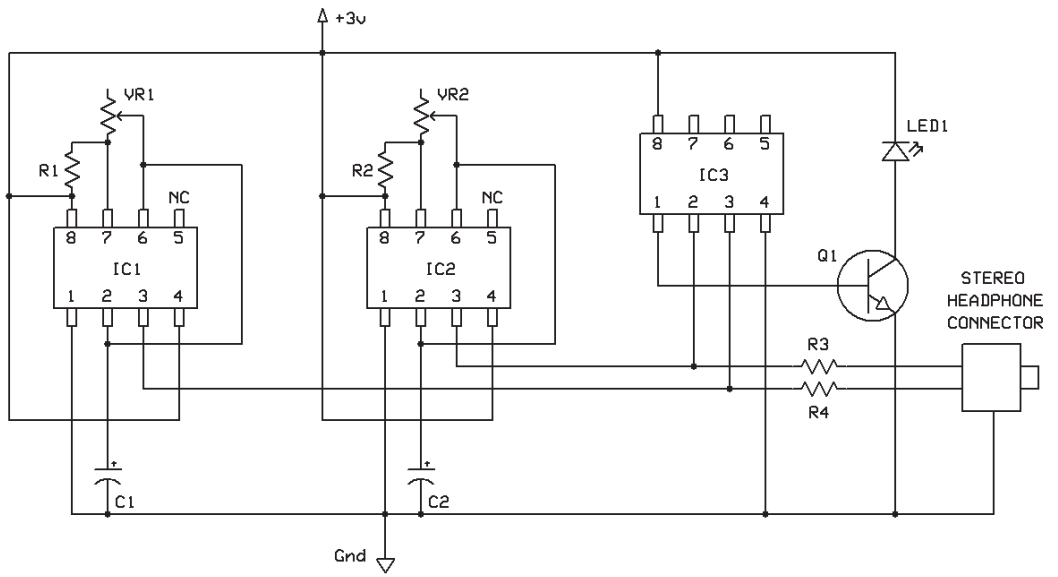
One extremely interesting and rare state of mind is called *hypnagogic sleep paralysis*, which is an almost accidental state of sleep and consciousness at the same time. If you have ever been through a sleep-paralysis episode, then you know what I mean when I say the experience can be terrifying, especially if you have no idea what is going on. You open your eyes, but your body fails to respond, although you have complete control over your eyes. Many people experience sleep paralysis a few times during their lives, often just after falling asleep during the day from a needed nap. If you look up "sleep paralysis" on the Internet, you will see that it is often related to what people think are out-of-body experiences (OBEs), demonic possessions, and many other psychic phenomena.

I have had several controlled sleep-paralysis episodes now and find the state of mind to be extremely interesting, almost like the "extreme sport" version of sleep research! If you know what to expect, the hallucinations and pure weightlessness feeling can be quite a wild ride. I found that using the synchro brain machine or even the alpha meditation goggles set to the 6-Hz range while trying to sleep during a quiet afternoon was a good way to trigger a sleep-paralysis episode. Having no caffeine that day or other stimulants and being a bit on the tired side also were factors that improved the chances of reaching this rare and almost mystical state. Find a quiet, comfortable place to lie on your back, and just try to clear your mind while using the brain entrainment system. I must warn you, though, before you consider trying to purposely put yourself in a hypnagogic state, read as much as you can on sleep paralysis and lucid dreaming so that you know what you are about to encounter. Going through a session of sleep paralysis may be the strangest, scariest, and most bizarre state of mind you have ever encountered, so be prepared! Okay, enough talk—let's solder some wires now!

The synchro brain machine is a pair of independently tunable audio oscillators that feed their outputs into the right and left sides of a stereo headphone worn by the user. There are a few examples of commercial units available that have many nice features included, such as a digital frequency readout, multiple waveforms, and even the ability to add white noise or mix the binaural beat frequencies with some type of external audio source. This extremely "bare bones" version of the device actually will do most of this as well, although you will need to use your multimeter to see the digital frequency display and some external audio source such as an MP3 player to mix in your music or meditation media.

Figure 19-1 shows the synchro brain machine schematic, and the two oscillators made from the two 555 timers (IC1 and IC2) can be seen clearly. The third IC (IC3) is an LM358 op amp that is working as a simple comparator that will flash an LED each time the two audio frequencies cross each other or become synchronous for a short period. The resulting LED flash rate will be the same binaural beat that your brain will hear when presented with the two different audio frequencies. Each oscillator output is fed into the right and left sides of a stereo headphone jack and to an optional jack that will allow you to monitor the frequency of each oscillator channel. To change the two oscillator frequencies, you just turn the variable resistors (VR1 and VR2) to some frequency that is comfortable to listen to. It is the resulting binaural-beat frequency that counts, not the initial oscillator frequencies.

The breadboarded synchro brain machine circuit is shown in Figure 19-2, running on a pair of AA batteries. By using AA or even AAA batteries, you could build the entire unit into a headphone casing or some very small box that could be affixed directly to the headphones, making a very portable device. I preferred the external box version so that I could have the frequency adjustment controls near my hand while relaxed. When you first power up the unit, the indicator LED probably will be steady until you tune both variable resistors to the exact same spot. When your oscillators are only slightly out of phase, you will see the LED flash very slowly or several times per second. Listening to the output and tuning the controls as if you were tuning a guitar is another way to get the oscillators close to phase. As soon as you have them outputting similar frequencies, you will hear



PARTS LIST FOR THE SYNCHRO BRAIN MACHINE
IC1,IC2 = NE555 TIMER
IC2 = LM358 OR SIMILAR OP AMP
Q1 = 2N3904 OR SIMILAR NPN TRANSISTOR
R1,R2 = 1K
R3,R4 = 10K
C1,C2 = 1UF
VR1, VR2 = 10K VARIABLE RESISTOR
LED1 = 3 VOLT RED LED

Figure 19-1 *The synchro brain machine schematic.*

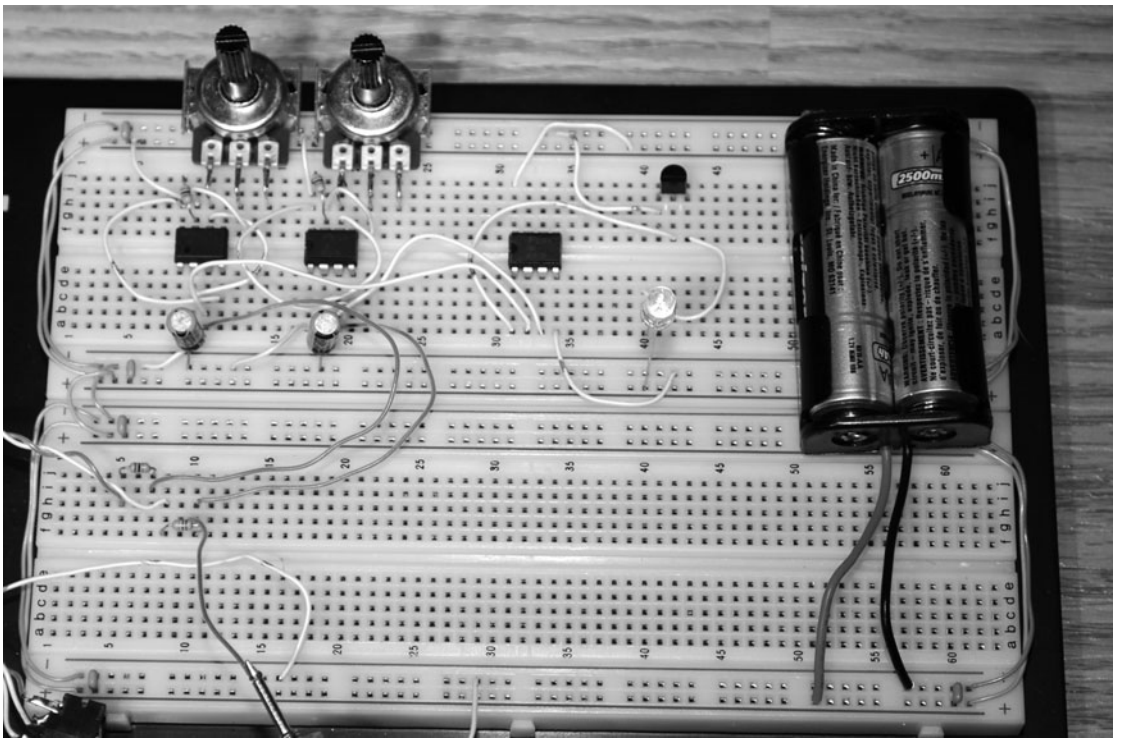


Figure 19-2 *The synchro brain machine circuit.*

that "wong, wong, wong, wong" sound that I was referring to earlier. This is the binaural beat.

From my research into binaural beats, it seems that frequencies between 200 and 400 Hz are used most commonly because they are easy to listen to compared with the very high or low frequencies. There is plenty of range in the oscillators to go well beyond or below 300 Hz, so you can experiment with the frequencies you are most comfortable with. Figure 19-3 shows the readout on my basic multimeter, indicating that the left channel is set to 316 Hz. I have the right channel set to 306 Hz, which places the binaural-beat frequency at 10 Hz, right into the alpha-wave range.

If you built the circuit on a small enough piece of perf board and use a pair of AAA batteries or even a camera battery, you might be able to cram the works inside a large set of headphones, making a nice all-in-one device. I wanted to add another stereo jack to the unit so that I could easily attach my frequency counter or input an external audio source into the sound stream. Thus I opted for the conventional black-box enclosure shown in Figure 19-4. Having the remotely located frequency control also was convenient when I was relaxing so I did not have to move my arms to adjust the controls. I became so used to the alpha frequencies I so often used that I could set them without actually needing the digital readout. This is the same skill a musician uses when picking out a certain note or key.

Figure 19-5 shows my completed synchro brain machine—the true source of my magical powers and psychic abilities! If you find the output of the device to be too loud or too quiet, then change the values of R3 and R4 or replace them with a pair of 1K variable resistors so that you can set the volume to a level that you find most comfortable. If you plan to use the same basic frequencies most of the time, then you even could replace the frequency-control variable resistors with smaller potentiometers or have multiple preset banks of potentiometers and a select switch to instantly choose a "program" that you have preset. I preferred to simply listen to the

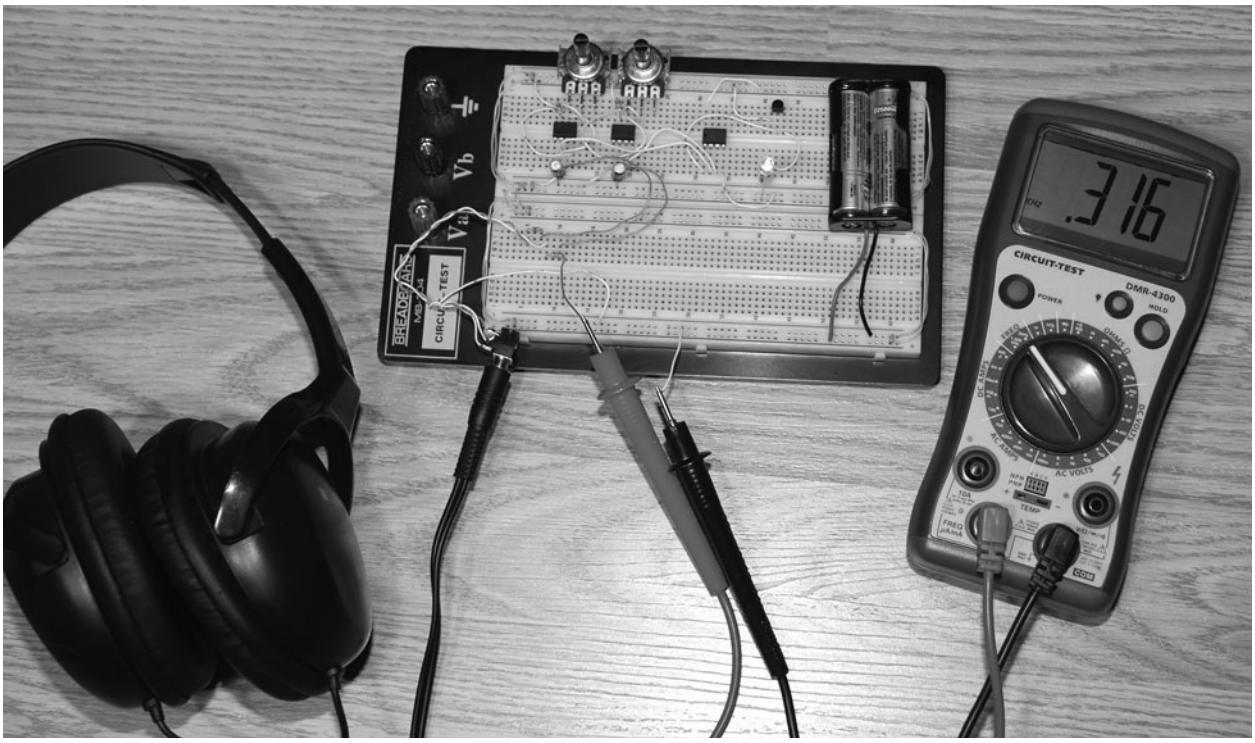


Figure 19-3 *Displaying the frequency on a multimeter.*

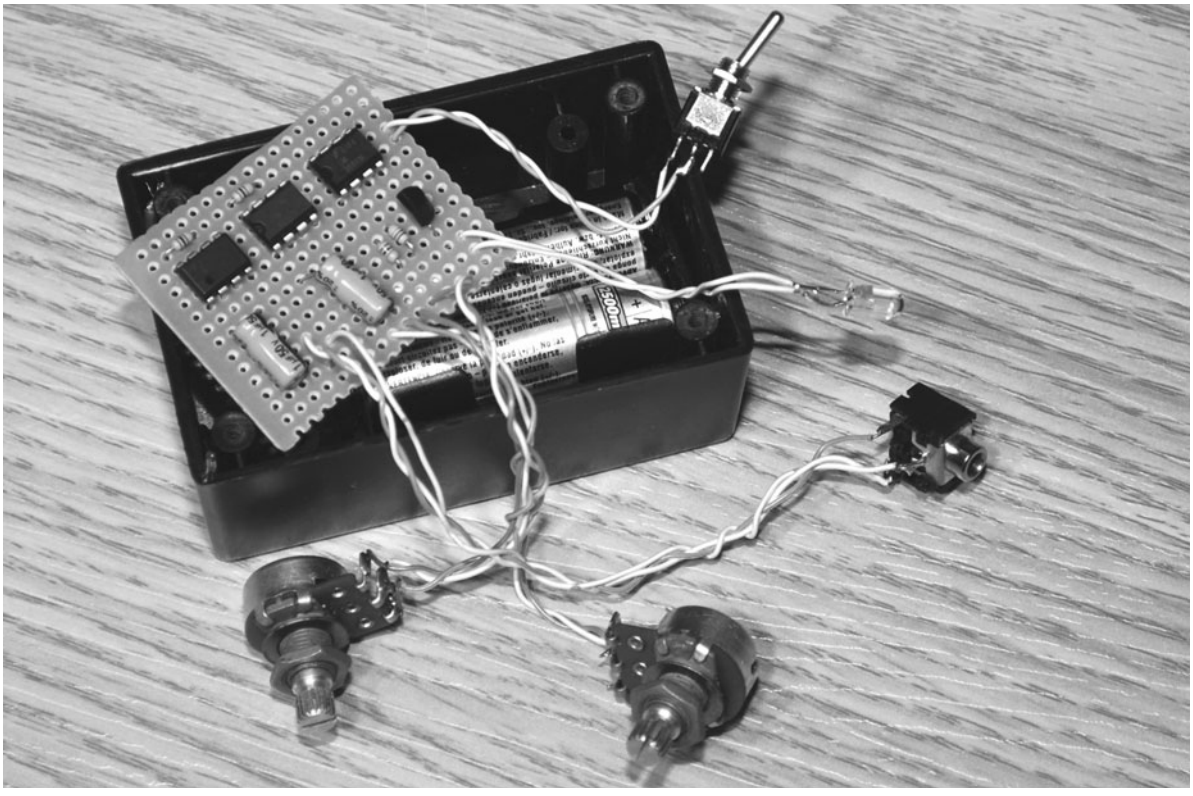


Figure 19-4 *Boxing up the synchro brain machine circuit.*

Figure 19-5 *The completed synchro brain machine.*

output and set the controls to a base frequency and beat frequency that felt right at the time. Some frequencies will just seem to feel right after you work with the device for some time. Maybe the brain learns to crave certain states much the way your mind craves certain tastes when the body requires refueling.

Before you start experimenting with this device, do a little Internet research on "binaural beats" and "brain entrainment" so that you will understand the way the brain responds to this technology and how you can use it to your benefit. As with all things in the public domain, you will have to toss away the far ends of the debate and look to the middle to learn anything useful. On one end of the scale, I have read that brain entrainment is the path to complete enlightenment and psychic powers, and on the end, I have read reports of brain entrainment being the most dangerous thing on earth. As a true scientist, I will leave you to make your own decisions based on actual research!

Some interesting modifications you might want to consider include the addition of a light stimulus such as the goggles presented in the alpha meditation section or the color-therapy device. How about adding white noise or mixing the binaural beats with some meditation or lucid-dream training media? There are so many ways you can enhance this unit, so enjoy the journey, but be prepared for the unexpected!

Conclusion

The electronics hobby is both entertaining and mind expanding, much like many of the devices presented in this book. With only a handful of inexpensive semiconductors and a breadboard, you can hack together just about any type of device in your "evil genius" workshop in a few evenings. Sure, there will be "bugs" and even the odd puff of smoke as you try out new designs, but as with all things, practice and patience will bring results.

Many of the projects presented here can be mixed and matched together to create entirely new devices, and there is always room for modifications and improvements, so feel free to let out your "crazy ideas" and try out new designs. When you want to further research a certain aspect of biofeedback or electronic design, the Internet is the ultimate source for free and comprehensive information, and there are many forums and user groups that you can become a part of.

When "plugging" yourself into any home-built device, remember to consider the safety aspects of your design, especially if there is a connection between the hardware and some other appliance, such as a computer or an external power supply. It is also a good idea to have a clean bill of health from your doctor before subjecting yourself to any type of sound- or light-pulsing device to reduce the risk of accidentally inducing health issues such as an epileptic seizure. Although the risk of harm is extremely remote when messing around with low-power projects running from a single battery, *safety first* is always a good rule to follow.

Have fun, and don't be afraid to take your ideas from the drawing board to the circuit board. You just never know what you might invent by thinking outside the box!