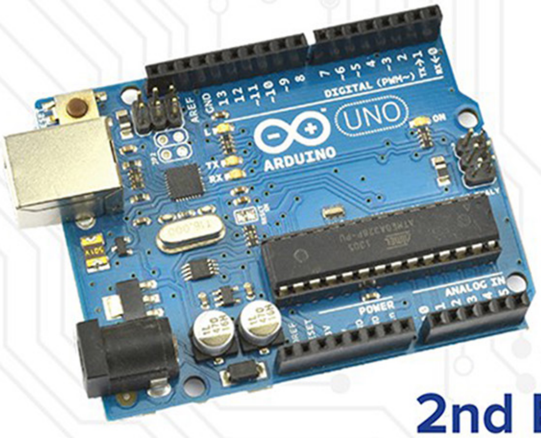
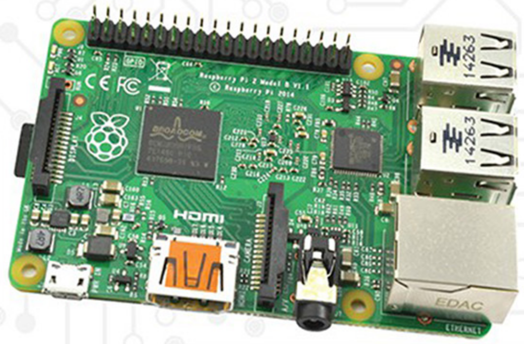
McGraw Hill Education

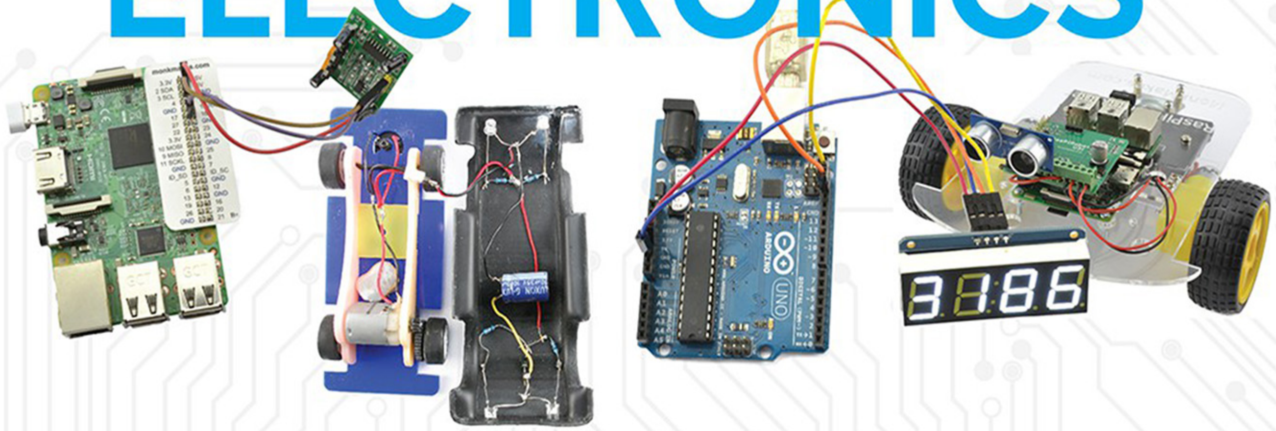
TAB Make Great Stuff!





2nd Edition

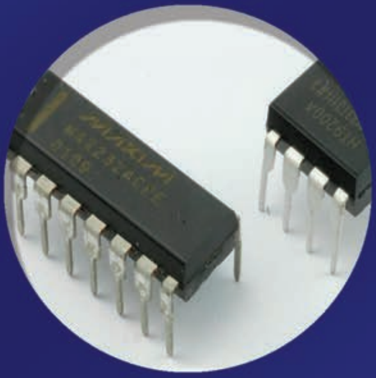
HACKING ELECTRONICS



Learning Electronics with
Arduino and Raspberry Pi

SIMON MONK

HACKING ELECTRONICS



About the Author

Dr. Simon Monk (Preston, UK) has a degree in Cybernetics and Computer Science and a PhD in Software Engineering. Monk spent several years as an academic before he returned to industry, co-founding the mobile software company Momote Ltd. He has been an active electronics hobbyist since his early teens and is a full-time writer on hobby electronics and open-source hardware. Dr. Monk is the author of numerous electronics books, specializing in open-source hardware platforms, especially Arduino and Raspberry Pi. He is also co-author with Paul Scherz of *Practical Electronics for Inventors, Fourth Edition*. You can follow Simon on Twitter, where he is @simonmonk2.

Hacking Electronics

Learning Electronics with Arduino® and Raspberry Pi

Second Edition

Simon Monk

Mc Graw Hill Education

New York Chicago San Francisco Lisbon
London Madrid Mexico City Milan New Delhi
San Juan Seoul Singapore Sydney Toronto

ISBN: 978-1-26-001221-7
MHID: 1-26-001221-2.

The material in this eBook also appears in the print version of this title: ISBN: 978-1-26-001220-0,
MHID: 1-26-001220-4.

eBook conversion by codeMantra
Version 1.0

To Roger, for making it possible for me to turn a hobby into an occupation.

This page intentionally left blank

Contents at a Glance

This page intentionally left blank

Contents

This page intentionally left blank

Acknowledgments

Many thanks to all those at McGraw-Hill Education who have done such a great job in producing this book. In particular, thanks to my editor Michael McCabe, Donna Martone, Lynn Messina, Patricia Wallenburg, and Claire Splan.

And last but not least, thanks once again to Linda, for her patience and generosity in giving me space to do this.

This page intentionally left blank

Introduction

This is a book about "hacking" electronics. It is not a formal, theory-based book about electronics. Its sole aim is to equip the reader with the skills he or she needs to use electronics to make something, whether it's starting from scratch, connecting together modules, or adapting existing electronic devices for some new use.

You will learn how to experiment and get your ideas into some kind of order, so that what you make will work. Along the way, you'll gain an appreciation for why things work and the limits of what they can do, and learn how to make prototypes on solderless breadboard, how to solder components directly to each other, and how to use protoboard to make more complex soldered circuits.

You will also learn how to use the popular Arduino microcontroller board, which has become one of the most important tools available to the electronics hacker. There are over 20 examples of how to use an Arduino with electronics in this book.

You will also learn how to use the Raspberry Pi (a tiny Linux computer) as a tool for electronics hacking.

Electronics has changed. This is a modern book that avoids theory you will likely never use and instead concentrates on how you can build things using readymade modules when they are available. There is, after all, no point in reinventing the wheel.

Some of the things explained and described in the book include

- Using LEDs, including high-power Lumileds and Addressable LED strips (Neopixels)
- Using LiPo battery packs and buck-boost power supply modules
- Using sensors to measure light, temperature, acceleration, sound level, and color
- Interfacing the Raspberry Pi and Arduino with external electronics
- Using servo motors

Some of the things described in the book that you can make along the way include

- A smartcard RFID tag reader
- An Internet-controlled hacked electric toy
- A device for measuring color
- An ultrasonic rangefinder
- A remote control robotic rover
- An accelerometer-based version of the "egg and spoon" race
- An audio amplifier
- A bug made from a hacked MP3 FM transmitter
- Working brakes and head lights that can be added to a slot car
- A smart-card reader/spoofer

You Will Need

This is a very practical, hands-on type of book. You will therefore need some tools and components to get the most out of it.

As far as tools go, you will need little more than a multimeter and soldering equipment.

You should also have a Raspberry Pi, or Arduino or both, as quite a few of the projects use these handy boards.

Every component used in this book is listed in the Appendix, along with sources where it can be obtained. The majority of the components can be found in a starter kit from SparkFun, but most electronic starter kits will provide a lot of what you will need.

In many of the "how-tos," there will be a You Will Need section. This will refer to a code in the Appendix that explains where to get the component.

How to Use This Book

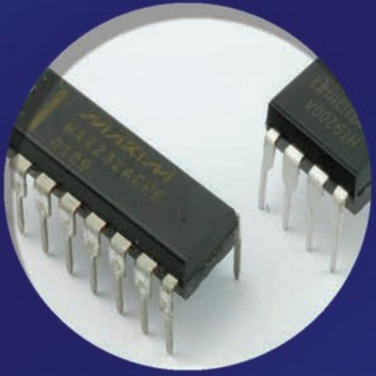
The book contains the following chapters:

Chapter	Title	Description
Chapter 1	Getting Started	The book starts off by telling you where you can buy equipment and components, as well as things to hack. This chapter also deals with the basics of soldering and focuses on a project to hack an old computer fan to make a fume extractor for use while soldering.
Chapter 2	Components	This chapter introduces electronic components—or at least the ones you are likely to use—and explains how to identify them and describes what they do. It also introduces a small amount of essential theory, which you will use over and over again.
Chapter 3	Basic Hacks	This chapter contains a set of fairly basic "hacking" how-tos, introducing concepts like using transistors with example projects. It includes hacking a "push light" to make it automatically turn on when it gets dark and how to control a motor using power MOSFETs.
Chapter 4	LEDs	In addition to discussing regular LEDs and how to use them and make them flash and so on, this chapter also looks at using constant current drivers for LEDs and how to power large numbers of LEDs and laser diode modules.
Chapter 5	Batteries and Power	This chapter discusses the various types of battery, both single use and rechargeable. It also covers how to charge batteries including LiPos. Automatic battery backup, voltage regulation, and solar charging are also explained.
Chapter 6	Hacking Arduino	The Arduino has become the microcontroller board of choice for electronics hackers. Its open-source hardware design makes using a complex device like a microcontroller very straightforward. The chapter gets you started with the Arduino and includes a few simple how-tos, like controlling a relay, playing sounds, and controlling servo motors from an Arduino. It also covers the use of Arduino expansion shields.
Chapter 7	Hacking with Raspberry Pi	The Raspberry Pi single board computer is great for hacking together electronic projects that require a bit more power than an Arduino can provide, or that need a network connection or large display. In this chapter you will learn how to set up and use a Raspberry Pi, as well as connect electronics to its GPIO pins.

Chapter	Title	Description
Chapter 8	Hacking with Modules	When you want to make something, you can often use readymade modules at least for part of the project. Modules exist for all sorts of things, from wireless remotes to motor drivers.
Chapter 9	Hacking with Sensors	Sensor ICs and modules are available for sensing everything from temperature to acceleration. In this chapter, we explore a good range of them and explain how to use them and connect some of them to an Arduino.
Chapter 10	Audio Hacks	This chapter has a number of useful how-tos relating to electronics and sound. It includes making and adapting audio leads, as well as audio amplifiers, and discusses the use of microphones.
Chapter 11	Mending and Breaking Electronics	Mending electronics and scavenging useful parts from dead electronics are a worthy activity for the electronics hacker. This chapter explains how to take things apart and sometimes put them back together again.
Chapter 12	Tools	The final chapter of the book is intended as a reference to explain more about how to get the most out of tools such as multimeters and lab power supplies.

This page intentionally left blank

HACKING ELECTRONICS



1

Getting Started

In this first chapter, we will investigate some of the tools and techniques needed to hack electronics. We will start with a little soldering, and wire up an old computer fan to help keep the solder fumes out of our lungs.

As it says in the title, this book is all about "hacking electronics." The word "hacking" has come to mean many things. But in this book, "hacking" means "just do it!" You don't need a degree in electronic engineering to create or modify something electronic. The best way to learn is by having a go at it. You will learn as much from your mistakes as from your successes.

As you start to make things and experiment, you will likely want to understand more of the theory behind it all. Traditional electronics textbooks are pretty terrifying unless you have a good grasp of complex mathematics. This book strives to, above all else, enable you to do things first and worry about the theory later.

To get started, you will need some tools, and also find out where to get components and parts to use in your projects.

Getting Stuff

In addition to buying components and tools, there are lots of low-cost and interesting electronic consumer items that can be hacked and used for new purposes, or that can act as donors of interesting components.

Buying Components

Most component purchases happen on the Internet, although there are local electronic stores like Micro Center and Fry's (in the U.S.) and Maplin (in the UK) where you can buy components. At traditional brick-and-mortar stores like those, the product range is often limited and the prices can be on the high side. They do, after all, have a shop to pay for. These stores are invaluable, however, on the odd occasion when you need something in a hurry. Perhaps you need an LED because you accidentally destroyed one, or maybe you want to look at the enclosures they sell for projects. Sometimes it's just nice to hold a box or look at tools for real, rather than trying to size them up from pictures on a web site.

As you get into electronics, you will likely gradually accumulate a set of components and tools that you can draw from when you start a new project. Components are relatively cheap, so when I need one of something, I generally order two or three or even five if they are cheap, enough that I have extras that can be used another time. This way, you will often find that when you start to work on something, you actually have pretty much everything you need already.

Component buying really depends on where you are in the world. In the U.S., Mouser and DigiKey are the largest suppliers of electronic components to the hobby electronics market. In fact, both of these suppliers sell worldwide. Farnell also supplies pretty much anything you could want, anywhere in the world.

When it comes to buying ready-made electronics modules for your projects, the SparkFun, Seeed Studio, Adafruit, and ITead Studio web sites can help. All have a wide range of modules, and much enjoyment can be had simply from browsing their online catalogs.

Nearly all the components used in this book have part codes for one or more of the suppliers I just mentioned. The only exceptions are for a few unusual modules that are better to buy from eBay.

There is also no end to the electronic components available on online auction sites, many coming direct from countries in the far east and often at extremely low prices. This is frequently the place to go for unusual components and things like laser modules and high-power LEDs that can be expensive in regular component suppliers. They are also very good for buying components in bulk. Sometimes these components are not grade A, however, so read the descriptions carefully and don't be disappointed if some of the items in the batch are dead-on-arrival.

Finally, a kit designed specifically for this book and designed by the author is available from MonkMakes Ltd. (https://monkmakes.com/hacking2).

Where to Buy Things to Hack

The first thing to consider, now that you are into hacking electronics, is an effect that your household and friends will have on you. You will become the recipient of dead electronics. But keep an eye open in your new role as refuse collector.

Sometimes these “dead” items may actually be candidates for straightforward resurrection.

Another major source of useful bits is the dollar store. Find the aisle with the electronic stuff: flashlights, fans, solar toys, illuminated cooling laptop bases, and so on. It’s amazing what can be bought for a single unit of currency. Often you will find motors and arrays of LEDs for a lower price than you would the raw components from a conventional supplier.

Supermarkets are another source of cheap electronics that can be hacked. Good examples of useful gadgets are cheap powered computer speakers, mice, power supplies, radio receivers, LED flashlights, and computer keyboards.

A Basic Toolkit

Don’t think you are going to get through this chapter without doing some soldering. Given this, you will need some basic tools. These do not have to be expensive. In fact, when you are starting out on something new, it’s a good idea to learn to use things that are inexpensive, so it doesn’t matter if you spoil them. After all, you wouldn’t learn the violin on a Stradivarius. Plus, what will you have to look forward to if you buy all your high-end tools now!

Many starter toolkits are available. For our purposes, you will need a basic soldering iron, solder, a soldering iron stand, some pliers, snips, and a screwdriver or two. SparkFun sells just such a kit (SKU TOL-09465), so buy that one or look for something similar.

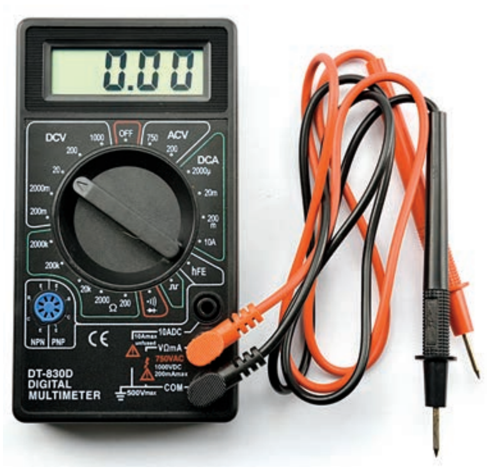


FIGURE 1-1 A digital multimeter

You will also need a multimeter (Figure 1-1). I would suggest a low-cost digital multimeter (don’t even think of going above USD 20). Even if you end up buying a better one, you will still end up using the other one since it’s often useful to measure more than one thing at a time. The key things you need are DC Volts, DC current, resistance, and a continuity test. Everything else is fluff that you will only need once in a blue moon. Again, look for something similar to the model shown in Figure 1-1. A multimeter like this is supplied with the MonkMakes Hacking Electronics Kit.

Test leads that end in alligator clips rather than a probe are very useful; some multimeters are supplied with such leads. If your multimeter does not include aligator-clip test leads, these are available from eBay for a few dollars.

Solderless breadboards (Figure 1-2) are very useful for quickly trying out designs before you commit them to solder. You poke the leads of components into the sockets, and metal clips behind the holes connect all the holes on a row together. They are not expensive (see T5 in the Appendix).

You will also need some solid core wire in different colors (T6) to make bridging connections on the breadboard. Another good idea is to buy special-purpose jumper wires with little plugs on the end—although these are useful, they are by no means essential.

Breadboard come in all shapes and sizes, but the most popular and the one used in this book is called “half-breadboard” or 400 point breadboard. This has 30 rows in two columns with two “supply” strips down each side (Figure 1-1). This kind of breadboard is widely available to buy and is also included in the MonkMakes Hacking Electronics Kit.

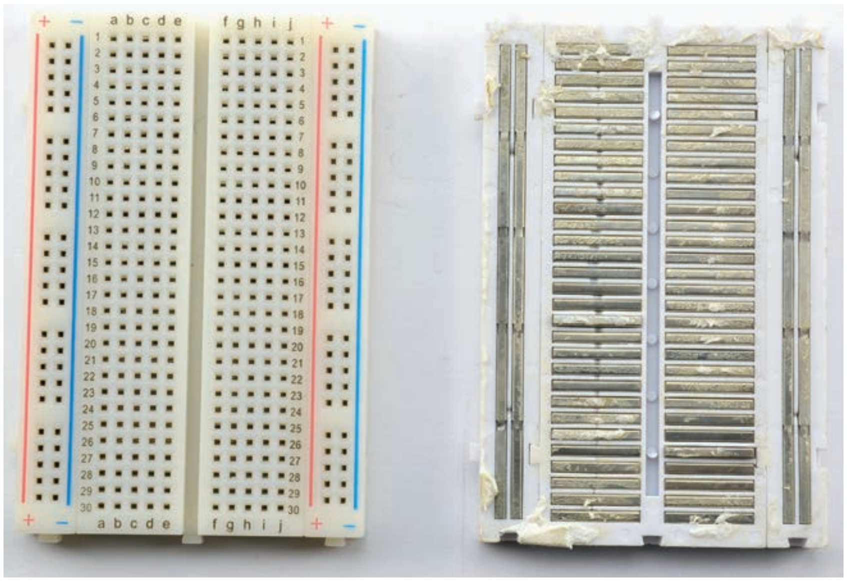


FIGURE 1-2 Solderless breadboard

Figure 1-2b shows a breadboard that has been disassembled so that you can see the metal conductive clips behind the plastic. The long strips down each side are used for the power supply to the components. One positive and one negative. They are color-coded red and blue or black.

Stripping Wire

Let's start with some basic techniques you need to know when hacking electronics. Perhaps the most basic of these is stripping wire.

You Will Need

Quantity	Item	Appendix Code
	Wire to be stripped	T9 or scrap
1	Pliers	T1
1	Snips	T1

Whenever you hack electronics, there is likely to be some wire involved, so you need to know how to use it. Figure 1-3 shows a selection of commonly used types of wire, set beside a matchstick to give them perspective.

On the left, next to the matchstick, are three lengths of solid-core wire, sometimes called hookup wire. This is mostly used with solderless breadboard, because being made of a single core of wire inside plastic insulation, it will eventually break if it is bent. Being made of a single strand of wire does mean it is much easier to push into sockets when prototyping since it doesn't bunch up like multi-core wire.

When using it with breadboard, you can either buy already-stripped lengths of wire in various colors as a kit (see Appendix, T6) or reels of wire that you can cut to the lengths you want yourself (see Appendix, T7, T8, T9). It is useful to have at least

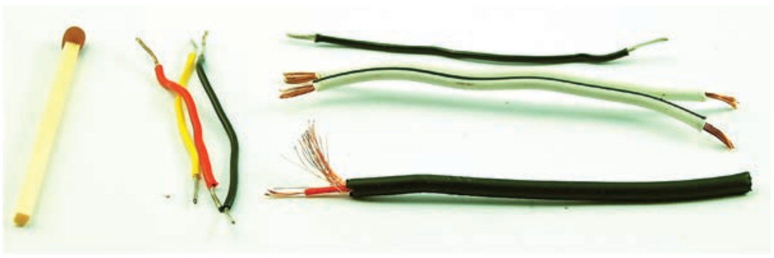


FIGURE 1-3 Common types of wire

three colors: red, yellow, and black are a good choice. It makes it easier to see how a project is connected up if you use red for the positive power supply, black for negative, and yellow for any other wires needed.

The top right of Figure 1-3 shows a length of multi-core wire, as well as some twin-strand multi-core wire. Multi-core wire is used when connecting up modules of a project. For instance, the wires to a loudspeaker from an amplifier module might use some twin, multi-core wire. It's useful to have some of this wire around. It is easily reclaimed from broken electronic devices, and relatively inexpensive to buy new (see Appendix, T10 and T11).

The wire at the bottom right of Figure 1-3 is screened wire. This is the type of wire you find in audio and headphone leads. It has an inner core of multi-core insulated wire surrounded by a screened wire on the outside. This type of wire is used where you don't want electrical noise from the environment such as mains hum (60 Hz electrical noise from 110V equipment) to influence the signal running through the central wire. The outer wire screens the inner wire from any stray signals and noise. There are variations of this where there is more than one core surrounded by the screening—for example, in a stereo audio lead.

Insulated wire is of no use to us unless we have a way of taking some of the insulation off it at the end, as this is where we will connect it to something. This is called "stripping" the wire. You can buy special-purpose wire strippers for this, which you can adjust to the diameter of the wire you want to strip. This implies that you know the width of the wire, however. If you are using some wire that you scavenged from a dead electronic appliance, you won't know the width. Having said that, with a bit of practice you will find you can strip wire just as well using a pair of pliers and some wire snips.

Wire snips and pliers are essential tools for the electronics hacker. Neither tool needs to be expensive. In fact, snips tend to get notches in them that make them annoying to use, so a cheap pair (I usually pay about USD 2) that can be replaced regularly is a good idea.

Figures 1-4a and 1-4b show how to strip a wire with pliers and snips. The pliers are used to hold things still with a firm grip, while the snips do the actual stripping.

Grip the wire in the pliers, about an inch away from the end (Figure 1-4a). Use the snips to grip the insulation where you want to take it off. Sometimes it helps to just nip the insulation all the way around before gripping it tightly with the snips, and then pull the insulation off (Figure 1-4b).

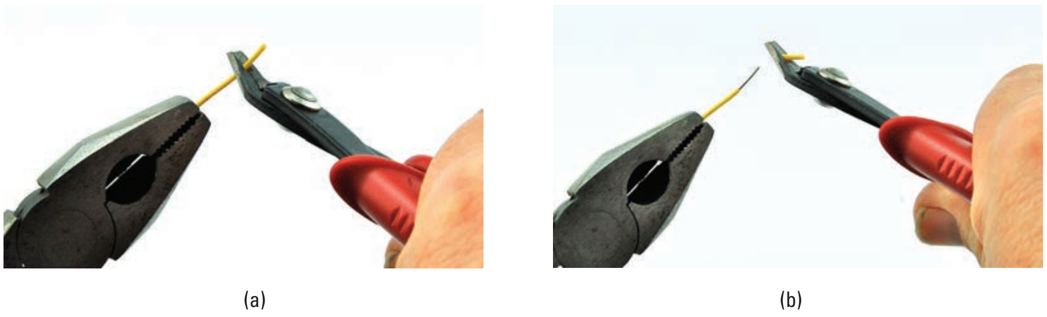


FIGURE 1-4 Stripping wire

For longer lengths of wire, you can just wrap the wire around your finger a few times instead of using pliers.

This takes a bit of practice. Sometimes you will have the snips grip it too tightly and accidentally cut the wire all the way through, while other times you won't grip it hard enough with the snips and the insulation will stay in place or stretch. Before attempting anything important, practice with an old length of wire.

Joining Wires Together by Twisting

It is possible to join wires without soldering. Soldering is more permanent, but sometimes this technique is good enough.

One of the simplest ways of joining wires is to simply twist the bare ends together. This works much better for multi-core wire than the single-core variety, but if done properly with the single-core, it will still make a reliable connection.

You Will Need

To try out joining two wires by twisting (there is slightly more to it than you might expect), you will need the following.

Quantity	Item	Appendix Code
2	Wires to be joined	T10
1	Roll of PVC insulating tape	T3

If you need to strip the wires first to get at the copper, refer back to the section "How to Strip a Wire."

Figures 1-5a thru 1-5d show the sequence of events in joining two wires by twisting them.

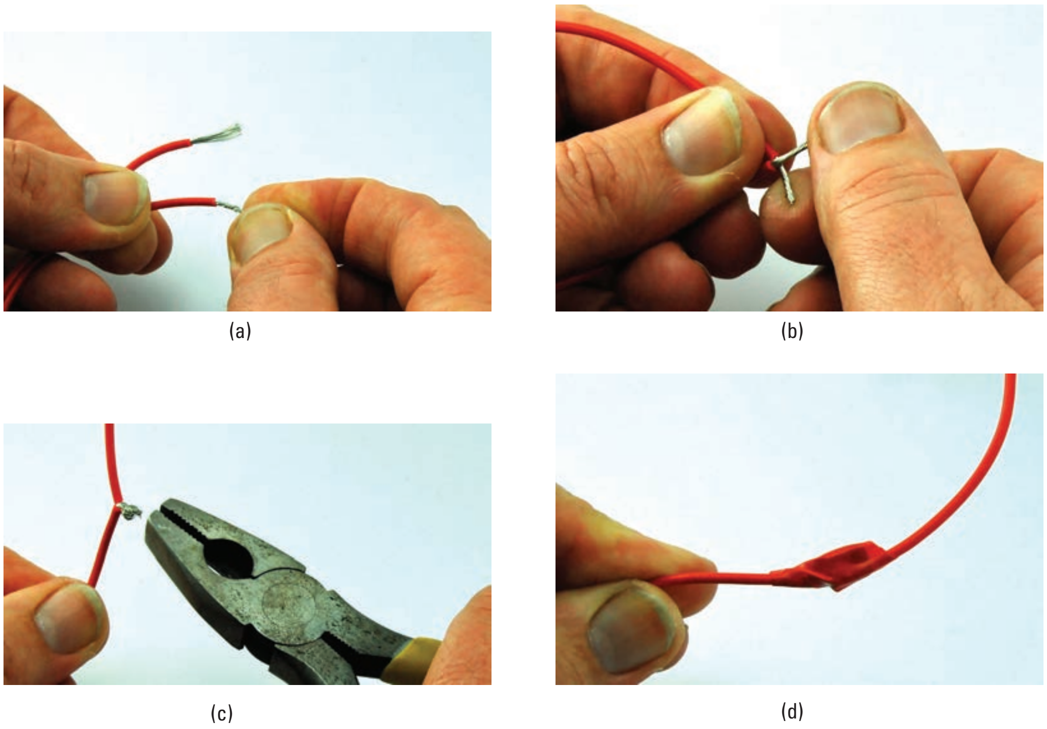


Figure 1-5 Joining wires by twisting

First, twist the strands of each wire up clockwise (Figure 1-5a). This just tidies up any straggling strands of the multi-core wire. Then, twist together the two pre-twisted wires (Figure 1-5b) so they are both twisting around each other. Try to avoid the situation where one of the wires twists around the second, while the second remains straight. If it does this, it is very easy for the first wire to just slip off the second. Next, twist the joined wires up into a neat little knot (Figure 1-5c). Note that a pair of pliers may be easier to use when making the knot, especially if the wire is on the thick side. Lastly, cover the joint with four or five turns of PVC insulating tape (Figure 1-5d).

Joining Wires by Soldering

Soldering is the main skill necessary for hacking electronics.

Safety

I don't want to put you off, but … be aware that soldering involves melting metal at very high temperatures. Not only that, but melting metal that's coupled with noxious fumes. It is a law of nature that anyone who has a motorbike eventually falls off it, and anyone who solders will burn their fingers. So be careful and follow these safety tips:

- Always put the iron back in its stand when you are not actually soldering something. If you leave it resting on the bench, sooner or later it will roll off. Or you could catch the wires with your elbow and if it falls to the floor, your natural reflex will be to try and catch it—and chances are you will catch the hot end. If you try and juggle it in one hand, while looking for something or arranging some components ready to solder, sooner or later you will either solder your fingers or burn something precious.
- Wear safety glasses. Blobs of molten solder will sometimes flick up, especially when soldering a wire or component that is under tension. You do not want a blob of molten solder in your eye. If you are long-sighted, magnifying goggles may not look cool, but they will serve the dual purpose of protecting your eyes and letting you see what you are soldering properly.
- If you do burn yourself, run cold water over the burned skin for at least a minute. If the burn is bad, seek medical attention.
- Solder in a ventilated room, and ideally set up a little fan to draw the fumes away from you and the soldering iron. Preferably have it blowing out of a window. A fun little project to practice your wire joining skills on is making a fan using an old computer (see the section "How to Hack a Computer Fan to Keep Soldering Fumes Away").

You Will Need

To practice joining some wires with solder, you will need the following items.

Quantity	Item	Appendix Code
2	Wires to be joined	T10
1	Roll of PVC insulating tape	T3
1	Soldering kit	T1
1	Magic hands (optional)	T4
1	Coffee mug (essential)	

Magic hands are a great help during soldering because they solve the problem that, when soldering, you really need three hands: one to hold the iron, one to hold the solder, and one to hold the thing or things you are trying to solder. You generally use the magic hands to hold the thing or things you are trying to solder. Magic hands are comprised of a small weighted bracket with crocodile clips that can be used to hold things in place and off the work surface.

An alternative that works well for wires is to bend them a little so that the end you are soldering will stick up from the workbench. It usually helps to place something heavy like a coffee mug on the wire to keep it from moving.

Soldering

Before we get onto the business of joining these two wires, let's have a look at soldering. If you haven't soldered before, Figures 1-6a thru 1-6c show you how it's done.

1. Make sure your soldering iron has fully heated up.
2. Clean the tip by wiping it on the damp (not sopping wet) sponge on the soldering iron stand.

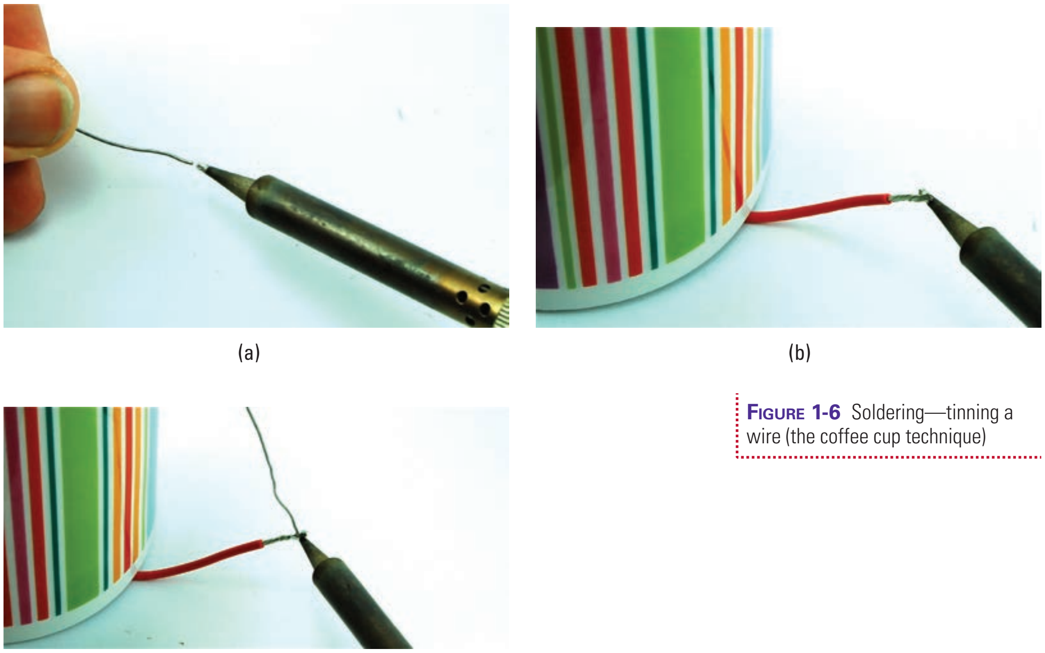


FIGURE 1-6 Soldering—tinning a wire (the coffee cup technique)

3. Touch a bit of solder onto the tip of the iron to “tin” it (see Figure 1-6a). After you have done this, the tip should be bright and shiny. If the solder doesn’t melt, then your iron probably isn’t hot enough yet. If the solder forms into a ball and doesn’t coat the tip of the iron, the tip of it may be dirty, so wipe it on the sponge and try again.
4. Hold the soldering iron to the wire and leave it there for a second or two (Figure 1-6b).
5. Touch the solder to the wire near the soldering iron. It should flow into the wire (Figure 1-6c).

Soldering is something of an art. Some people are naturally very neat at soldering. So do not worry if your results are a bit blobby at first. You will get better. The main thing to remember is that you heat up the item you want to solder and only apply the solder when that thing is hot enough for the solder to melt onto it. If you are struggling, it sometimes helps to apply the solder to the spot where the soldering iron meets the thing being soldered.

The following section offers a bit more soldering practice for you—in this case, by soldering wires together.

Joining Wires

To join two wires with solder, you can use the same approach described in the section “How to Join Wires Together by Twisting” and then flow solder into the little knot. An alternative way—that makes for a less lumpy joined wire—is illustrated in Figures 1-7a thru 1-7d.

1. The first step is to twist each end. If it is multi-core wire (a), tin it with solder as shown in Figure 1-7a.
2. Hold the wires side by side and heat them with the iron (see Figure 1-7b). Note the chopstick technique of holding both the second wire and the solder in one hand.
3. Introduce the solder to the wires so they join together into one wire and look something like that shown Figure 1-7c.
4. Wrap the joint in three or four turns of insulating tape—half an inch is probably enough (see Figure 1-7d).

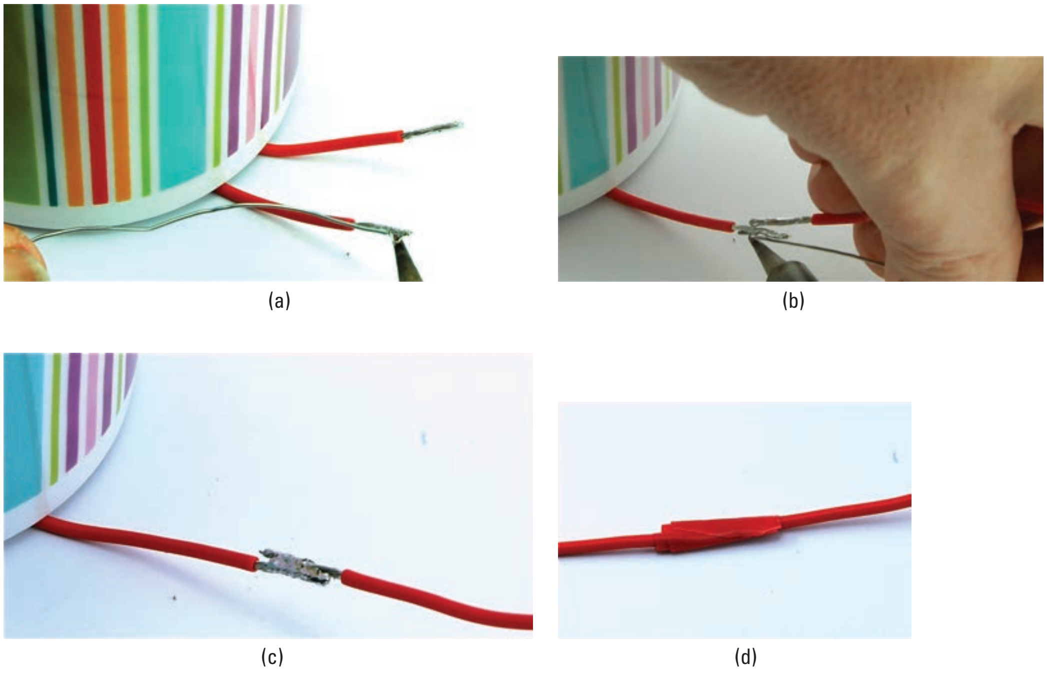


Figure 1-7 Joining wires by soldering

Testing a Connection

For the joints that we have made in the section "Joining Wires by Soldering," it is fairly obvious that they are connected. However, especially with solid-core wire, it is not uncommon for the wire core to break somewhere under the insulation. If you own an electric guitar, you will probably be familiar with the problem of a broken guitar lead.

You Will Need

Quantity	Item	Appendix Code
1	Multimeter	K1, T2
1	Connections to be tested	

Nearly all multimeters have a "Continuity" mode. When set in this useful mode, the multimeter will beep when the leads are connected to each other.

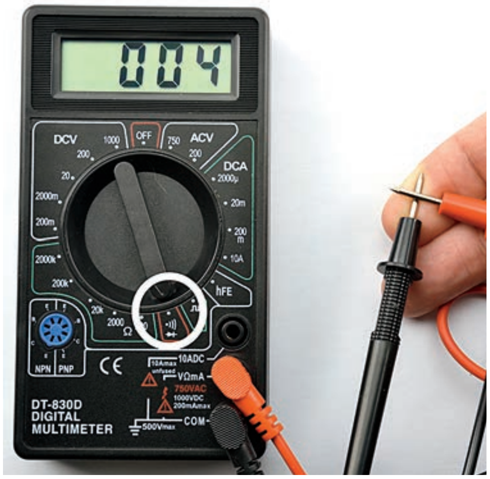


FIGURE 1-8 A multimeter in Continuity mode

Set your multimeter to "Continuity mode," and then try touching the leads together—this should cause the multimeter to make a beeping sound. The Continuity mode is often indicated by a musical note or some other icon to indicate making a noise. Now take a length of wire and try touching the multimeter leads to each end of the wire (Figure 1-8). The buzzer should sound if the wire is okay.

You can use this technique on circuit boards. If you have an old bit of circuit board from something, try testing between the soldered connections on the same track (Figure 1-9).

If there is no connection where you would expect there to be a connection, then there may be a "dry joint," where the solder hasn't flowed properly or there is a crack in the track on the circuit board (this sometimes happens if the board gets flexed).

A dry joint is easily fixed by just applying a bit of solder and making sure it flows properly. Cracks on a circuit board can be fixed by scraping away some of the protective lacquer over the track and then soldering up the split in the track.

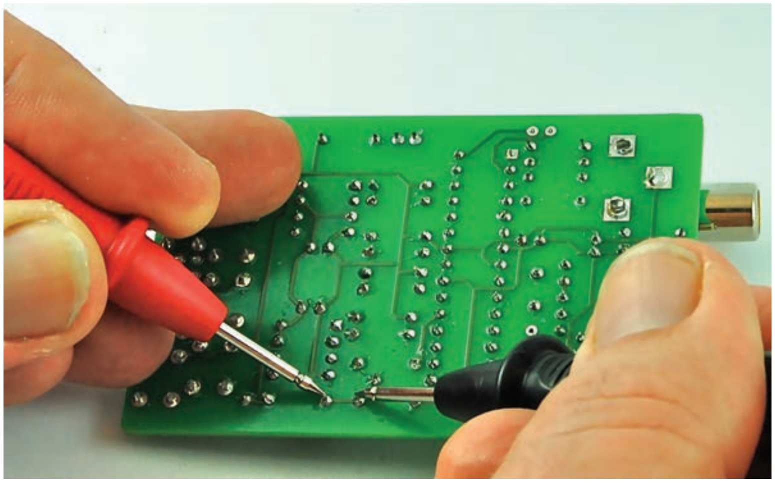


FIGURE 1-9 Testing a circuit board

Hacking a Computer Fan to Keep Soldering Fumes Away

Solder fumes are unpleasant and bad for you. If you can sit by an open window while you solder, then great. If not, then this is a good little construction project to enhance your electronics hacking skills (Figure 1-10).

Okay, so it's not going to win any awards for style, but at least the fumes are directed away from my face.

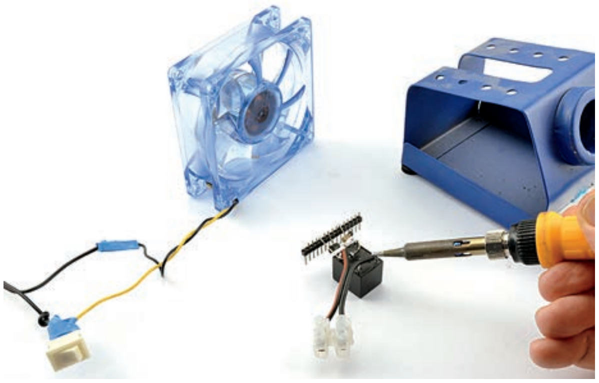


FIGURE 1-10 A homemade fume extractor

You Will Need

Quantity	Item	Appendix Code
1	Soldering kit	T1
1	An old computer fan (two-lead)	
1	12V power supply	M1
1	Toggle switch	K1

Construction

Figure 1-11 shows the schematic diagram for this mini-project.

Newcomers to electronics often view schematic diagrams like this with suspicion, thinking it better just to show the components as they actually are, with wires where wires need to be—just like in Figure 1-12. It is worth learning how to read a schematic diagram. It really isn't that hard and in the long term it will pay dividends, not least because of the vast number of useful circuit diagrams published on the Internet. It's a bit like being able to read music. You can get so far playing by ear, but there are more options if you can read and write musical notation.

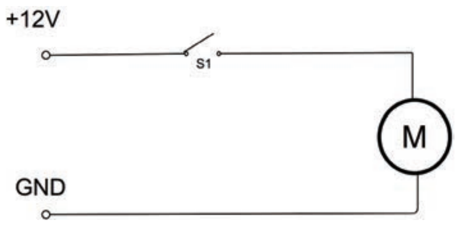


FIGURE 1-11 The schematic diagram for the fume extractor

So, let's examine our schematic diagram. Over on the left we have two labels that say "+12V" and "GND." The first is the 12V positive supply from the 12V power supply. GND actually refers to the negative connection of the power supply. GND is short for "ground" and just means zero volts. Voltage is relative,

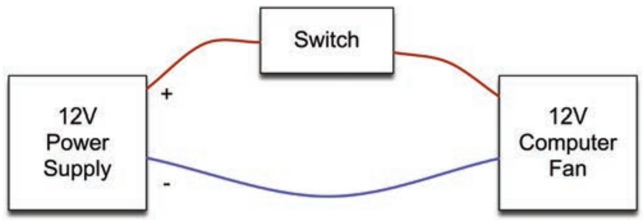


FIGURE 1-12 The wiring diagram for the fume extractor

so the 12V connection of the power supply is 12V above the other connection (the GND connection). We will learn more about voltage in the next chapter.

Moving toward the right, we have a switch. This is labeled “S1,” and if we had more than one switch in a schematic, they would be labeled “S2,” “S3,” and so on. The symbol for a switch shows how it operates. When the switch is turned to the on position, its two connections are connected together, and when it is in the off position, they aren’t. It’s as simple as that.

The switch is just controlling the supply of electricity to the motor of the fan (M) as if it were a faucet.

Step 1. Strip the Power Supply Leads

We have a power supply and we are going to cut the plug off the end of it and strip the wires (see the section “Stripping a Wire”). Before you cut off the plug, make sure the power supply is NOT plugged in. Otherwise, if you snip both wires at the same time, the cutters will probably short the two connections together, which may damage the power supply.

Step 2. Identify the Power Supply Lead Polarity

Having cut the wires, we need to know which one is the positive one. To do this, let’s use a multimeter. Set the multimeter to its 20V DC range. Your multimeter will probably have two voltage ranges, one for AC and one for DC. You need to use the DC range. This is often marked by a solid line above a dotted line. The AC range will either be marked as AC or have a picture of a little sine wave next to it. If you select AC instead of DC, it will not damage the meter, but you will not get a meaningful reading. (See Chapter 11 if you need more information on multimeters.)

First making sure that the stripped leads from the power supply are not touching, plug the power supply in and turn it on.

Connect the multimeter leads to the power supply leads (Figure 1-13). You can use the normal test leads, but in this case,

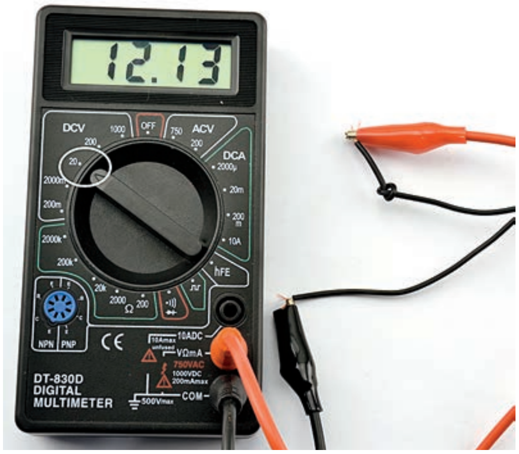


FIGURE 1-13 Using a multimeter to find the power supply polarity

alligator-clip test leads are easiest to use. If the number on the multimeter is positive, then the red test lead of the multimeter is connected to the positive lead. Mark the lead in some way (I tied a knot in it). However, if the multimeter shows a negative voltage, then the leads are swapped over, so tie a knot in the power supply lead connected to the black test lead of the multimeter—in this case, this is the positive lead from the power supply.

Step 3. Connect the Negative Leads Together

Unplug your power supply. You should never solder anything that is powered up.

Cut any plug off the end of your computer fan and strip the two wires. Mine had one black (negative) and one yellow (positive) lead. Three lead fans are more complex and should be avoided. If you get the leads the wrong way around, no harm will befall you. The fan will just rotate in the opposite direction.

We are now going to join the negative lead of the fan to the negative lead (no knot) of the power supply (Figure 1-14).

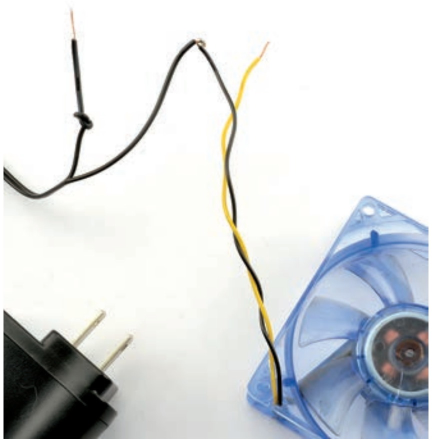


FIGURE 1-14 Connecting the negative leads together

Step 4. Connect the Positive Lead to the Switch

Solder the positive lead from the power supply to one of the outer connections on the switch (it doesn't matter which). (See

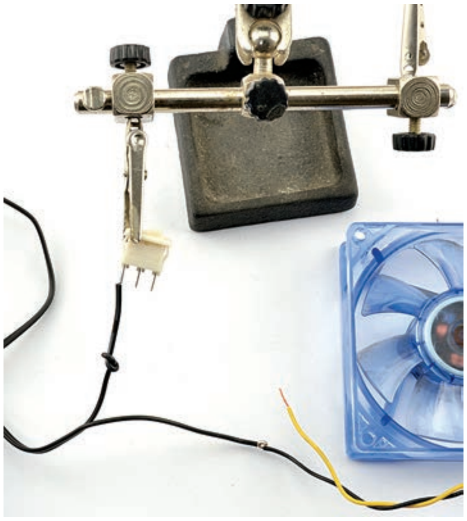


FIGURE 1-15 Connecting the positive lead to the switch

Figure 1-15.) It will help to tin the switch connection with a little solder before you start.

Finally, connect the remaining lead from the fan to the center connection of the switch (see Figure 1-16).

Step 5. Try It Out

Wrap the bare connections with insulating tape, plug it in, turn it on, and presto! When you flick the switch, the fan should come on.

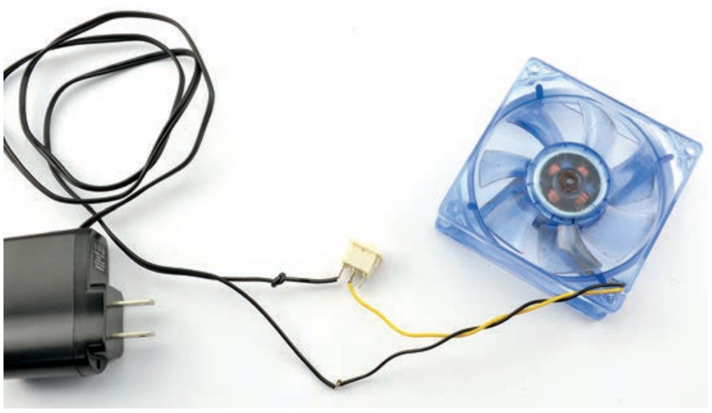


FIGURE 1-16 Connecting the fan to the switch

Summary

Now that we have the basics and are confident about a bit of soldering and dealing with wires and switches, we can move on to Chapter 2. There, we will start looking at a few electronic components, as well as some of the basic ideas you will need to understand to successfully hack electronics.

2

Components

There are a few fundamentals that will help us get the most out of our electronics. I have no intention of overloading you with theory, so you may find you come back to this chapter as and when you need to. But before we start on any theory, let's look at getting together some of the components we will use.

A Starter Kit of Components

In Chapter 1, we assembled a few tools and did some soldering. The only thing we made used a scavenged computer fan, an off-the-shelf power supply, and a switch.

Certain components you will find that you use over and over again. To get yourself a basic stock of components, I recommend you buy a starter kit. SparkFun, Adafruit, and MonkMakes all sell general-purpose, basic component kits (see Appendix, K1). Once you have these, you will have a useful collection of components that should cover 80 percent of what you need.

Other suppliers sell starter kits, and although none of them will contain everything you need for this book, most will give you a very good starting point.

The MonkMakes Hacking Electronics Mega Kit includes many of the components used in this book, along with a few tools such as a multimeter and breadboard, as well as jumper wires and a DC motor. For a full list of the components included in the kit, see https://monkmakes.com/hacking2.

You will find that projects and experiments in the book usually have a "You Will Need" section that lists the parts required. These are listed with an "Appendix Code," and if you refer to the Appendix at the end of this book, you can look up the component along with information on where to get it. Where a component has an Appendix Code of K1, it means that that part is included in the MonkMakes Hacking Electronics Mega Kit.

Identifying Electronic Components

So, what have we just bought here? Let's go through some of the components you'll find in the starter kits and explain what they do, starting with the resistors.

Resistors

Figure 2-1 shows an assortment of resistors. Resistors come in different sizes to be able to cope with different amounts of power. High-power resistors are physically big to cope with the heat they produce. Since "parts getting hot" is generally a bad thing in electronics, we will mostly avoid that. Nearly all of the time we can use 0.25-watt resistors, which are perfect for general use.

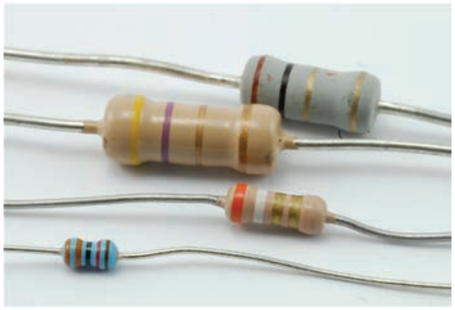


FIGURE 2-1 Assorted resistors

As well as having a maximum power rating, resistors also have a "resistance." As the word suggests, resistance is actually resistance to the flow of current. So a high-resistance resistor will not allow much current to flow, while a low-value resistor will allow lots of current to flow.

Resistors are the most commonly used component you can find. Since we will be using them a lot, we will go into greater detail on the subject in the section "What Are Current, Resistance, and Voltage?" later in this chapter.

Resistors have little stripes on them that tell you their value. You can learn to read the stripes (more in a moment on that) or you can avoid all of this by storing them in a bag or in the drawer of a component box with the value written on the box or bag. If in doubt, check the value with the resistance measurement feature of your multimeter.

However, an essential piece of geekiness is to know your resistor color codes. Each color has a value, as shown in Table 2-1.

Gold and silver, as well as representing the fractions 1/10 and 1/100, are also used to indicate how accurate the resistor is. So gold is ±5% and silver is ±10%.

There will generally be three of these bands grouped together at one end of the resistor. This is followed by a gap, and then a single band at the other end of the resistor. The single band indicates the accuracy of the resistor value. Since none of the projects in this book require very accurate resistors, there is no need to select your resistors on the basis of accuracy.

Color	Value
Black	0
Brown	1
Red	2
Orange	3
Yellow	4
Green	5
Blue	6
Violet	7
Gray	8
White	9
Gold	1/10
Silver	1/100

TABLE 2-1 Resistor Color Codes

Figure 2-2 shows the arrangement of the colored bands. The resistor value uses just the three bands. The first band is the first digit, the second the second digit, and the third "multiplier" band is how many zeros to put after the first two digits.

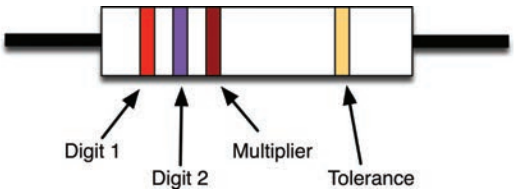


FIGURE 2-2 Resistor stripes

So, a 270Ω (*ohm*) resistor will have first digit 2 (red), second digit 7 (violet), and a multiplier of 1 (brown). Similarly, a 10kΩ resistor will have bands of brown, black, and orange (1, 0, 000).

In addition to fixed resistors, there are also variable resistors (a.k.a., potentiometers or pots). This comes in handy with volume controls, where turning a knob changes the resistance and alters the level of sound.

Capacitors

When hacking electronics, you will occasionally need to use a capacitor. Luckily, you do not need to know much about what they do. They are often used to head-off problems like the instability of a circuit or unwanted noise. Their use is often given a name like "decoupling capacitor" or "smoothing capacitor." There are simple rules you can follow about where you need a capacitor. These will be highlighted as we encounter them in later sections.

For the curious, capacitors store charge, a bit like a battery, but not much charge, and they can store the charge and release it very quickly.

Figure 2-3 shows a selection of capacitors.

If you look closely at the second capacitor from the left, you will see the number 103. This is actually the value of the capacitor in picofarads. The unit of capacitance is farad, but a 1F capacitor would be considered a huge capacitor, storing a great deal of charge. So, while such beasts do exist, everyday capacitors are either measured in nanofarads (nF = 1/1,000,000,000F) or microfarads (μF = 1/1,000,000F). You will also find capacitors in the picofarad range (pF = 1/1,000,000,000,000F).

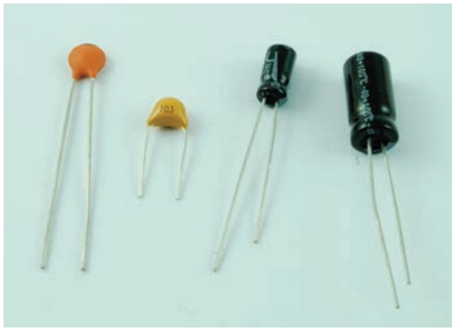


FIGURE 2-3 Assorted capacitors

Returning to 103, rather like resistors, this means 10 and then 3 zeros, in units of pF. So in this case that's 10,000pF or 10nF.

Larger capacitors, like those on the right of Figure 2-3, are called electrolytic capacitors. They are usually in the μF (micro Farads) range and have their value written on their side. They also have a + and a – side, and unlike most other capacitors must be connected the right way around.

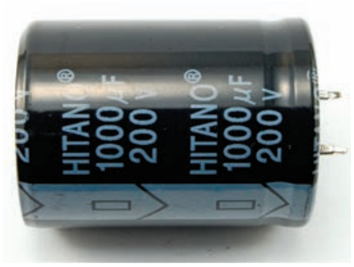


FIGURE 2-4 An electrolytic capacitor

Figure 2-4 shows a large electrolytic, with value (1000μF) and its negative lead clearly indicated at the bottom of the figure. If the capacitor has one lead longer than the other, the longer one will normally be the positive lead.

The capacitor in Figure 2-4 also has a voltage written on it (200V). This is the capacitor's maximum voltage. So if you put more than 200V across its leads, it will fail. Big electrolytic capacitors like this have a reputation for failing spectacularly and may burst, spewing forth goo.

Diodes

You will occasionally need to use diodes. They are kind of a one-way valve, only allowing current to flow in one direction. They are therefore often used to protect sensitive components from accidental reverse voltage that could damage them.

Diodes (Figure 2-5) have a stripe at one end. That end is called the cathode, while the other end is called the anode. We will hear more about diodes later.

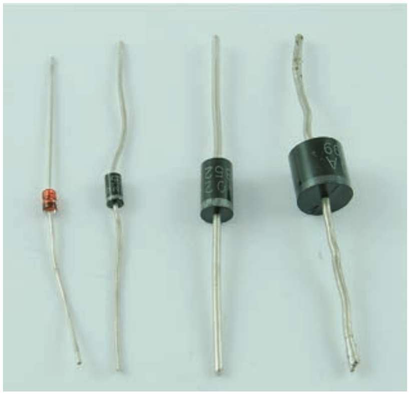


FIGURE 2-5 A selection of diodes

As with resistors, the bigger the diode physically, the more power it can cope with before it gets too hot and expires.

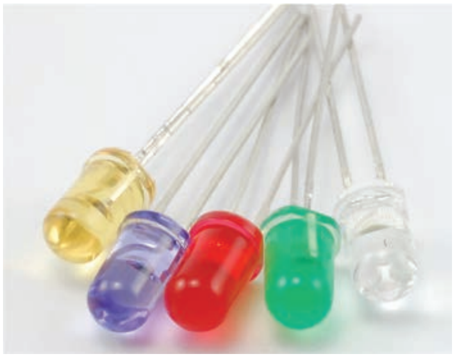


FIGURE 2-6 Assorted LEDs

Ninety percent of the time, you will just be using one of the two diodes on the left-hand side of the figure.

LEDs

LEDs light up, and generally look pretty. Figure 2-6 shows a selection of LEDs.

LEDs are a little sensitive, so you should not connect them directly to a battery. Instead you have to use a resistor to reduce the current flowing into the LED. If you do not do this, the LED will probably die almost instantly.

Later on, we will see how to select the right resistor for the job.

Just like regular diodes, LEDs have a positive and a negative lead (anode and cathode). The positive anode is the longer of the two leads. There is also usually a flat side to the LED case on the cathode side.

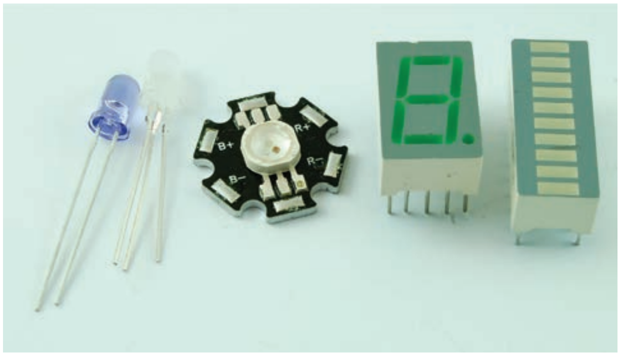


FIGURE 2-7 More LEDs

As well as single LEDs, you also get LEDs in more complicated arrangements within a single package. Figure 2-7 shows some interesting-looking LEDs.

From left to right, these LEDs are an ultraviolet LED, an LED with both red and green LEDs in the same package, a high-power RGB (red, green, blue) LED that can be controlled to produce any color of light, a seven-segment LED display, and an LED bar graph display.

This is just a small selection of LED types. There are many others to choose from. In later sections, we will explore some of these more exotic LEDs.

Transistors

While transistors can be used in audio amplifiers and in many circumstances, for the casual electronics hacker, the transistor can be thought of as a switch. But rather than a switch controlled by a lever, it is a switch that switches a big current, yet is controlled by a small current.

Generally speaking, the physical size of the transistor (Figure 2-8) determines how big the current that it switches can be before it starts producing smoke.

Of the transistors in Figure 2-8, the right-hand two are quite specialized and employed for high power use.

Generally, the rule for a component is that if it's ugly and has three legs, it's probably some kind of transistor.

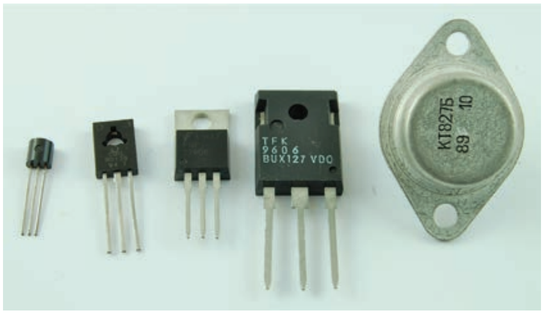


Figure 2-8 Transistors

Integrated Circuits

An integrated circuit (IC), or just "chip," is a load of transistors and other components printed onto silicon. The purpose of the IC varies wildly. It can be a microcontroller (mini-computer), or an entire audio amplifier, or a computer memory, or any one of thousands of other possibilities.

ICs make life easy, because as they say, often "there's a chip for that." Indeed, if there is something you want to make, there may well be a chip for it already, and if there isn't, then there will probably be a general-purpose chip that takes you halfway there.

ICs look like bugs (Figure 2-9).

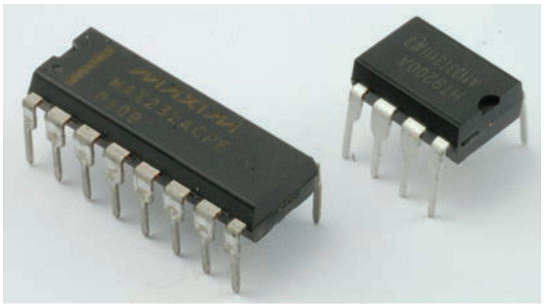


Figure 2-9 Integrated circuits

Other Stuff

There are so many other components out there, some of which are very familiar, such as batteries and switches. Others are less familiar and include potentiometers (variable resistors found in volume controls), phototransistors, rotary encoders, light dependent resistors, and so on. We will explore these as they arise later in the book.

Surface Mount Components

Let's touch a little on the subject of surface mount devices (SMDs). These components are just resistors, transistors, capacitors, ICs, and so on, but in tiny packages designed to be soldered onto the top surface of circuit boards by machines.

Figure 2-10 shows a selection of SMDs.

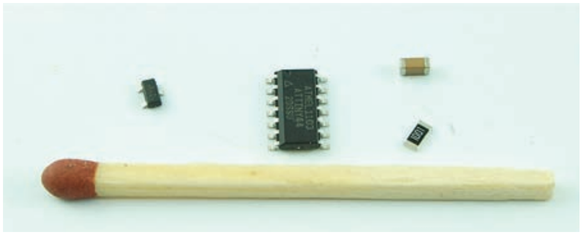


FIGURE 2-10 Surface mount components

The matchstick shows you just how small these devices are. It is perfectly possible to do surface mount soldering by hand, but you need a steady hand and a high-quality soldering iron. Not to mention a lot of patience. You are also likely to need a circuit board to solder them onto, as they are not easy to use with breadboard or other prototyping tools.

In this book, we mostly look at using the conventional "through-hole" components rather than SMDs. However, as your experience grows and you feel you might like working with SMDs, do not be afraid to try.

What Are Current, Resistance, and Voltage?

Voltage, current, and resistance are three properties that are fundamental to almost everything you will do in electronics. They are intimately related, and if you can master the relationship between them, you will be a wise hacker indeed.

Please take the time to read and understand this little bit of theory. Once you understand it, many other things should automatically fall into place.

Current

The problem with electrons is that you cannot see them, so you just have to imagine how they do things. I like to think of electrons as little balls flowing through pipes. Any physicists reading this will probably be clutching their heads or hurling this book to the floor in disgust now. But it works for me.

Each electron has a charge and it's always the same—lots of electrons, lots of charge, few electrons, and a little bit of charge.

Current, rather like the current in a river, is measured by counting how much charge passes you per second (Figure 2-11).

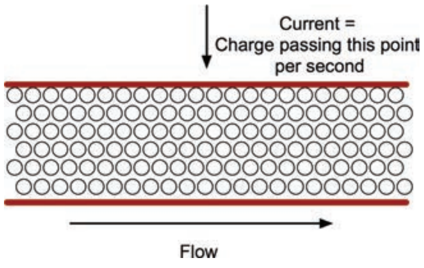


FIGURE 2-11 Current

Resistance

A resistor's job is to provide resistance to the flow of current. So, if we keep thinking about our river, it is like a constriction in a river (Figure 2-12).

The resistor has reduced the amount of charge that can pass by a point. And it doesn't matter which point you measure at (A, B, or C) because, if you look upstream of the resistor, the charge is hanging around waiting to move through the resistor. Therefore, less is moving past A per second. In the resistor (B), it's restricted.

The "speed" analogy does not really hold true for electrons, but one important point is that the current will be the same wherever you measure it.

Imagine what happens when a resistor stops too much current from flowing through an LED.

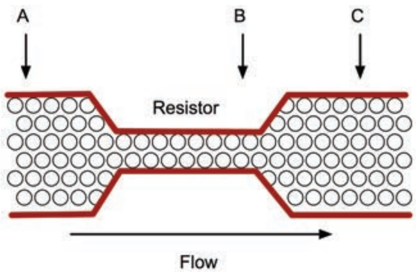


FIGURE 2-12 A resistor

Voltage

Voltage is the final part of the equation (that we will come to in a minute). If we persist with the water-in-a-river analogy, then voltage is like the height that the river drops (Figure 2-13).

As everyone knows, a river that loses height quickly flows fast and furious, whereas a relatively gently sloped river will have a correspondingly gentle current.

This analogy helps with the concept of voltage being relative. That is, it does not matter if the river is falling from 10,000 ft to 5,000 ft or from 5,000 ft to 0 ft. The drop is the same and so will be the rate of flow.

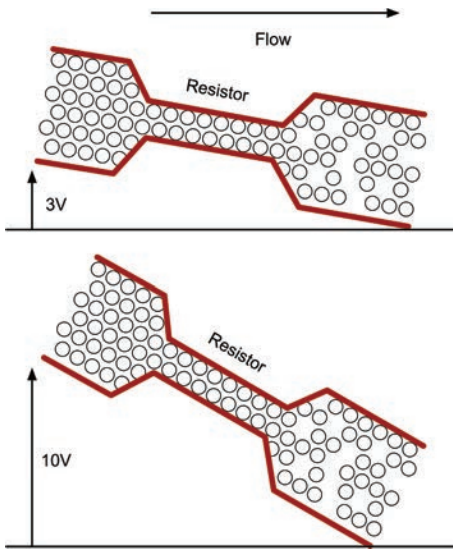


FIGURE 2-13 Voltage

Ohm's Law

Before we get into the math of this, let's think for a moment about current, voltage, and resistance and how they relate to each other.

Try this little quiz. Think in terms of the river if you find it helps.

1. If the voltage increases, will the current (a) increase or (b) decrease?

2. If the resistance increases, will the current (a) increase or (b) decrease?

Did you get the answers (a) for question 1 and (b) for question 2?

If you write this down as an equation, it is called Ohm's law and can be written as:

$I = V / R$

I for current (I guess "C" was already taken), V for Volts, and R for resistance.

So, the current flowing through a resistor, or any wire connecting to it, will be the voltage across the resistor divided by the resistance of the resistor.

The units of resistance are in Ω (the abbreviation for ohms), while units of current are in A (short for amps, which is short for amperes) and in voltage V (the easy one).

So, if we have a voltage of 10V across a resistor of 100Ω the current flowing will be

$10V / 100\Omega = 0.1A$

For convenience, we often use mA (1/1000 of an amp). So 0.1A is also 100mA.

That's enough about Ohm's law for now; we will meet it again later. It is the single most useful thing you can know about electronics. In the next section, we will look at the only other truly essential math you will need—power.

What Is Power?

Power is all about energy and time. So, in a way, it's a bit like current. But, instead of being the amount of charge passing a point, it is the amount of energy transformed into heat per second when a current passes through something that resists the flow (like a resistor). Forget the river, it doesn't really help much here.

Restricting the flow of a current generates heat, and the amount of heat generated can be calculated as the voltage across a resistor times the current flowing through it. The units of power are the watt (W). You would write this in math as:

$P = I \times V$

So, in our earlier example, we had 10V across a 100Ω resistor, so the current through the resistor was 100mA and will generate 0.1A × 10V, or 1 W of power. Given that the resistors that we have from any starter kit are 250 mW (0.25 W), our resistor will get hot and may eventually break.

If you don't know the current, but you do know the resistance, another useful formula for calculating the power is

$$P = V^2 / R$$

Or, power is voltage squared (times itself) divided by the resistance. So, for the example earlier:

$$P = 10 \times 10 / 100 = 1 \text{ W}$$

That's reassuringly the same answer as we got before.

Most components have a maximum power rating like this, so when selecting a resistor, transistor, diode, and so on, it is worth doing a quick check and multiplying the voltage across the component by the current that you expect to flow through it. Then, choose a component with a maximum power rating comfortably greater than the expected power.

Power is the best measure of how much electricity is being used. It is the electrical energy being used per second, and unlike current it can be compared for devices operating from both 110 volt outlets and low voltage. It is good to have a basic understanding of just how much—or how little—electricity devices use. Table 2-2 shows some devices you might find around the home and lists how much power they use.

Device	Power
Battery-powered FM radio (volume down)	20 mW
Battery-powered FM radio (volume up)	500 mW
Arduino Uno microcontroller board (9V supply)	200 mW
Raspberry Pi Model 3	2 W
Home WiFi router	10 W
Compact fluorescent (low-power) light bulb	15 W
Filament light bulb	60 W
LCD TV 40-inch	200 W
Electric kettle	3000 W (3 kW)

TABLE 2-2 Power Usage

So, now you know why you don't get battery-powered kettles!

Reading a Schematic Diagram

Hacking electronics often involves trawling the Internet, looking for people who have made something like the thing you want

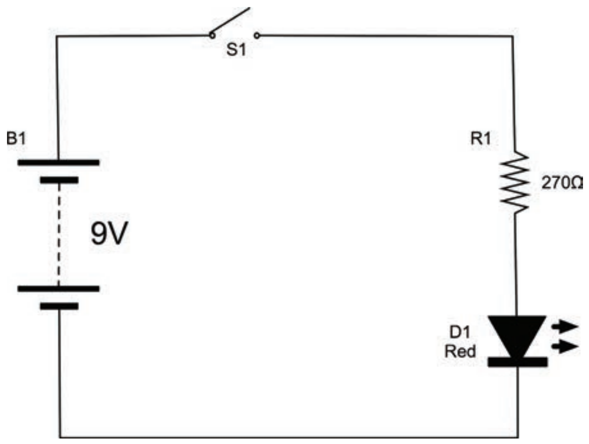


FIGURE 2-14 A simple schematic

to make or adapt. You will often find schematic diagrams that tell you how to make and do things. So you need to be able to understand these schematics in order to turn them into real electronics.

These may at first sight seem a little baffling, but schematics obey a few simple rules and tend to use the same patterns over and over again. So there is a lot less to learn than you might think.

Ponder Figure 2-14 while we consider some of these rules—or more accurately conventions—because sometimes they are broken.

Figure 2-14 goes a long way to explaining why we sometimes talk of electronic circuits. It's kind of a loop. The current is flowing out of the battery, through the switch (when it's closed), through the resistor and LED (D1), and then back to the battery. The lines on the schematic can be thought of as perfect wires without any resistance.

The First Rule of Schematics: Positive Voltages Are Uppermost

A convention that most people follow when drawing a schematic is to put the higher voltages near the top, so on the left-hand side of the diagram, we have a 9V battery. The bottom of the battery is at 0V or GND (Ground), while the top of the battery will be 9V higher than that.

Notice that we draw the resistor R1 above the LED (D1). This way, we can think of some of the voltage as being lost across the resistor, before the remainder is lost through the diode and flows back to the negative connection of the battery.

Second Rule of Schematics: Things Happen Left to Right

Western civilization invented electronics and writes from left to right. You read from left to right and, culturally, more things happen from left to right. Electronics is no different, so it is common to start with the source of the electricity—the battery or power supply on the left—and then work our way from left to right across the diagram.

So, next we have our switch, which controls the flow of the electricity, and then the resistor and LED.

Names and Values

It is normal to give every component in a schematic a name. So, in this case the battery pack is called B1, the switch S1, the resistor R1, and the LED D1. This means that when you go from a schematic to a breadboard layout and eventually a circuit board, you can see which components on the schematic correspond to which components on the breadboard or circuit board.

It is also normal to specify the value of each of the components where appropriate. So, for example, the resistor's value of 270Ω is marked on the diagram. The rest of the components don't need much else said about them.

Component Symbols

Table 2-3 lists the most common circuit symbols you will encounter. This is nothing like a complete list, but we will discuss other symbols later in the book.

There are two main styles of circuit symbol: American and European. Fortunately, they are similar enough to avoid difficulties in recognizing them.

In this book, we will use the U.S. circuit symbols.

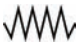
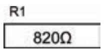
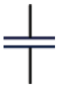
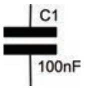
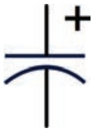
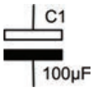
Symbol (U.S.)	Symbol (European)	Photo	Component	Use
			Resistor	Resisting the flow of current
			Capacitor	Temporary charge storage
			Capacitor (polarized)	High capacity temporary charge storage

TABLE 2-3 Common Schematic Symbols

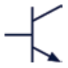
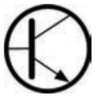
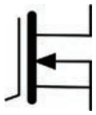
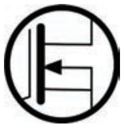
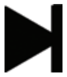
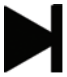
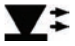
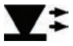
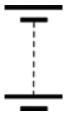
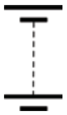
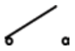
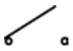
Symbol (U.S.)	Symbol (European)	Photo	Component	Use
			Transistor (bipolar NPN)	Using a small current to control a larger current
			Transistor (MOSFET N-channel)	Using a control voltage to control a larger current
			Diode	Prevents current from flowing in the wrong direction
			LED	Indication and illumination
			Battery	Power supply
			Switch	Turning things on and off

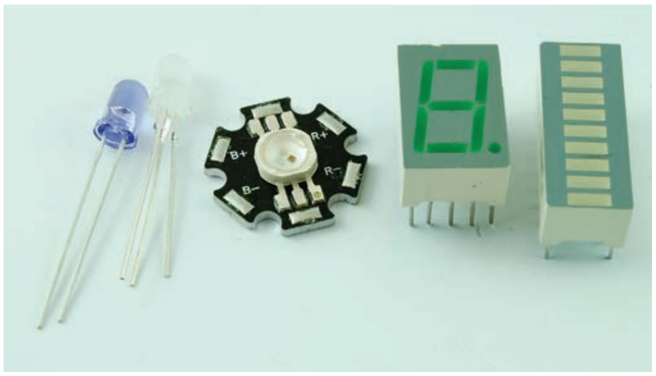
TABLE 2-3 Common Schematic Symbols

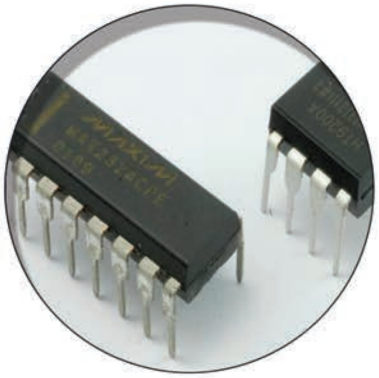
Summary

In the next chapter, we get a much more practical look at some basic hacks and hone our electronic construction skills. This includes using prototyping boards and taking our soldering beyond simply connecting wires to other wires.

We will also learn how to use solderless breadboard so we can build electronics quickly and get underway.

ARDUINO
UNO
DIGITAL (PWM~)


B+
R+
R−
B−



3

Basic Hacks

This chapter contains a set of fairly basic “hacking” knowledge. You will also be carrying out a few experiments and coming to grips with transistors and LEDs for the first time.

Making a Resistor Get Hot

Sometimes things will get hot when you are hacking electronics. It’s always better when this is expected rather than when it’s a surprise, so it’s worth doing a little experimenting in this area.

You Will Need

Quantity	Item	Appendix Code
1	100Ω 0.25-watt resistor	K2
1	4 × AA battery holder	H1
1	4 × AA batteries (the rechargeable type is a good idea)	

Figure 3-1 shows the schematic diagram.

The Experiment

All we will do is connect the 100Ω resistor across the battery terminals and see how hot it gets.

Caution Be careful when doing this because the resistor’s temperature will rise to about 50°C/122°F. The resistor’s leads, however, will not get very hot.

We are using a battery holder that takes four AA cells, each providing about 1.5V. They are each connected, one after the other, providing us with 6V total. Figure 3-2 shows how the batteries are actually connected within the battery box as a schematic diagram. In this kind of arrangement, the batteries are said to be in series.

Figure 3-3 shows the resistor heater in action.

Simply touch a finger to the resistor to confirm it's hot.

Is this bad/good? Will the resistor eventually break because it's warm? No, it won't. Resistors are designed to cope with a bit of heat. If we do the math, the power that the resistor is burning is the voltage squared divided by the resistance, which is:

$(6 \times 6) / 100 = 0.36W$

If it is a 0.25W resistor, then we are exceeding its maximum power. This would be a foolish thing to do if we were designing a product for mass production. However, that's *not* what we are doing, and the chances are the resistor would continue to work like that indefinitely.

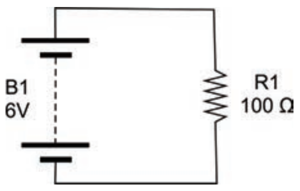


FIGURE 3-1 The schematic for heating a resistor

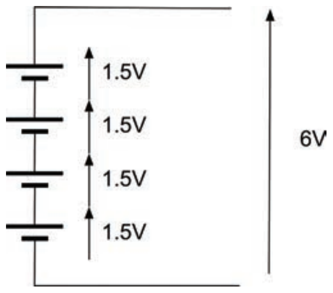


FIGURE 3-2 The schematic diagram for a battery holder

FIGURE 3-3 Making a resistor get hot

Using Resistors to Divide a Voltage

Sometimes voltages are too big. For example, in an FM radio, the signal going from the radio part to the audio amplifier part will be deliberately too large so it can be reduced using the volume knob.

Another example might be when you have a sensor that produces a voltage between 0 and 10V but you want to connect it to an Arduino microcontroller that expects it to be between 0 and 5V.

A very common technique in electronics is to use a pair of resistors (or a single variable resistor) as a "voltage divider."

You Will Need

Quantity	Item	Appendix Code
1	10kΩ trimpot (tiny variable resistor)	K1, R1
1	Solderless breadboard	K1, T5
2	Male-to-male jumper wires or solid core wire	K1, T6
1	4 × AA battery holder	K1, H1
1	4 × AA batteries	
1	Battery clip	K1, H2
1	Multimeter	K1, T2

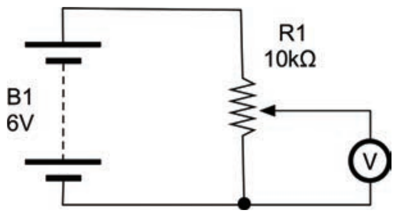


Figure 3-4 A voltage divider schematic diagram

Figure 3-4 shows the schematic diagram for our experiment. There are a couple of new schematic symbols here. The first is the variable resistor (or pot). This looks like a regular resistor symbol, but has a line with an arrow connecting to the resistor. This is the moving slider connection of the variable resistor.

The second new symbol is the circle with a "V" in it. This is a voltmeter, which in our case is the multimeter set to its DC voltage range.

The variable resistor has three leads. One lead is fixed at each end of a conductive track, while a third connection to the central slider moves from one end of the track to the other. The overall resistance of the whole track is 10kΩ.

Our voltage in is going to be supplied from the battery pack and will be roughly 6V. We are going to use a multimeter to measure the output voltage and see how much it is being reduced by our voltage divider as we rotate the variable resistor's shaft.

Construction

Figure 3-5 shows the breadboard layout for this experiment.

If you remember, the grey bars indicate where the connections underneath the holes are connected together. Take some time to follow the lines on the stripboard and reassure yourself that everything is connected in the same way as the schematic (Figure 3-4).

Plug the trimpot (variable resistor) into the breadboard as shown, and then connect the battery by carefully pushing the leads into the breadboard at the positions shown in Figure 3-5. If you struggle to get the multi-core wires of the battery clip into the holes, solder a bit of solid-core wire to the end of the leads.

Finally, attach the multimeter. If your multimeter has alligator clips, use these in preference to the normal probes, clipping short jumper wires into the alligator clips and then pushing the other ends into the positions shown in Figure 3-5. When you have done all this, your breadboard should look something like Figure 3-6a.

Turn the trimpot to its fully clockwise position. The multimeter should read 0V (Figure 3-6a). Now turn it fully anti-clockwise and it should read something around 6V (Figure 3-6b)—in other words, the full battery voltage. Finally, turn it to roughly the middle position and you should see that the meter indicates about 3V (Figure 3-6c).

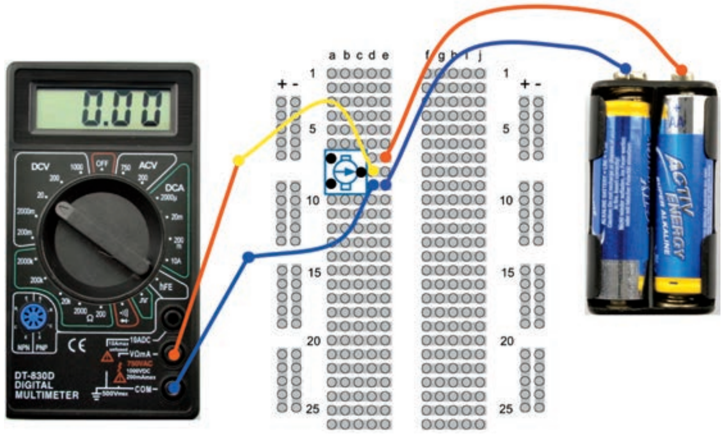


FIGURE 3-5 A voltage divider breadboard layout

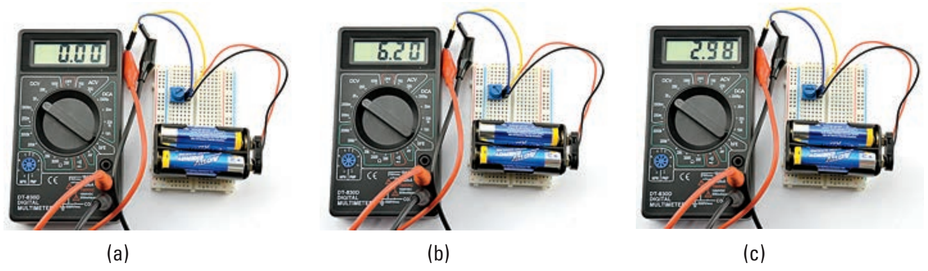


FIGURE 3-6 A voltage divider breadboard

Think of the variable resistor as behaving a bit like two resistors, R1 and R2, as shown in Figure 3-7.

The formula to calculate Vout if we know Vin, R1 and R2 is as follows:

Vout = Vin * R2 / (R1 + R2)

So, if R1 and R2 are both 5 kΩ and Vin is 6V, then:

Vout = 6V * 5kΩ / (5kΩ + 5kΩ) = 30 / 10 = 3V

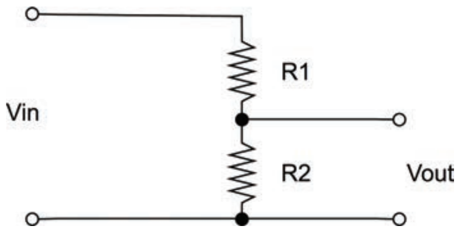


FIGURE 3-7 A voltage divider with fixed resistors

This ties in with what we found when we put the trimpot to its middle position. It is exactly the same as having two fixed resistors of 5 kΩ each.

As with many of the calculations you make in electronics, people have made handy calculating tools. If you type "voltage divider calculator" into a search engine, you will find them. One such example can be found here: www.electronics2000.co.uk/calc/potential-divider-calculator.php.

These calculators will also usually match to the nearest available fixed resistor value.

Converting a Resistance to a Voltage (and Make a Light Meter)

A photoresistor is a resistor whose resistance changes depending on the amount of light falling on its transparent window. We will use one of these devices to demonstrate the idea of converting a resistance to a voltage by using it as one-half of a potential divider.

You Will Need

Quantity	Item	Appendix Code
1	Photoresistor (1kΩ)	K1, R2
1	Solderless breadboard	K1, T5
3	Male-to-male jumper wires or solid core wire	K1, T6
1	4 × AA battery holder	K1, H1
1	4 × AA batteries	
1	Battery clip	K1, H2
1	Multimeter	K1, T2

Before we get the breadboard out, let's just experiment directly with the photoresistor. Figure 3-8 shows the photoresistor connected directly to the multimeter on its 20kΩ resistance setting. It does not matter which leg of the photoresistor is connected to which multimeter lead. As you can see, the resistance of my photoresistor was 3.59kΩ. Putting my hand over the photoresistor to screen out some of the light increased that resistance to a few tens of kΩ. So, the way the photoresistor works, the more light that reaches it, the lower the resistance.

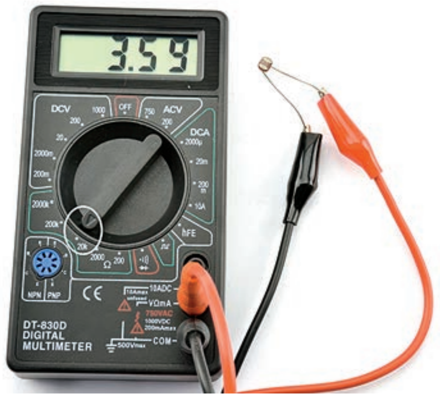


FIGURE 3-8 Measuring the LDR resistance

Microcontrollers such as the Arduino can measure voltages and do things with them, but not directly measure resistance. So to convert our photoresistor's resistance into a more easily used voltage, we can put it in a voltage divider as one of the resistors (Figure 3-9).

Note that the symbol for the photoresistor is like a resistor but with little arrows pointing to it to indicate its sensitivity to light.

We can make up this schematic on our breadboard, this time setting our multimeter to the 20V DC range and watching how the voltage changes as we cover the photoresistor to reduce the light getting to it (Figures 3-10 and 3-11).

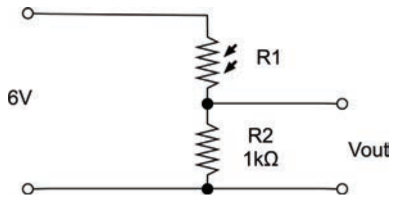


Figure 3-9 Measuring light level with an LDR and voltage divider

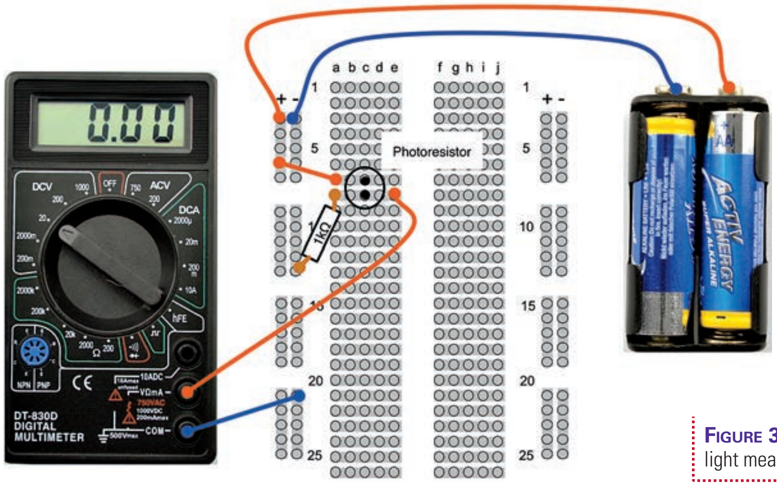


Figure 3-10 A breadboard layout for light measurement

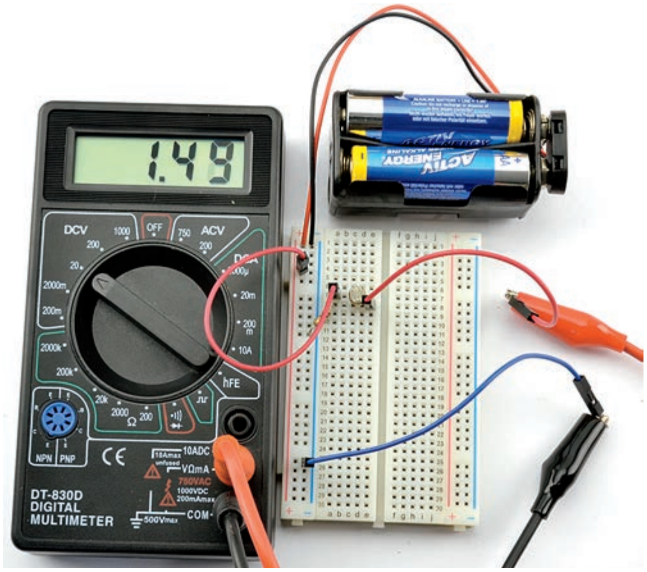


Figure 3-11 Light measurement

Hacking a Push Light to Make It Light Sensing

Battery-powered push lights are one of the many glorious bargains you are likely to find in a dollar/euro/pound store. These are intended for use in cupboards and other dark locations where a bit of extra light would be useful. Push them once and they light, push them again and they turn off.

It will not surprise you to hear that we are going to use our photoresistor to turn the light on and off. But we are also going to use a transistor.

Our approach will be to get it working on breadboard first and then solder up the design onto the push light. In fact, we will use a single LED in place of the push lamp until we know that it will work.

You Will Need

Quantity	Name	Item	Appendix Code
1	R1	Photoresistor	K1, R2
1	T1	Transistor 2N3904	K1, S1
1	R2	10kΩ resistor	K1, R5
1*	R3	270Ω resistor	K1
1*	D1	High brightness white LED	K1 or S2
1		Male-to-male jumper wire or solid core wire	T6
1		Solderless breadboard	K1, T5
1		Push light	

* These components are only needed for the breadboard experiment.

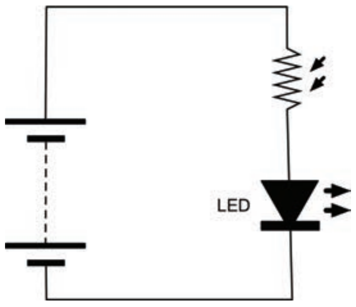


FIGURE 3-12 An LED and LDR

We want the photoresistor to control an LED, so a first thought at a circuit might be as shown in Figure 3-12.

There are two fatal flaws in this design. First, as more light falls on the photoresistor its resistance decreases, allowing more current to flow so the LED will get brighter. This is the opposite of what we want. We want the LED to come on when it's dark.

We need to use a transistor.

The basic operation of a transistor is shown in Figure 3-13. There are many different types of transistors, and probably the most common (and the type we will use) is called an NPN bipolar transistor.

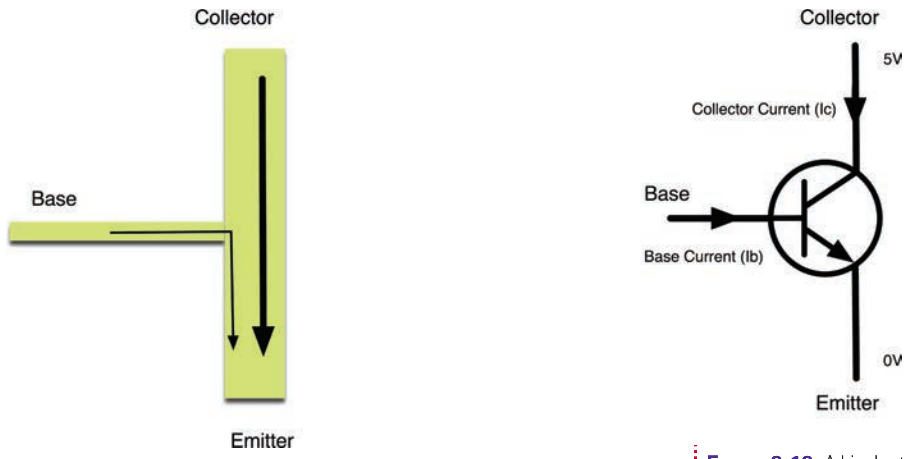


FIGURE 3-13 A bipolar transistor

This transistor has three leads: the emitter, the collector, and the base. The basic principal is that a small current flowing through the base will allow a much bigger current to flow between the collector and the emitter.

Just how much bigger the current is depends on the transistor, but it's typically a factor of 100.

Breadboard

Figure 3-14 shows the schematic diagram we will build on the breadboard. To understand this circuit, let's consider two cases.

Case 1: When It's Dark

In this case, the photoresistor R1 will have a very high resistance, so you could almost imagine that it isn't there at all. In that case, current will flow through R2, through the base and emitter of the transistor, allowing as much current as it needs to flow through R3, the LED, and T1 into its collector and out through the emitter. When enough current flows into the base of a transistor to allow current to flow from the collector to the emitter, this is called "turning on" the transistor.

We can calculate the base current using Ohm's law. In this situation, the base of the transistor will be at only about half a volt, so we can assume there is more or less the full 6V across the 10kΩ resistor R2. Since I = V / R, the current will be 6 / 10,000 A or 0.6mA.

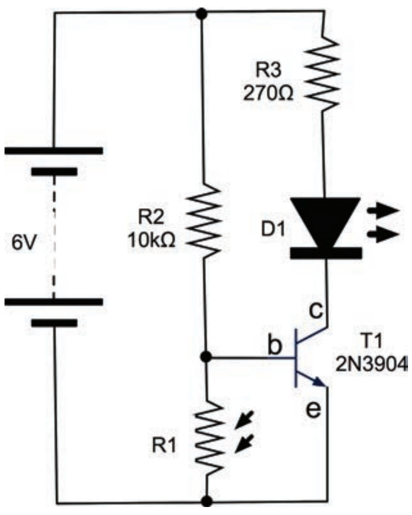


FIGURE 3-14 Using an LDR and transistor to switch an LED

Case 2: When It's Light

When it is light, we have to consider the resistance of the photoresistor R1. The lighter it is, the lower the resistance of R1 and the more of the current otherwise destined for the base of the transistor will be diverted through R1, preventing the transistor from turning on.

I think the time has come to build the project on breadboard. Figure 3-15 shows the breadboard layout, and Figures 3-16a and 3-16b the finished breadboard.

When placing the LED on the breadboard, make sure you get it the right way around. The longer lead is the positive lead, and it should be on row 10 connected to R3. (See Figure 3-16a.)

If everything is fine, you should find that when you cover the photoresistor, the LED should light (Figure 3-16b).

Construction

Now that we have proved our circuit works, we can get on with modding the push light. Figure 3-17 shows the push light the author used. Unless you are very lucky, yours is likely to be

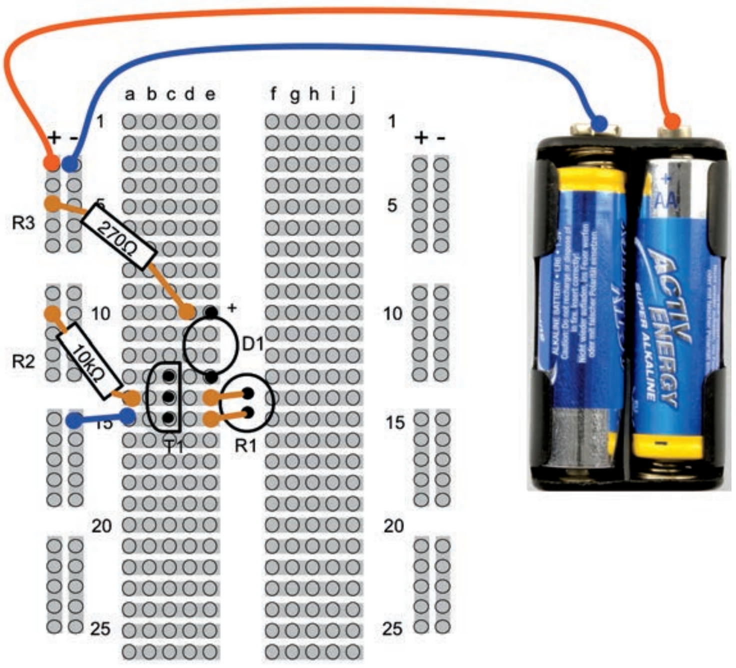


FIGURE 3-15 The light switch breadboard layout

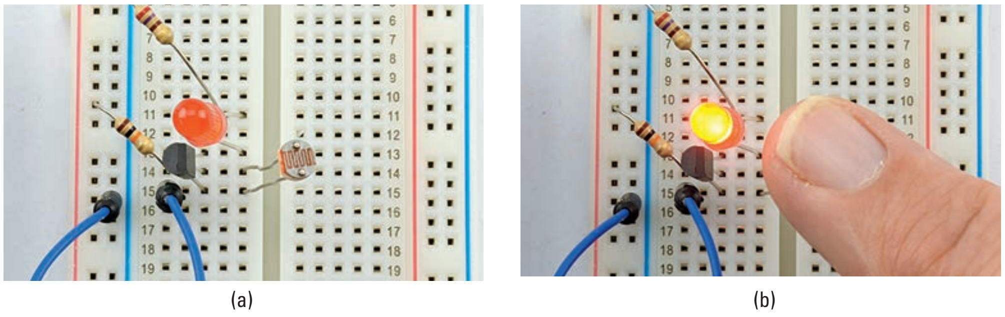


FIGURE 3-16 The light switch breadboard

different, so read through the following sections carefully and you should be able to work out how to change your light. To make life easy for yourself, try and find a push light that operates from 6V (4 AA or AAA cells).

You will probably find screws on the back of the push light. Remove these and put them somewhere safe. The inside of the push light is shown in Figure 3-18. The various connections on the light are marked. You can find the corresponding connections on your light using a multimeter.

Setting the multimeter to its 20V DC range will let you determine which battery lead is positive and which is negative. Looking at the wiring, we can draw a schematic diagram for the light as it stands, before we start altering it (Figure 3-19).

FIGURE 3-17 A push light

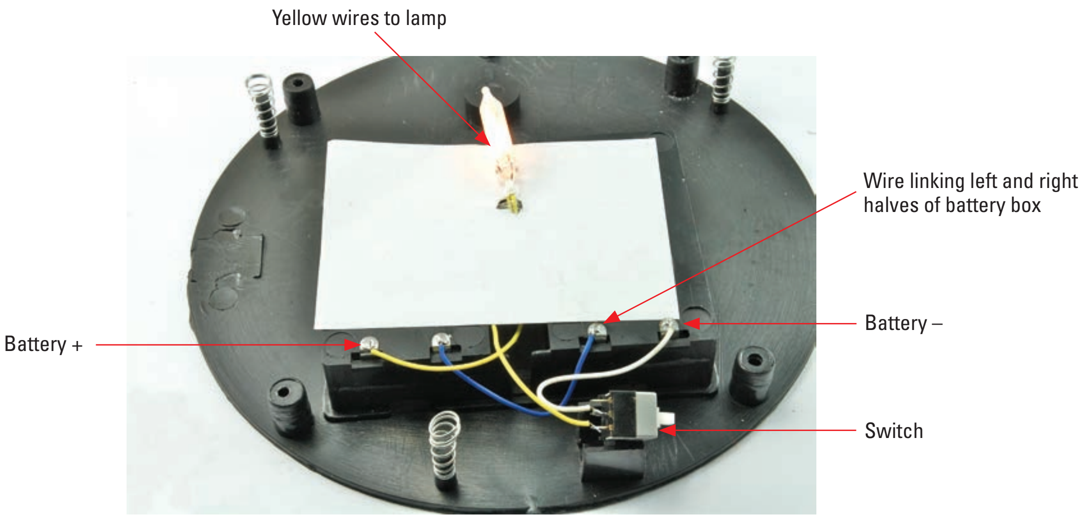


FIGURE 3-18 Inside the push light

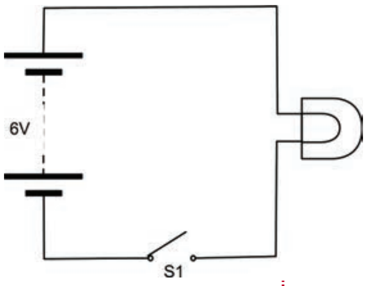


FIGURE 3-19 The schematic diagram for the original push light

This light uses an old-fashioned incandescent bulb. We will replace that with a high-brightness LED. If you don't have one of these, a regular LED of the color of your choice will work, but not be very bright.

Figure 3-20 shows how we replaced the bulb with the LED and the 270Ω resistor. Make sure the longer positive lead of the LED is connected to the resistor and the far side of that resistor is connected to the positive terminal of the battery.

Try pressing the switch to make sure the LED is working.

We can now draw a schematic that combines what we have in the existing lamp and our light-detecting circuit (Figure 3-21).

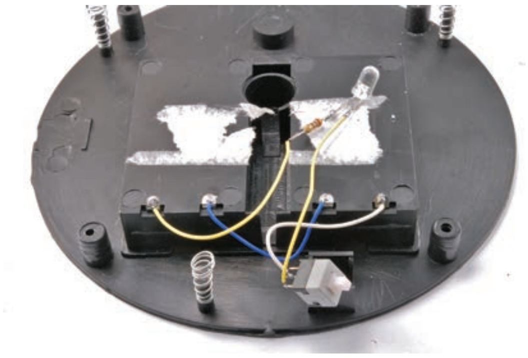


FIGURE 3-20 Replacing the bulb with an LED and a resistor

In fact, all this really amounts to is adding in the switch to the original LED schematic. We have already installed R3 and D1 when we replaced the bulb with an LED. The switch is already there, so all we need to add is the transistor, photoresistor, and R2. Figure 3-22 shows how we will rewire the push light.

Figure 3-23 shows the sequence of steps in soldering the extra components onto the light. Make sure you take the batteries out before you start soldering.

1. Start by desoldering the lead from the switch that isn't connected to the negative battery terminal (Figure 3-23a).
2. Solder the 10kΩ resistor R2 between the middle lead of the transistor (the base) and the positive terminal on the battery box.
3. With the flat of the transistor facing upward, as shown in the diagram, connect the left-hand lead of the transistor to the wire you just disconnected from the switch (Figure 3-23b).
4. Solder the photoresistor between the left and middle pins of the transistor, and connect the combined left-hand transistor lead and photoresistor lead to the connection on the switch that the wire used to be attached to. (See Figure 3-23c.)
5. Tuck the components away neatly, bending the leads to make sure there is no way the bare leads can touch each other. (See Figure 3-23d.)

There you are! You have hacked some electronics.

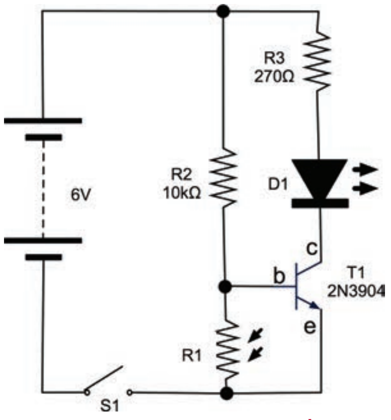


FIGURE 3-21 The final schematic

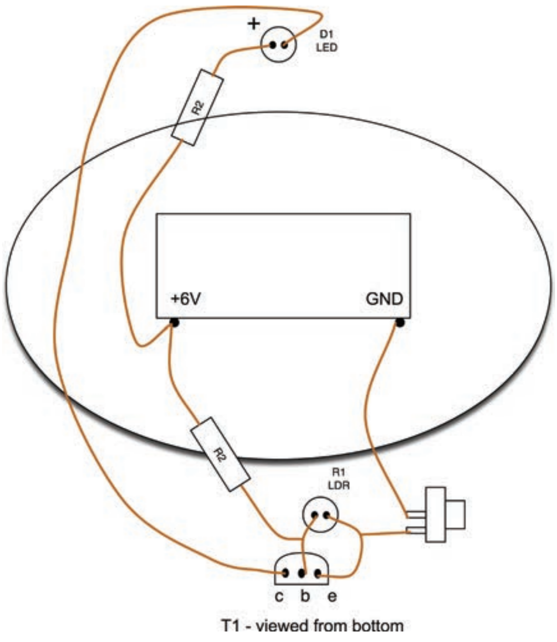
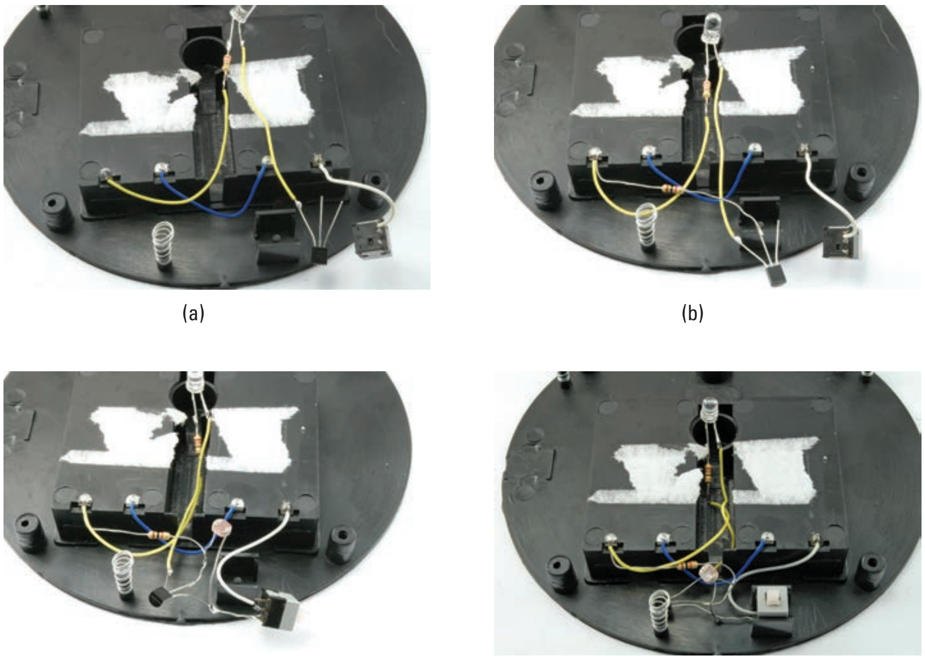


Figure 3-22 The push light wiring diagram

Figure 3-23 Soldering the project



MOSFET Transistors

The 2N3904 that you used in the previous project is called a bipolar transistor. It's basically a device that amplifies current. A small current into the base controls a much bigger current flowing through the collector. Sometimes, the current gain of just 100 or so is not nearly enough.

There is another type of transistor that does not suffer from this limitation called the MOSFET (Metal Oxide Semiconductor Field Effect Transistor). You can see why it gets shortened to MOSFET. These transistors are controlled by voltage rather than current and make very good switches.

MOSFETs do not have emitters, bases, and collectors, they have "sources," "gates," and "drains." They turn on when the gate voltage passes a threshold, usually about 2V. Once on, quite large currents can flow through the "drain" to the "source" rather like a bipolar transistor. But since the gate is isolated from the rest of the transistor by a layer of insulating glass, hardly any current flows into the gate. It is the voltage at the gate that determines what current will flow.

We will meet MOSFETs again later in the section "Using a Power MOSFET to Control a Motor."

PNP and P-Channel Transistors

The automated light switch of the previous section switched on the "negative side of the load." That is, if you go back to Figure 3-21, you can see that the resistor and LED that make up the light are not connected to GND except through the transistor. If for some reason (and this does happen) we wanted to switch the positive side, then we would need to use a PNP equivalent of the NPN 2N3904, such as the 2N3906. NPN stands for Negative-Positive-Negative, and yes, you can guess what PNP stands for. That is because transistors are kind of semiconductor sandwiches, with material of either N or P type as the bread. If the bread is N type (the most common), then the base voltage needs to be higher than the emitter voltage (by about 0.5V) before the transistor starts to turn on. On the other hand, a PNP transistor turns on when the base voltage is more than 0.5V lower than the emitter voltage.

If we wanted to switch the positive side, we could use a PNP transistor (as shown in the PNP alternative to Figure 3-21) displayed in Figure 3-24.

MOSFETs also have their own equivalent of PNP transistors called P-channel, their version of the more common NPN being called N-channel.

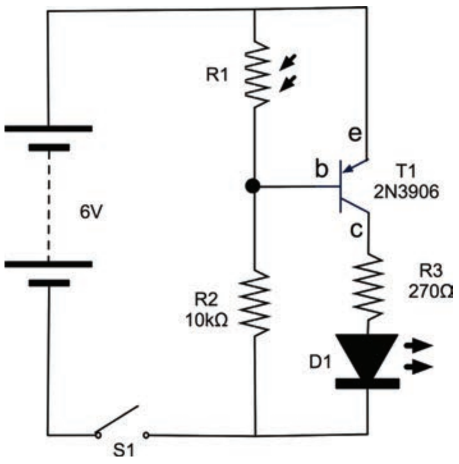


FIGURE 3-24 Using a PNP bipolar transistor

Common Transistors

The transistors in Table 3-1 will cover a wide range of transistor applications. There are thousands and thousands of other transistors, but in this book we only really use them for switching, so these will cover most "bases"!

Using a Power MOSFET to Control a Motor

Figure 3-25 shows the schematic symbol and the pinout for the FQP30N06L N-channel MOSFET.

This MOSFET is capable of controlling loads of up to 30A. We are not going to push it any way near that far, we are just going to use it to control the power to a small electric motor that might have a peak load of 1 or 2A. While this would be too much for the bipolar transistors that we have been using so far, this MOSFET will hardly notice!

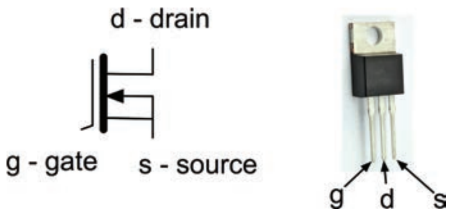


FIGURE 3-25 The FQP30N06L N-channel MOSFET

Name	Appendix Code	Type	Max Switching Current	Notes
Low/medium-current switching				
2N3904	S1	NPN bipolar	200mA	Current gain about 100
2N3906	S4	PNP bipolar	200mA	Current gain about 100
2N7000	S3	N-channel MOSFET	200mA	2.1V gate-source threshold voltage; turns on when gate is 2.1V higher than source
High-current switching				
FQP30N06L	S6	N-channel MOSFET	30A	2.0V gate-source threshold voltage; turns on when gate is 2.0V higher than source

TABLE 3-1 Really Useful Transistors

You Will Need

To try out this high-power MOSFET, you will need the following items.

Quantity	Item	Appendix Code
1	Solderless breadboard	K1, T5
4	Male-to-male jumper wire or solid core wire	K1, T6
1	4 × AA battery holder	K1, H1
1	4 × AA batteries	
1	Battery clip	K1, H2
1	Multimeter	K1, T2
1	10kΩ trimpot	K1
1	FQP30N06L MOSFET Transistor	K1, S6
1	6V DC motor or gearmotor	K1, H6

The DC motor can be any small motor you can find that is around 6V. A motor rated at 12V should still turn at 6V. To test it, just connect its terminals directly to the 6V battery.

Breadboard

The schematic diagram for what we will make is shown in Figure 3-26.

The variable resistor will control the voltage at the gate of the MOSFET. When that gate voltage exceeds the gate threshold, the transistor will turn on and the motor will start.

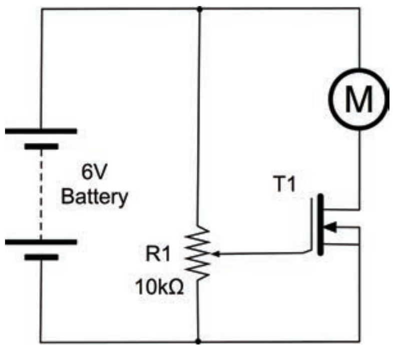


FIGURE 3-26 A schematic for the MOSFET experiment

The breadboard layout for the project and a photograph of the experiment in action are shown in Figure 3-27 and Figure 3-28.

To connect the motor to the breadboard, you will probably need to solder a pair of leads to it. It does not matter which way around you connect the motor. The polarity just determines which direction the motor turns. So if you swap the motor leads over, it will turn in the opposite direction.

Try turning the knob on the variable resistor. You will see that you do not have a great deal of control over the speed of the motor. If you hover around the threshold voltage, you can

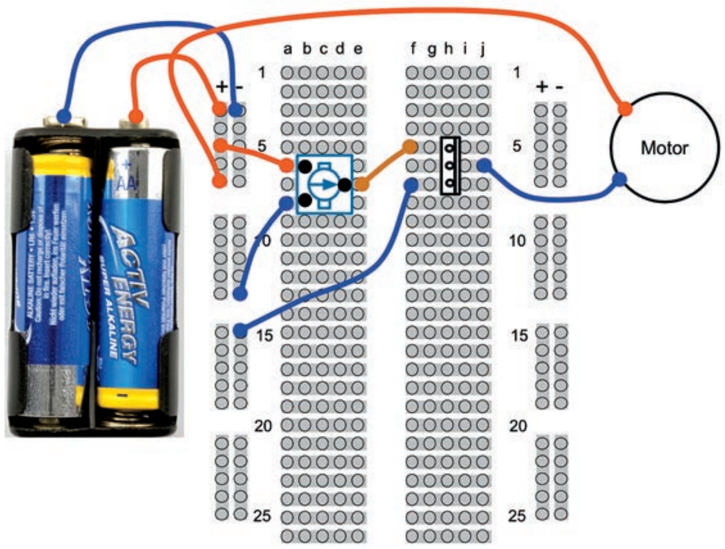


FIGURE 3-27 The breadboard layout for the MOSFET experiment

control the motor speed, but you can probably see why the MOSFET is most commonly used as a switch that is either on or off.

This kind of MOSFET is called a logic level MOSFET, because its gate voltage is low enough to be controlled directly by digital output pins on a microcontroller. This is not true of all MOSFETs. Some have gate threshold voltages of 6V or more.

In Chapter 7, you will use a MOSFET to finely control the motor speed.

To prove to yourself that it really is voltage that controls the MOSFET and not current, try this little experiment:

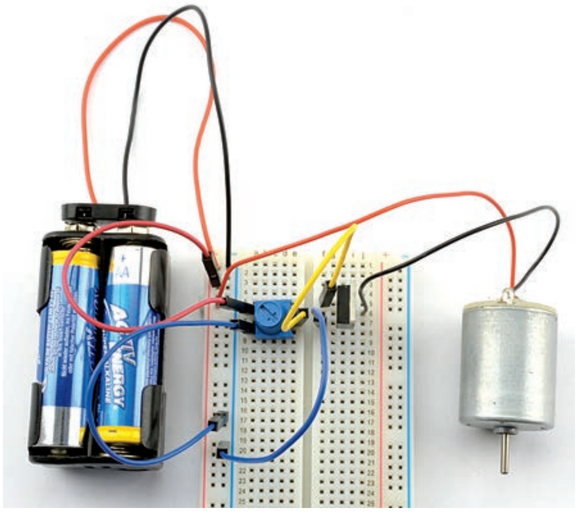


FIGURE 3-28 The MOSFET experiment

- Disconnect the wire where it connects to the middle connection of the variable resistor. This gives you a lead that connects to the MOSFET's gate.
- Disconnect the other two lead ends that go to the variable resistor so that you have bare wires connected to 6V and GND.

- Touch the bare metal of the wire connected to the gate with one finger and the wire to 6V with another finger. The motor should turn on.
- Now repeat the experiment, but using fingers on the GND and gate wires and the motor should go off.

You will probably find that once set on, the MOSFET will stay on, for a while, even with the gate disconnected. This is because the gate acts like a capacitor, holding the voltage it is set to. After a while this will leak away and the MOSFET will turn off again.

Selecting a Switch

On the face of it, a switch is a very simple thing. It closes two contacts, making a connection. Often, that is all you need, but other times you will require something more complicated. For example, let's say you want to switch two things at the same time.

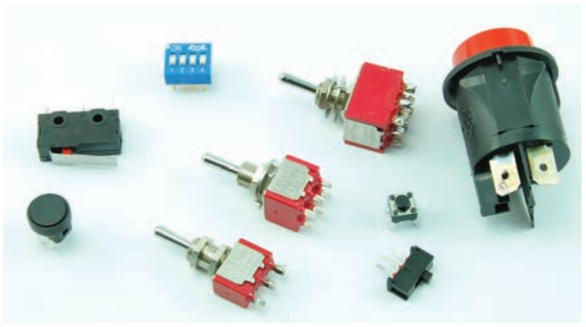


FIGURE 3-29 Switches

There are also switches that only make the contact while you are pressing them, or ones that latch in one position. Switches may be push button, toggle, or rotary. There are many options to choose from and in this section we will attempt to explain the options.

Figure 3-29 shows a selection of switches.

Push-Button Switches

Where so many things use a microcontroller, a simple push switch is probably the most common type of switch (Figure 3-30).

This kind of switch is designed to be soldered directly onto a circuit board. It will also fit onto our breadboard, which makes it quite handy.

The confusing thing about this switch is that it has four connections where you would only expect there to be two. Looking at Figure 3-30, you can see that connections B and C are always connected together, as are A and D. However, when the button is pressed, all four pins are all connected together.

This does mean that you need to be careful to find the right pins or your switch will be connected all the time.

If there is any doubt about how the switch works, use your multimeter set to Continuity mode to work out what is connected to what—first without the switch pressed and then with the switch pressed.

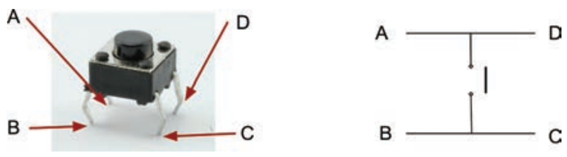


FIGURE 3-30 A push switch

Microswitches

A microswitch is another type of handy switch. They are not designed to be pressed directly, but are often used in things like microwave ovens to detect that the door is closed, or as anti-tamper switches that detect when the cover is removed from an intruder alarm box.

Figure 3-31 shows a microswitch—with three pins!

The reason a microswitch has three pins rather than just two is that it is what is known as a "double throw" or "change-over" switch. In other words, there is one common connection C and two other connections. The common connection will always be connected to one of those contacts, but never both at the same time. The normally open (n.o.) connection is only closed when the button is pressed; however, the normally closed (n.c.) connection is normally closed, and only opens when the button is released.

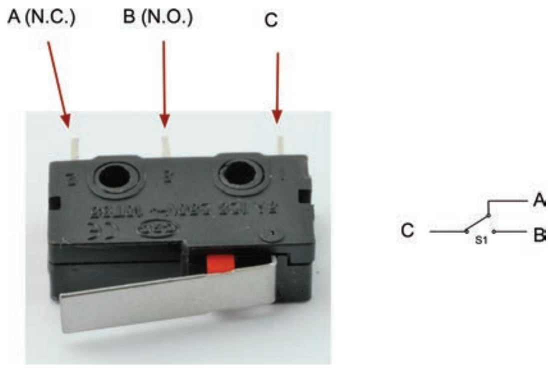


FIGURE 3-31 A microswitch

If you have one of these switches, you might like to connect your multimeter to it. Attach one lead to the common connection and use it to find the n.c. connection, then press the button and the beep should stop.

Toggle Switches

If you look through a component catalog (which every good electronics hacker should), you will find a bewildering array of toggle switches. Some will be described as DPDT, SPDT, SPST, or SPST, momentary on, and so forth.

Let's untangle some of this jargon, with a key for these cryptic letters:

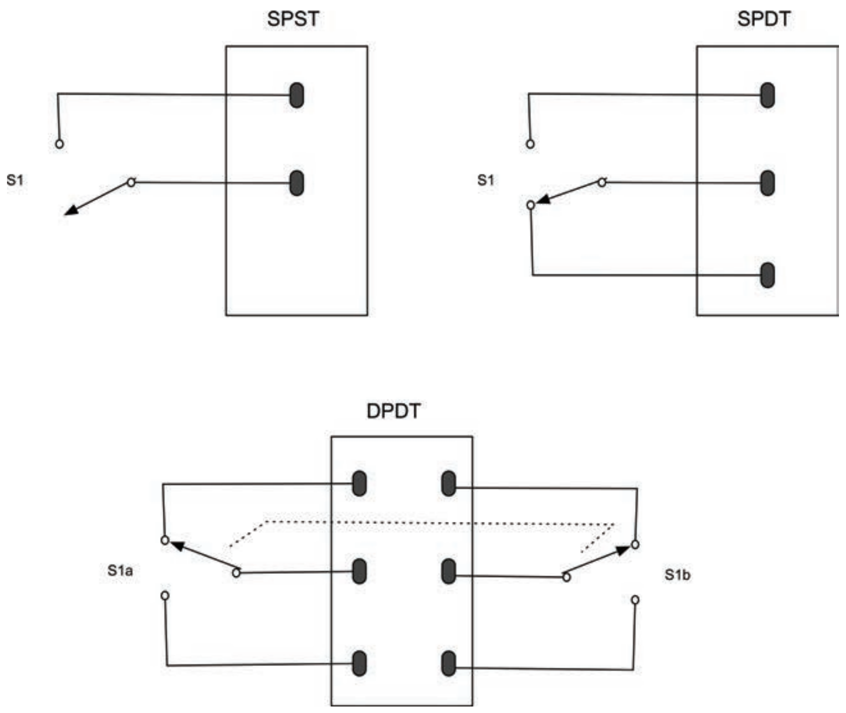
- D = Double
- S = Single
- P = Pole
- T = Throw

So, a DPDT switch is double pole, double throw. The word "pole" refers to the number of separate switch contacts that are controlled from the one mechanical lever. So, a double pole switch can switch two things on and off independently. A single throw switch can only open or close a contact (or two contacts if it is double pole). However, a double throw switch can make the common contact be connected to one of two other contacts. So, a microswitch is a double throw switch because it has both normally closed and normally open contacts.

Figure 3-32 summarizes this.

Notice in Figure 3-32 that when drawing a schematic with a double pole switch, it is normal to draw the switch as two switches (S1a and S1b) and connect them with a dotted line to show they are linked mechanically.

FIGURE 3-32 Toggle switches—poles and throws



The matter is further complicated because you can have three poles or even more on a switch, and double throw switches are sometimes sprung, so they do not stay in one or both of these positions. They may also have a center-off position where the common contact is not connected to anything.

You might see a switch described as "DPDT, On-Off-Mom." Well, we know what the DPDT bit means. It will have six legs for a start. The "On-Off-Mom" part means that it also has a center position, where the common connection is not made to anything. Switch it one way and it will be on to one set of contacts and stay in that position. Switch it the other way and it will be sprung to return to the central position, allowing you to make a "momentary" connection.

A lot of this terminology applies to other kinds of switches in addition to toggle switches.

Summary

We now know a bit about voltage, current resistance, and power. In the next chapter, we will use these ideas in looking at how to use LEDs.

4

LEDs

LEDs (light-emitting diodes) are diodes that emit light when a current passes through them. Well on the way to completely replacing filament light bulbs in almost all applications, they can be used as indicators and, with the very high brightness types of LED, can provide illumination.

They are much more efficient than conventional light bulbs, producing far more light per watt of power.

LEDs do, however, require a little more thought when used. They have to be powered with the correct polarity and require circuitry to limit the current flowing through them.

Preventing an LED from Burning Out

LEDs are delicate little things and quite easy to destroy accidentally. One of the quickest ways of destroying an LED is to attach it to a battery without using a resistor to limit the current.

To get to grips with LEDs, we will put three different color LEDs on our breadboard (Figure 4-1).

You Will Need

Quantity	Names	Item	Appendix Code
1		Solderless breadboard	K1, T5
1	D1	Red LED	K1
1	D2	Yellow LED	K1
1	D3	Green LED	K1
3	R1, R2, R3	270Ω resistor	K1
3		Male-to-male jumper wire or solid core wire	K1, T6
1		4 × AA battery holder	K1, H1
1		Battery clip	K1, H2
4		AA batteries	

Diodes

We need to understand LEDs a little better if we are to successfully use them. LED stands for "light-emitting diode," so let's start by looking at what a diode is (Figure 4-2).

A diode is a component that only lets current flow in one direction. It has two leads, one called the anode, and the other called the cathode. If the anode is at a higher voltage than the cathode (it has to be greater by about half a volt), then it will conduct electricity and is said to be "forward-biased." If on the other hand the anode isn't at least half a volt higher than the cathode, it is said to be "reverse-biased" and no current flows.

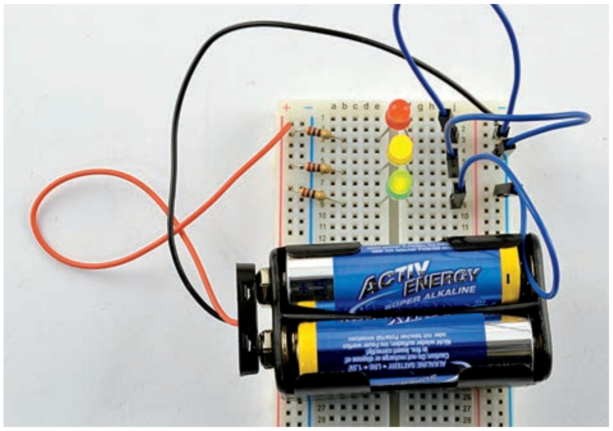


FIGURE 4-1 LEDs on a breadboard

LEDs

An LED is just like a regular diode except that when it is forward-biased, it conducts and generates light. It also differs from a regular diode in that the anode usually needs to be at least 2V higher than the cathode for it to produce light.

Figure 4-3 shows the schematic diagram for driving an LED.

The key to this circuit is to use a resistor to limit the current flowing through the LED. A normal red LED will typically just be lit at about 1mA and is designed to be used at around 10 to 20mA (this is called the "forward current" or I_F). We will aim for 15mA for our LED. We can also assume that when it is conducting, there will be about 2V across it. This is called the "forward voltage" or V_F. That means there will be 6 – 2 = 4V across the resistor.

So, we have a resistor that has a current flowing through it (and the LED) of 15mA and a voltage across it of 4V. We can use Ohm's law to calculate the value of resistance we need to achieve this:

$$R = V / I = 4V / 0.015A = 267\Omega$$

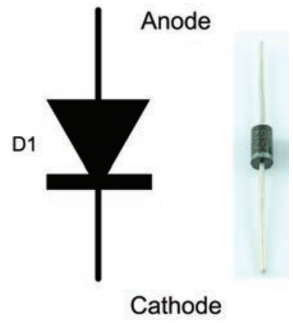


FIGURE 4-2 A diode

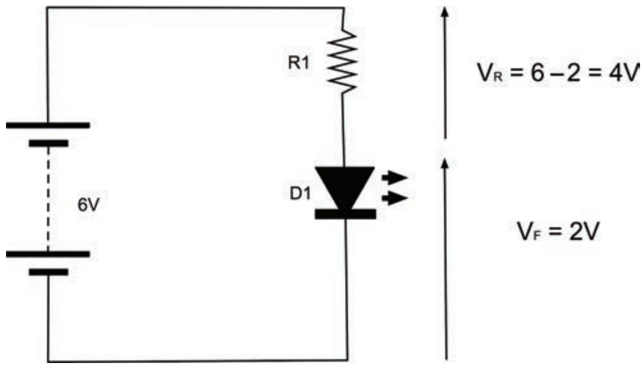


FIGURE 4-3 Limiting current to an LED

Parameter	Red	Green	Yellow	Orange	Blue	Units
Maximum forward current (I_F)	25	25	25	25	30	mA
Typical forward voltage (V_F)	1.7	2.1	2.1	2.1	3.6	V
Maximum forward voltage	2	3	3	3	4	V
Maximum reverse voltage	3	5	5	5	5	V

TABLE 4-1 An LED Datasheet

Resistors come in standard values, and the nearest higher value in our resistor starter kit is 270Ω.

As I mentioned earlier, a red LED will almost always light quite brightly with something like 10–20mA. The exact current is not critical. It needs to be high enough to make the LED light, but not exceed the maximum forward current of the LED (for a small red LED, typically 25mA).

Table 4-1 shows a section of the datasheet for a typical range of LEDs of different colors. Note how V_F changes for different color LEDs.

The other parameter you should be aware of is the "maximum reverse voltage." If you exceed this by, say, wiring your LED the wrong way around, it is likely to break the LED.

Many online series resistor calculators are available that—given the supply voltage V_F and current I_F for your LED—will calculate the series resistor for you. For example:

www.electronics2000.co.uk/calc/led-series-resistor-calculator.php

Table 4-2 is a useful rough guide, assuming a forward current of around 15mA.

Supply Voltage (V)	Red	Green, Yellow, Orange	Blue
3	91Ω	60Ω	none
5	220Ω	180Ω	91Ω
6	270Ω / 330Ω	220Ω	180Ω
9	470Ω	470Ω	360Ω
12	680Ω	660Ω	560Ω

TABLE 4-2 Series Resistors for LEDs

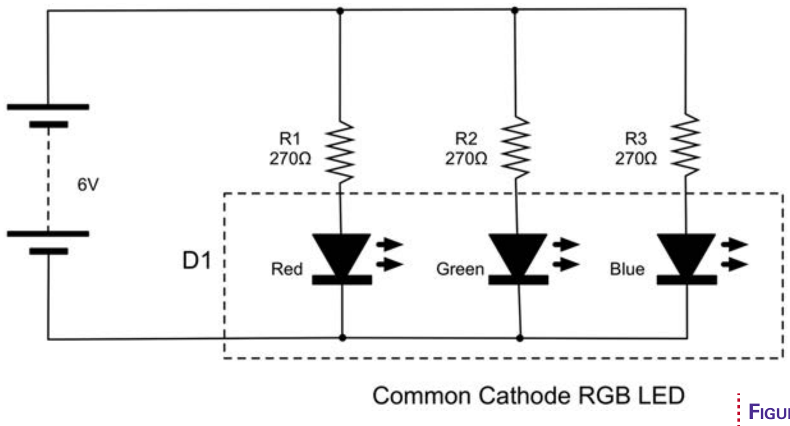


FIGURE 4-4 An LED's schematic

Trying It Out

You might like to try out your LEDs and get them lit up on the breadboard. So, using Figures 4-4 and 4-5 as a guide, wire up your breadboard. Remember that the longer lead of the LED is normally the anode (positive) and thus should be to the left of the breadboard.

An important point to notice here is that each LED has its own series resistor. It is tempting to use one lower value current limiting resistor and put the LEDs in parallel, but don't do this. If you do, the LED with the lowest V_F will hog all the current and probably burn out, at which point the LED with the next lowest V_F will do the same, until all the LEDs are dead.

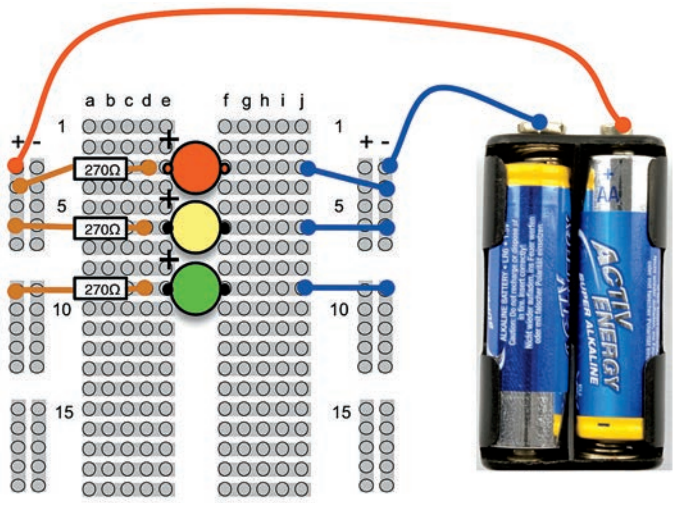


FIGURE 4-5 An LED breadboard layout

Selecting the Right LED for the Job

LEDs come in all colors, shapes, and sizes. Many times, you just want a little indicator light, in which case a standard red LED is usually fine. However, there are many other options, including LEDs bright enough to be used as lamps.

Brightness and Angle

When selecting an LED, they may simply be described as “standard” or “high brightness” or “ultra-bright.” These terms are subjective and open to abuse by unscrupulous vendors. What you really want to know is the LED’s luminous intensity, which is how much light the LED produces. You also want to know the angle over which the LED spreads the light.

So, for a flashlight, you would use LEDs with a high luminous intensity and a narrow angle. Whereas for an indicator light to show that your gadget is turned on, you would probably use an LED with a lower luminous intensity but a wider angle.

Luminous intensity is measured in millicandela or mcd, and a standard indicator type LED will typically be around 10 to 100 mcd, with a fairly wide viewing angle being 50 degrees. A “high brightness” LED might be up to 2000 or 3000 mcd, and an ultra-bright, anything up to 20,000 mcd. A narrow beam LED is about 20 degrees.

Multicolor

We have already explored the more common LED colors, but you can also get LED packages that actually contain two or three LEDs of different colors in the same package. Common varieties are red/green as well as full-color RGB (Red Green Blue). By varying the proportion of each color, you can change the color of the light produced by the LED package (Figure 4-6).

In the section “Experimenting with RGB LEDs,” you will get to try out an LED that combines red, green, and blue LEDs into a single package.

IR and UV

As well as visible LEDs, you can also buy LEDs whose light is invisible. This is not as pointless as you might think. Infrared LEDs are used in TV remote controls, and ultraviolet LEDs are

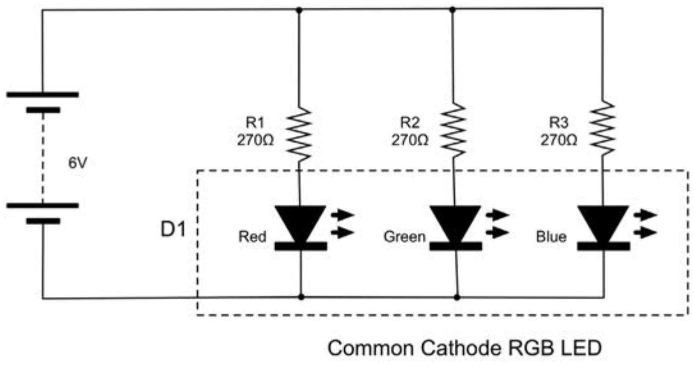


Figure 4-6 An RGB LED test schematic

used in specialist applications such as checking the authenticity of bank notes and making people's white clothes light up in clubs.

Use these LEDs just like any other LED. They will have a recommended forward current and voltage and will need a series resistor. Of course, checking that they are working is trickier. Digital cameras are often a little sensitive to infrared and you may see a red glow on the screen.

LEDs for Illumination

LEDs are also finding their way into general household lighting. This has come about because of improvements in LED technology that have produced LEDs with a brightness comparable to incandescent light. Figure 4-7 shows one such high-brightness LED. In this case, it's a 1W LED, although 3W and 5W LED modules are also available.

The cool-looking star shape is thanks to the aluminum heat sink that the LED is attached to. At full power, these LEDs produce enough heat to warrant such a heat sink to disperse the heat into the air.

These LEDs can use a resistor to limit the current, but a quick calculation will show you that you will need quite a high-power resistor. A better approach to using these LEDs is to use a constant current driver, which you will investigate after the next section.

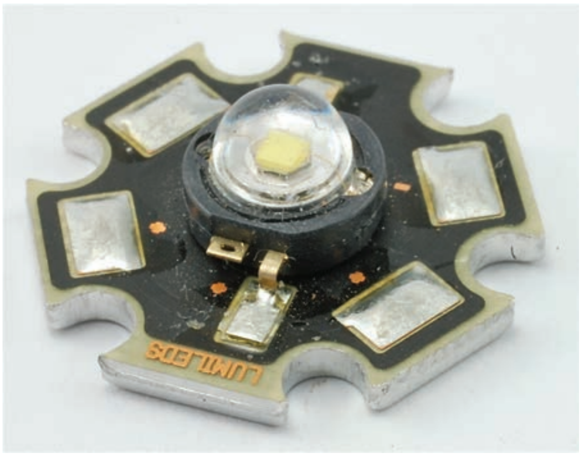


Figure 4-7 A high-power LED

Experimenting with RGB LEDs

You can experiment with an RGB LED, mixing up different colored light on breadboard.

You Will Need

Quantity	Names	Item	Appendix Code
1		Solderless breadboard	K1, T5
1	D1	RGB common cathode LED	K1, S4
3	R1–R3	270Ω resistor	K1
1		Male-to-male jumper wire or solid core wire	K1, T6
1		4 × AA battery holder	H1
1		Battery clip	K1, H2
4		AA batteries	

Build up the breadboard as shown in Figure 4-8 and power up the circuit (Figure 4-9). The LED should be producing light that is more or less white. This is a mixture of red, green, and blue light. Now try pulling out the resistors one at a time and noticing how the color changes. Note that the common cathode lead of the LED is the longest lead and is connected to GND.

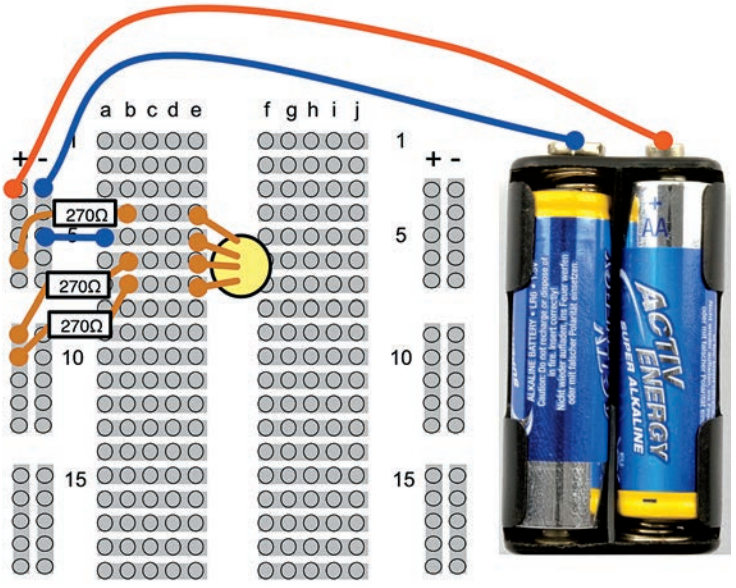


FIGURE 4-8 An RGB LED test breadboard layout

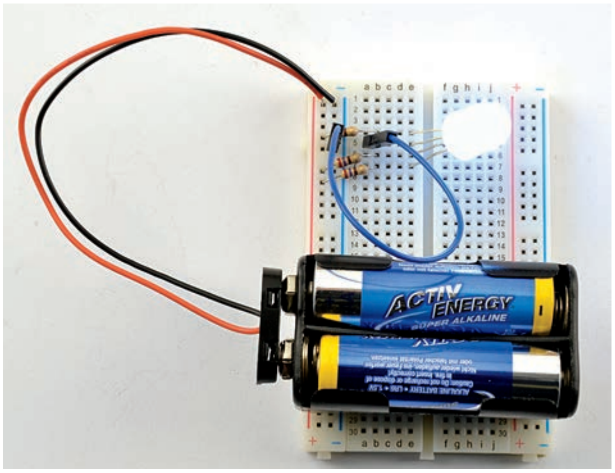


Figure 4-9 An RGB test

Making a Constant Current Driver

Using a resistor to limit the current is all right for small LEDs. For high-power LEDs, you can use a series resistor (it will need to be quite high power), but a better way is to use a constant current driver.

As the name suggests, the constant current driver will supply the same current whatever voltage it is supplied with and whatever the forward voltage of the LED. You just set the current and that is how much current will flow through the high-power LED.

A very useful IC that is often used for this purpose is the LM317. This IC is primarily intended as an adjustable voltage regulator, but can easily be adapted for use in regulating current.

This project will start off on breadboard and then we will cut the top off a battery clip and solder the LM317 and resistor to it to make an emergency 1W LED light.

You Will Need

Quantity	Names	Item	Appendix Code
1		Solderless breadboard	K1, T5
1	D1	1W white Lumiled LED	S3
1		LM317 voltage regulator IC	S13
3	R1	4.7Ω resistor (0.5W)	R3

Quantity	Names	Item	Appendix Code
1		Battery clip (to destroy)	K1, H2
1		PP3 9V battery	
		Solid core wire	K1, T9

Design

Figure 4-10 shows the schematic diagram for regulating the current to a high-power LED like the one shown in Figure 4-9.

The LM317 is very easy to use in a constant current mode. It will always strive to keep its output voltage at exactly 1.25V above whatever voltage the Adj (adjust) pin is at.

The LED we are going to use is a 1W white light LED. It has an I_f (forward current) of 300mA and a V_f (forward voltage) of 3.4V.

The formula for calculating the right value for R1 for use with the LM317 is

$$R = 1.25V / I$$

so in this case, R = 1.25 / 0.3 = 4.2Ω

If we used a standard resistor value of 4.7Ω, then this would reduce the current to:

$$I = 1.25V / 4.7\Omega = 266 \text{ mA}$$

Checking the power rating for the resistor, the LM317 will always have 1.25V between Out and Adj. So:

$$P = V \times I = 1.25V \times 266mA = 0.33W$$

A half-watt resistor will therefore be fine.

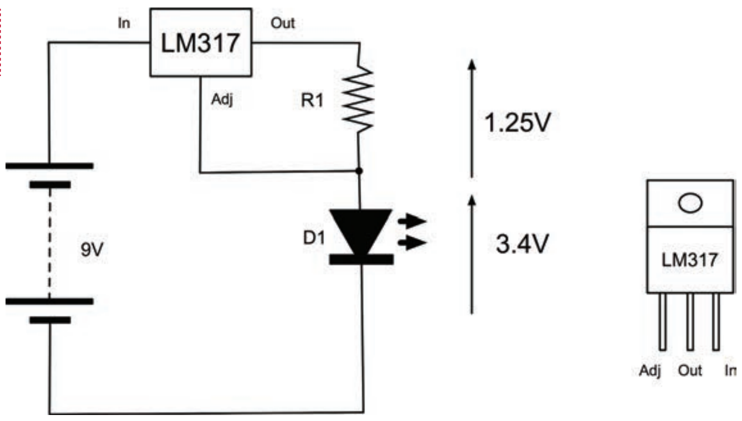


FIGURE 4-10 An LM317 constant current LED driver schematic

The LM317 also needs its input to be about 3V higher than its output to guarantee 1.25V between Adj and the output. This means that a 6V battery would not be quite high enough because the forward voltage is 3.4V. However, we could drive the circuit using a 9V battery or even a 12V power supply without modification, since whatever the input voltage, the current will always be limited to about 260mA.

A quick calculation of the power consumed by the LM317 will reassure us that we are not going to come near exceeding its maximum power rating.

For a 9V battery, the voltage between In and Out will be $9 - (1.25 + 3.4) = 4.35\text{V}$. The current is 260mA, so the power is: $4.35 \times 0.26 = 1.13\text{W}$.

According to its data sheet, the maximum power handling capability of the LM317 is 20W, and it can cope with a current of up to 2.2A for a supply voltage of less than 15V. So we are fine.

Breadboard

Figure 4-11 shows the breadboard layout for this, and Figure 4-12 displays the actual breadboard. These LEDs are almost painfully bright, so avoid staring at them. When working with them, I cover them with a sheet of paper so I can see when they are on without being temporarily blinded!

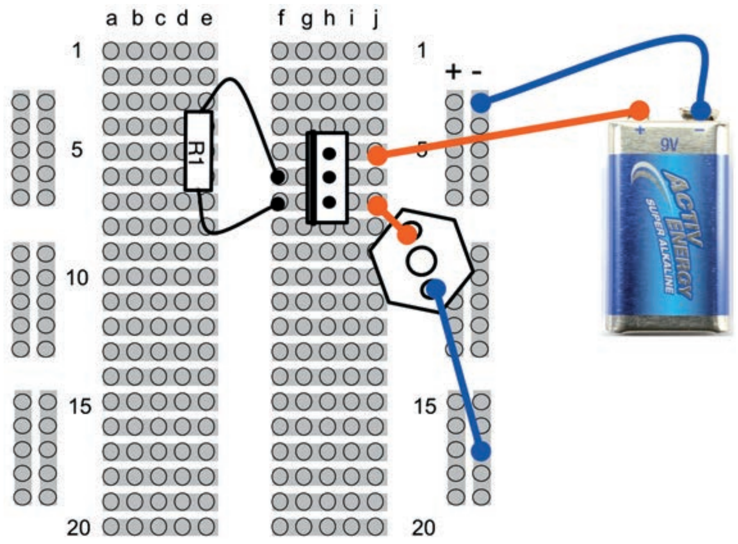


FIGURE 4-11 The LED constant current driver breadboard layout

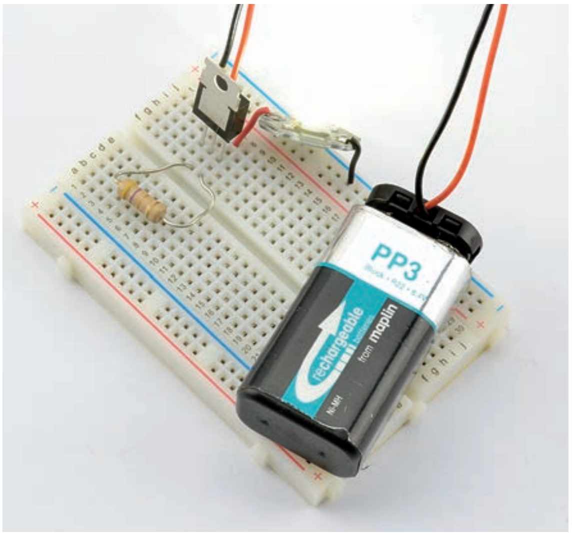


FIGURE 4-12 The LED constant current driver

You will need to solder lengths of solid-core wire to the LED's terminals so it can plug into the breadboard. It is a good idea to leave the insulation on so there is no chance of the bare wires touching the heat sink and shorting. Using different colors of wire insulation also allows you to tell which lead is positive.

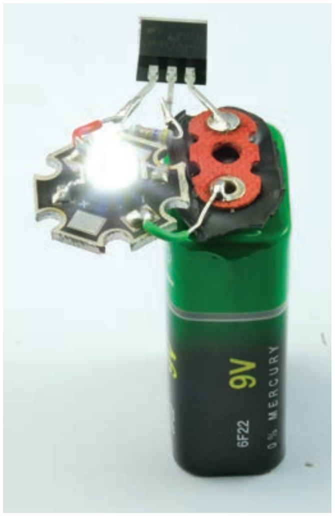


FIGURE 4-13 Emergency LED lighting

Construction

We will use this to make a little emergency lantern by hacking a battery clip to build the electronics on top of it so it can be clipped on top of a PP3 battery in the event of a power failure (Figure 4-13).

Figures 4-14a through 4-14d show the stages involved in soldering this up.

First, remove the plastic from the back of the battery clip using a craft knife. Then, unsolder the exposed leads (Figure 4-14a).

The next step (Figure 4-14b) is to solder the Input lead of the LM317 to the positive terminal of the battery clip. Remember that the positive connector on the clip will be the opposite of the connector on the battery itself, so the positive connector on the clip is the socket-shaped connector. Gently bend the leads of the LM317 apart a little to make this easier.

Now solder the LED in place making sure the cathode of the LED goes to the negative connection on the battery clip (Figure 4-14c).

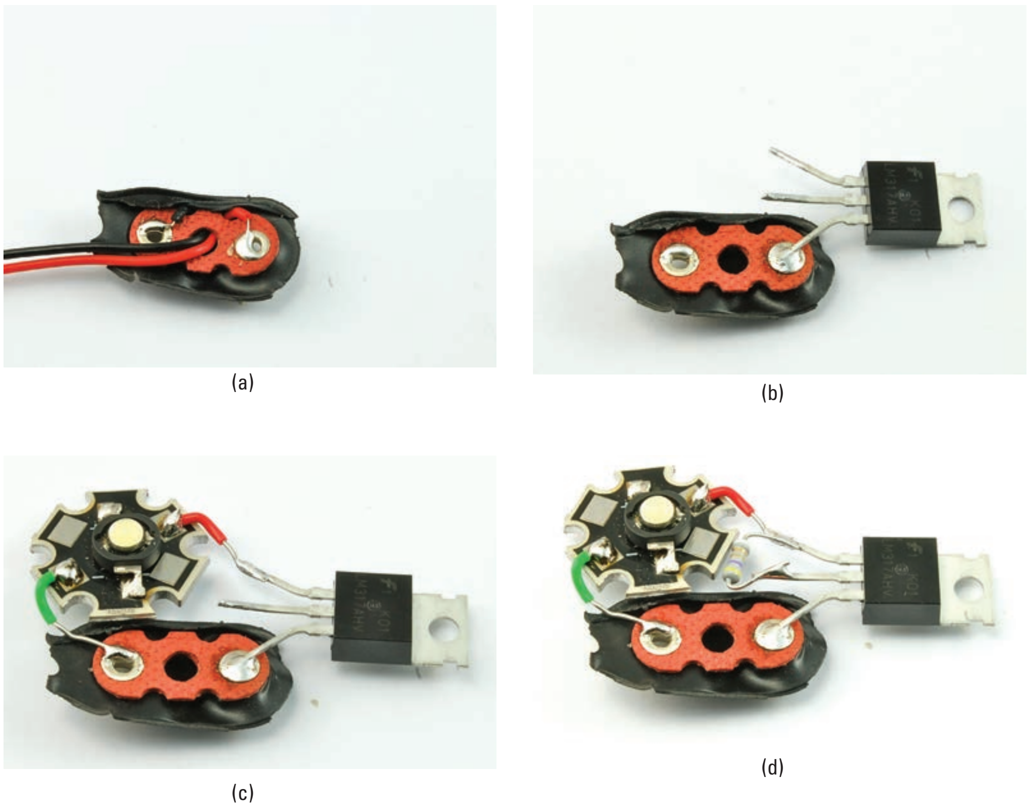


Figure 4-14 Making an emergency 1W LED light

Finally, solder the resistor across the two topmost leads of the LM317 (Figure 4-14d).

Powering Large Numbers of LEDs

If you use something like a 12V power supply, you can put a number of LEDs in series, with just one LED. In fact, if you know the forward voltages fairly accurately, and the power supply is well regulated, you can get away without any series resistor at all.

So, if you have fairly standard LEDs that have a forward voltage of 2V, then you could just put six of them in series. However, it will not be terribly easy to predict how much current the LEDs will take.

A safer approach is to arrange the LEDs in parallel strings, each string having its own current-limiting resistor (Figure 4-15).

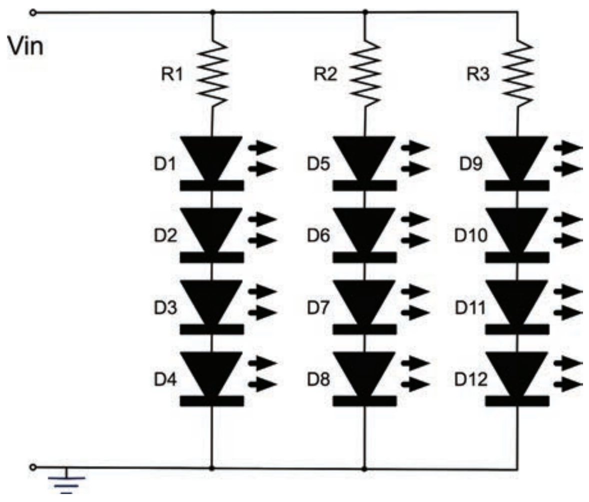
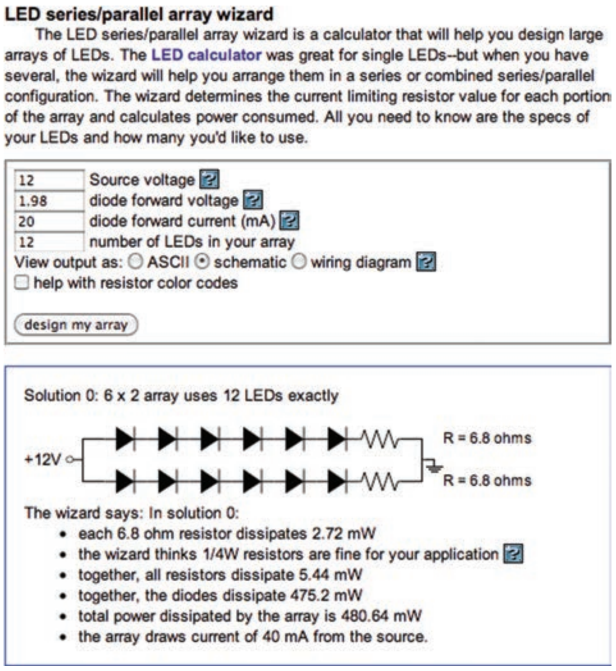


FIGURE 4-15 Powering multiple LEDs

Although the math for this isn't too hard, there can be a fair bit of it, so you can save yourself a lot of time by using an online calculator like http://led.linear1.org/led.wiz (Figure 4-16).

In this particular designer, you enter the source voltage for the overall supply, the LED forward voltage, the desired current for each LED, and the number of LEDs you want to light. The wizard then does the math and works out a few different layouts.

One consideration is that where you have a string of LEDs in series, if any of the LEDs fail, then all the LEDs will be off.



LED series/parallel array wizard

The LED series/parallel array wizard is a calculator that will help you design large arrays of LEDs. The LED calculator was great for single LEDs--but when you have several, the wizard will help you arrange them in a series or combined series/parallel configuration. The wizard determines the current limiting resistor value for each portion of the array and calculates power consumed. All you need to know are the specs of your LEDs and how many you'd like to use.

12	Source voltage
1.98	diode forward voltage
20	diode forward current (mA)
12	number of LEDs in your array

View output as: ASCII [✓ schematic] wiring diagram

help with resistor color codes

Solution 0: 6 x 2 array uses 12 LEDs exactly

The wizard says: In solution 0:
- each 6.8 ohm resistor dissipates 2.72 mW
- the wizard thinks 1/4W resistors are fine for your application
- together, all resistors dissipate 5.44 mW
- together, the diodes dissipate 475.2 mW
- total power dissipated by the array is 480.64 mW
- the array draws current of 40 mA from the source.

FIGURE 4-16 The LED wizard

Making LEDs Flash

The 555 timer IC is a useful little IC that can be used for many different purposes, but is particularly convenient for making LEDs flash or generating higher frequency oscillations suitable for making audible tones (see Chapter 10).

We are going to make this design on breadboard and then transfer it to a more permanent home on a bit of stripboard.

You Will Need

Quantity	Names	Item	Appendix Code
1		Solderless breadboard	K1, T5
1	D1	Red LED	K1
1	D1	Green LED	K1
1	R1	1kΩ resistor	K1
1	R2	470kΩ resistor	K1
2	R3, R4	270Ω resistor	K1
1	C1	1µF capacitor	K1
1	IC1	555 timer	K1
4		Male-to-male jumper wire or solid core wire	K1, T6
1		4 × AA battery holder	K1, H1
1		Battery clip	K1, H2
4		AA batteries	

Breadboard

The schematic for the LED flasher is shown in Figure 4-17.

The breadboard layout is shown in Figure 4-18. Make sure you have the IC the right way up. There will be a notch in the IC body next to the top (pins 1 and 8). The capacitor and LEDs must both be the correct way around, too.

Figure 4-19 shows the finished breadboard. You should find that the LEDs alternate, each staying on for about a second.

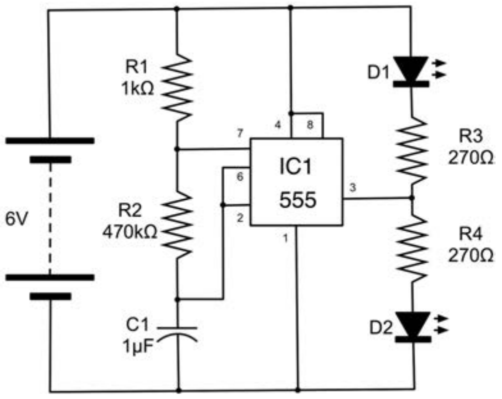


FIGURE 4-17 The LED flasher schematic

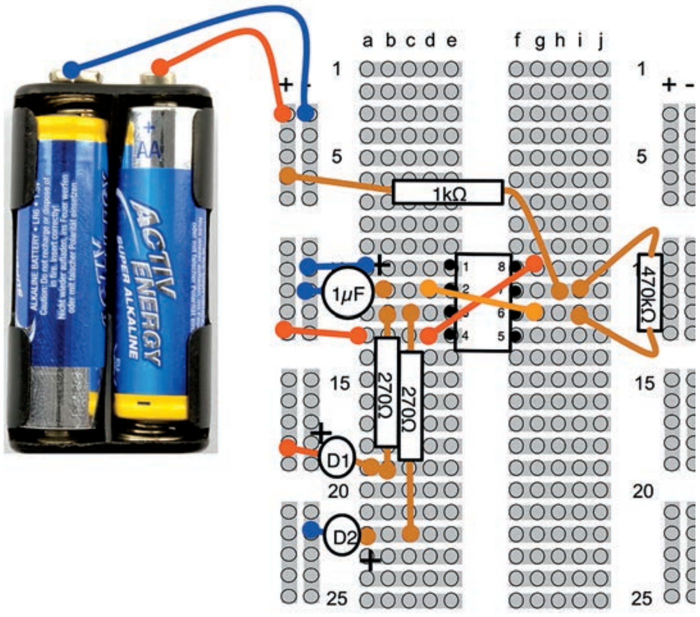


FIGURE 4-18 The LED flasher breadboard layout

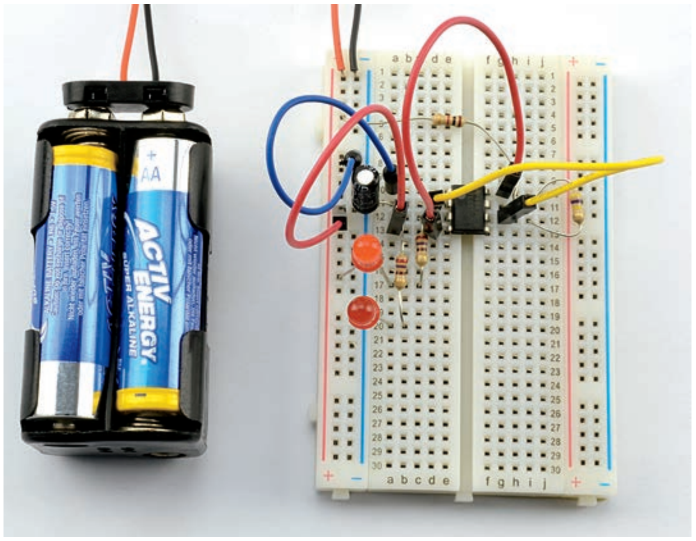


FIGURE 4-19 The LED flasher on breadboard

Now that we know that the design is right and everything works, try swapping out R2 with a 100kΩ resistor and notice the effect on the flashing.

The 555 timer is a very versatile device, and in this configuration it oscillates at a frequency determined by the formula:

frequency = 1.44 / ([R1 + 2 * R2] * C)

where the units of R1, R2, and C1 are in Ω and F. Plugging in the values for this design, we get

frequency = 1.44 / ([1000 + 2 * 470000] * 0.000001) = 1.53 Hz

One hertz (or Hz) means one oscillation per second. When we use the 555 timer Chapter 10 to generate an audible tone, we will be using the same circuit to generate a frequency in the hundreds of hertz.

As with so many electronic calculations, there are also online calculators for the 555 timer.

How to Use Protoboard (LED Flasher)

Breadboard is very useful for trying things out, but not so useful as a permanent home for your electronics. The problem is that the wires tend to fall out after a while.

The easiest way to transfer a breadboard design onto something more permanent is to use “Protoboard,” that is, a PCB with the same layout as a breadboard. There are various types of such breadboard including PermaProto board from Adafruit and the board shown in Figure 4-20, which is MonkMakes Protoboard (see Appendix).

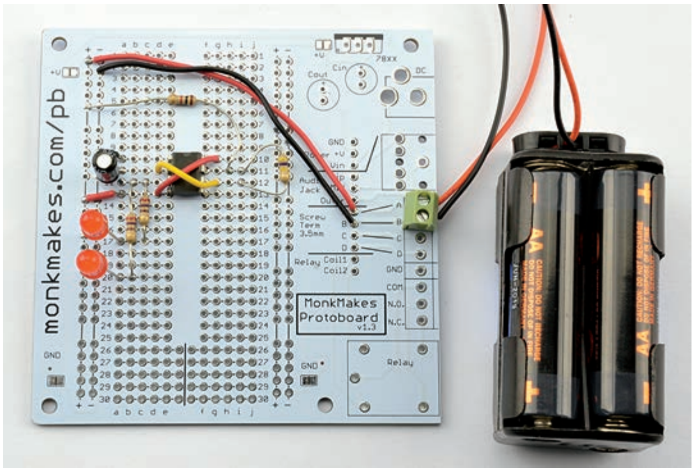


Figure 4-20 An LED flasher built on MonkMakes Protoboard

The components can just be transferred from their positions on the breadboard to the Protoboard and soldered to the pads on the underside of the board. The MonkMakes Protoboard also has an area on the side where a PCB screw terminal and other components that don't fit well onto breadboard can be soldered. In this case, the screw terminal is used to connect the battery clip.

Using Stripboard (LED Flasher)

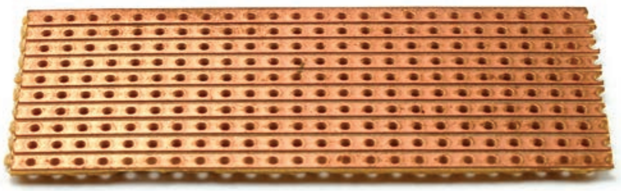


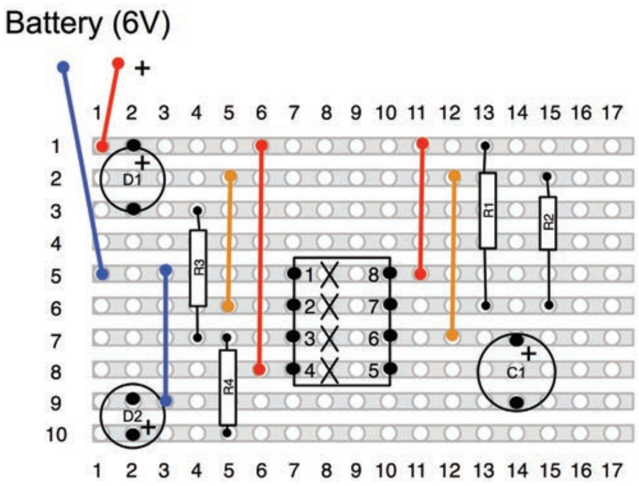
FIGURE 4-21 Stripboard

Stripboard (Figure 4-21) is a bit like general-purpose printed circuit board. It is a perforated board with conductive strips running underneath, rather like breadboard. The board can be cut to the size you need and components and wires soldered onto it.

Designing the Stripboard Layout

Figure 4-22 shows the final stripboard layout for the LED flasher that we made in the previous section. It is not easy to explain how we got to this from the schematic and breadboard layout. There is a certain amount of trial and error, but there are a few principals you can follow to try and make it easier.

FIGURE 3-22 An LED flasher stripboard layout



The free software Fritzing (http://fritzing.orh) is designed to simplify drawing breadboard layouts. It also has a feature that allows you to draw stripboard layouts.

The Xs underneath the IC are breaks in the track, which we will make with a drill bit. One of the goals of a good stripboard layout is to try and avoid making too many breaks in the track. Breaks are unavoidable for an IC like this. If we did not make them, pin 1 would be connected to pin 8, pin 2 to pin 7, and so on, and nothing would work.

The colored lines on the board are linking wires. So, for instance, from the schematic diagram of Figure 4-17, we can see that pins 4 and 8 of the IC should be connected together and both go to the positive supply. This is accomplished by the two red linking wires. Similarly, pins 2 and 6 need to be connected together. This is accomplished by using the orange leads.

Although logically the stripboard layout is the same as the schematic, the components are in rather different places. The LEDs are on the left in the stripboard layout and on the right on the schematic. It is not always like this, and it's easier if they are similar, but in this case the left-hand pins of the IC include the output pin 3 that the LEDs need, and the pins connected to R1, R3, and C1 are all on the right-hand side of the IC.

Try making a stripboard layout from the schematic, you may well come up with a different and better layout than the one I produced.

The steps I went through in designing this layout are as follows:

1. Place the IC fairly centrally, with a bit more room above than below and with pin 1 uppermost (convention).
2. Find a good place for R3 and R4 to be put so the strips are at least three holes apart for one resistor lead, when the other lead of each resistor goes to pin 3.
3. Pick the top track of the stripboard to be +V so it can be close to the positive end of one of the LEDs.
4. Pick row 5 to be the ground connection. This way it can run straight on to pin 1 of the IC.
5. Add a link wire from row 5 to row 9 to provide the negative connection for the LED D2.
6. Put a jumper wire from pin 4 of the IC to row 1 (+V).

Turning now to the right-hand side of the board:

1. Put a jumper in connecting pin 8 of the IC to row 1 (+V).
2. R1 and R2 both have one end connected to pin 7, so put them side by side with the far end of R1 going to row 1 (+V).
3. R2 needs to connect to pin 6, but pin 6 and pin 7 of the IC are too close together for the resistor to lie flat, so take that lead up to the unused row 2 instead, then put jumpers from row 2 down to both pin 6 and pin 2 of the IC.
4. Finally, C1 needs to go between pin 6 (or pin 2, but 6 is easier) and GND (row 9).

A good way of checking that you have made all the connections you need is to print off the schematic and then go through each connection on the stripboard and check off its counterpart on the schematic.

This may all seem a little like magic, but try it. It's not as hard to do as it is to describe.

Before we start soldering, it is worth considering what kind of LEDs you want to use for this project. You may decide to use higher-brightness LEDs or power the project from a lower voltage. If you do decide on this, recalculate the values for R3 and R4 and try it out on the breadboard layout. The 555 timer IC needs a supply voltage between 4.5V and 16V, and the output can supply up to 200mA.

Construction

Step 1. Cut the Stripboard to Size

There is no point in having a large bit of stripboard with just a few components on it, so the first thing we need to do is cut the stripboard to the right size. In this case, it's ten strips each of 17 holes. Stripboard doesn't actually cut very well. You can use a rotary tool, but wear a mask because the dust from the stripboard is nasty and you really do not want it in your lungs. I find the easiest way to cut stripboard is actually to score it with a craft knife and metal ruler on both sides and then break it over the edge of your work surface.

Score it across the holes, not between them. When the board is cut, the copper underside will look like Figure 4-23.

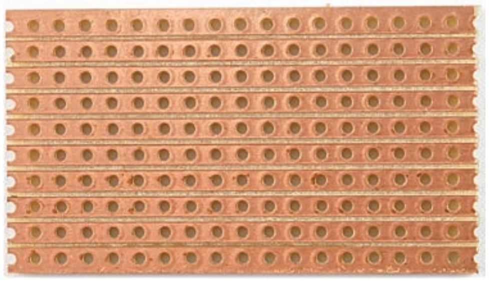


FIGURE 4-23 A stripboard cut to size

Step 2. Make the Breaks in the Tracks

A good tip is to use a permanent marker and put a dot in the top-left corner. Otherwise, it is very easy to get the board turned around, resulting in breaks and links being put in the wrong place.

To make the breaks, count the position in rows and columns of the break from the top of the board layout and then push a bit of wire so you can identify the right hole on the copper side of the stripboard (Figure 4-24a). Using a drill bit, just "twizzle" it between thumb and forefinger to cut through the copper track. It usually only takes a couple of twists (Figure 4-24b and c).

When you have cut all four breaks, the bottom side of the breadboard should look like Figure 4-25. Check very carefully

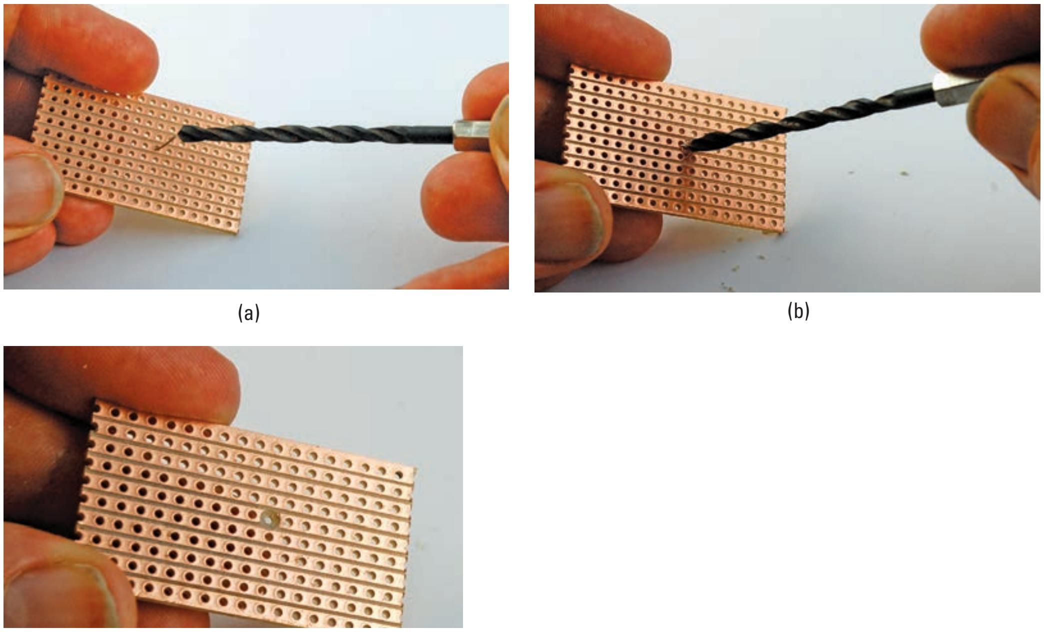

(a) (b) (c)

Figure 4-24 Cutting a track on stripboard

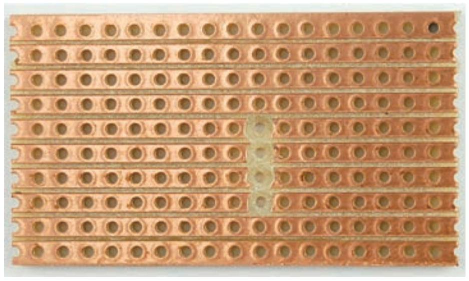


Figure 4-25 The stripboard with breaks cut

that none of the burrs from the copper have lodged between the tracks and that the breaks are complete. Photographing the board and then zooming in is a great way of actually checking the board.

Step 3. Make the Wire Links

A golden rule of any type of circuit board construction, including when using stripboard, is to start with the lowest-lying components. This is so that when you turn the board on its back to solder it, the thing being soldered will stay in place through the weight of the board.

In this case, the first thing to solder is the links.

Strip and cut solid-core wire to slightly longer than the length of the link. Bend it into a U-shape and push it through the holes at the top, counting the rows and columns to get the right holes (Figure 4-26a). Some people get very skilled at

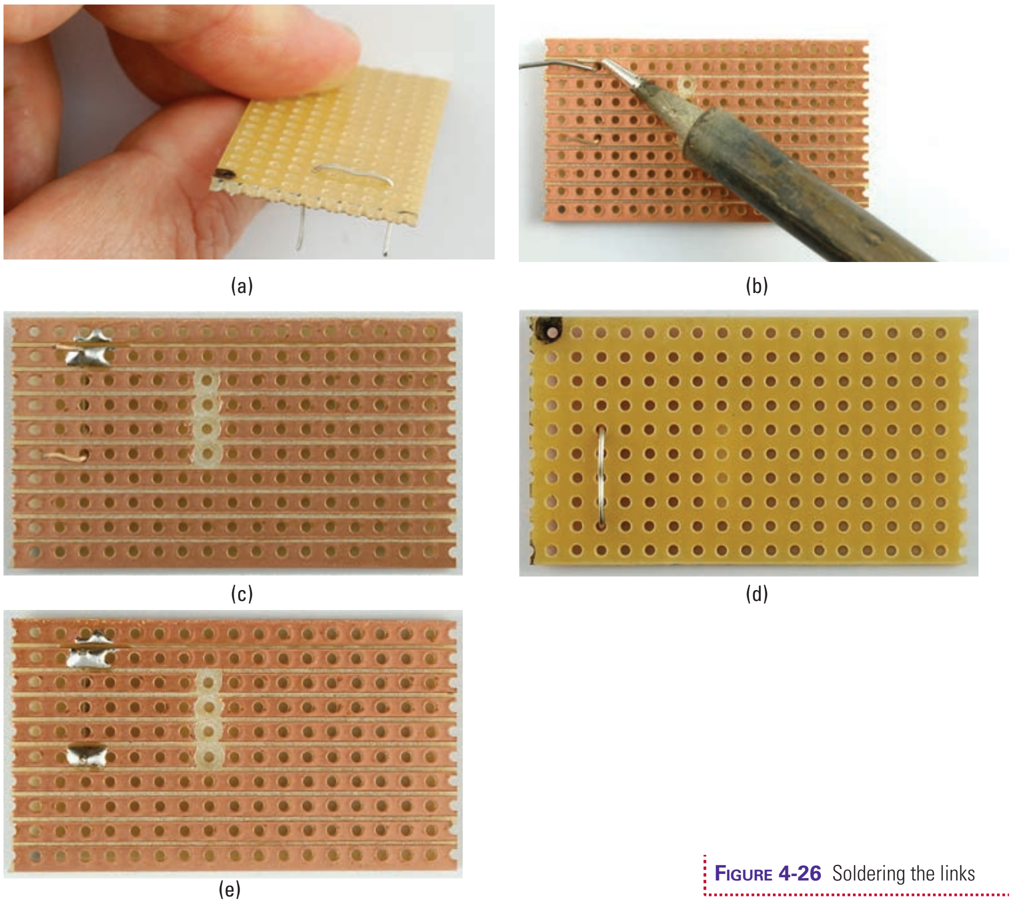

(a) (b) (c) (d) (e)

FIGURE 4-26 Soldering the links

bending the wires with pliers to just the right length. I find it easier to bend the wires with a bit of a curve so they will kind of squash into the right holes. I find this easier than trying to get the length just right from the start.

Turn the board over (see how the wire is held in place) and solder the wire by holding the iron to the point where the wire emerges from the hole. Heat it for a second or two and then apply the solder until it slows along the track, covering the hole and wire (Figure 4-26b and c).

Repeat this process for the other end of the lead and then snip off the excess wire (Figure 4-26d and e).

When you have soldered all the links, your board should look like the one in Figure 4-27.

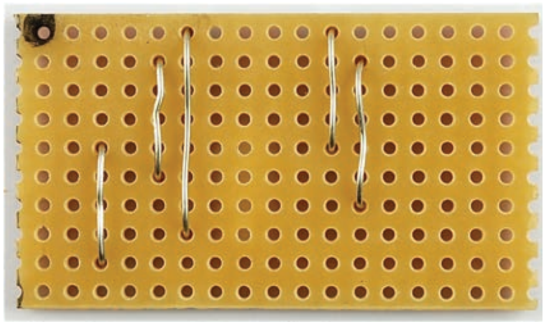


FIGURE 4-27 The stripboard with all its links

Step 4. Solder the Resistors

The resistors are the next lowest components to the board, so solder these next, in the same way as you did the links. When they are all soldered, the stripboard should look like Figure 4-28.

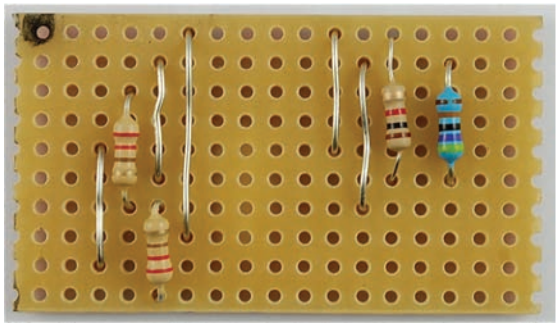


FIGURE 4-28 The stripboard with resistors in place

Step 5. Solder the Remaining Components

Next, solder the LED, capacitor (which can be laid on its side as shown in Figure 4-29), and finally the LEDs and connectors to the battery clip.

That's it. Now it's time for the moment of truth. Before you plug it in, do a very careful inspection for any shorts on the underside of the board.

If everything seems in order, connect the battery clip to the battery.

Troubleshooting

If it does not work, immediately unplug it and go back through and check everything, especially that the LEDs,

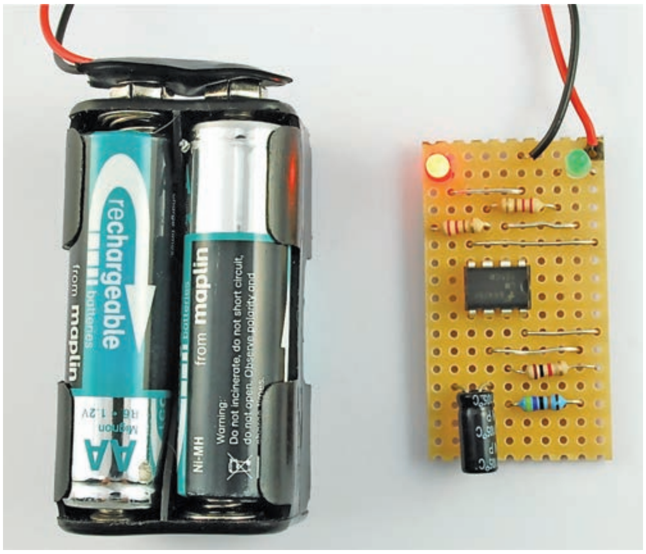


FIGURE 4-29 The LED flasher on the stripboard

IC, and capacitor are the correct way around. Also check that the batteries are okay.

Laser Diode Modules

FIGURE 4-30 A laser diode module

Lasers are best bought as laser modules. The difference between a laser module and a laser diode is that the module includes a laser diode as well as a lens to focus the beam of laser light and a drive circuit to control the current to the laser diode.

If you buy a laser diode, you will have to do all this yourself.

A laser diode module, such as the 1mW module shown in Figure 4-30, comes with a datasheet that says it needs to be supplied with 3V. So, all you need to do is find it a 3V battery and connect it up.

Hacking a Slot Car Racer

Slot cars are a lot of fun, but could be improved by adding headlights and working brake lights to the car (Figure 4-31).

LEDs are just the right size to be fitted front and back into a slot car.

You Will Need

You will need the following items to add lights to your slot car.

Quantity	Name	Item	Appendix Code
1		Slot car racer for modification	
1	D1	1N4001 diode	K1, S5
2	D2, D3	High brightness white LED	K1
2	D4, D5	Red LED	K1, S11
4	R1–4	1kΩ resistor	K1
1	C1	1000µF 16V-capacitor	C1
		Red, yellow, and black hookup wire	K1, T7, T8, T9
1		* Two-way header plug and socket	

* I used a scavenged two-way header socket and plug to make it easier to work on the two halves of the car. This is not essential.

The slot car used here was part of a build-your-own slot car that has plenty of room inside for the electronics. Plan ahead and make sure you can fit everything in.

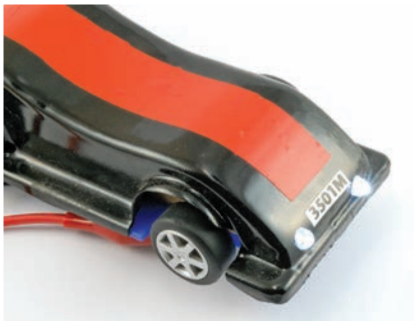


FIGURE 4-31 A modified slot car racer

Storing Charge in a Capacitor

To make the brake lights stay on for a few moments after the car has stopped, you will need a capacitor to store charge.

If you think back to the idea of electricity as water flowing in a river, then a capacitor is a bit like a tube containing an elastic membrane (think balloon material) that stretches with the water pressure. Figure 4-32 shows how a capacitor can be used to store charge.

Figure 4-32a shows the capacitor (C1) being filled with water through A. Throughout this, water will also flow along the top and drive a water wheel, turning electrical energy into motion, a bit like how a light bulb or LED turns electrical energy into light. The water falls out of the bottom, returning to ground. Imagine a pump (like a battery) pulling the water back up for another circuit. If the water stops flowing into C1 through A, then the pressure on the elastic membrane will be relaxed and the membrane will return to its normal flat state, pushing water back over the top to keep the water wheel turning until the water level in C1 drops below that of the outlet of the water well.

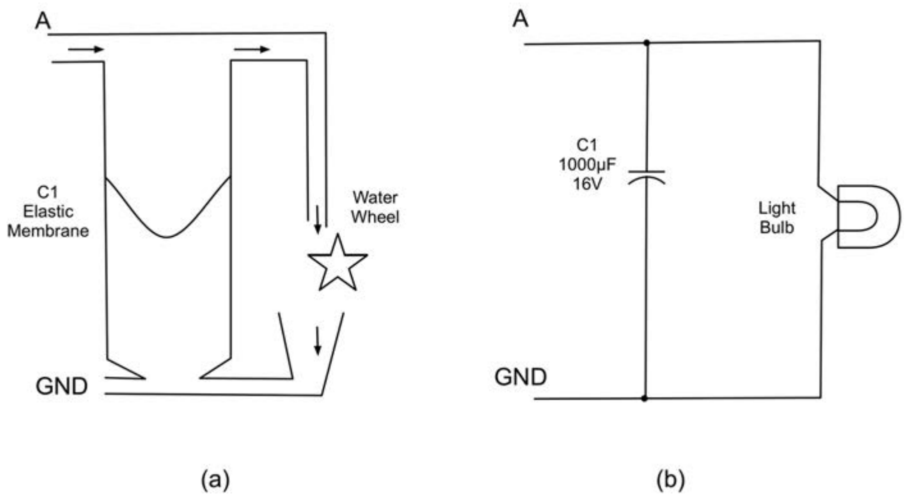


FIGURE 4-32 A capacitor as a tank

Figure 4-32b shows the electronic equivalent of this circuit. While the voltage at A is higher than GND, C1 will fill with charge and the light will be lit.

When the voltage at A is disconnected, the capacitor will discharge through the light bulb, lighting it. As the level of voltage drops in the capacitor, the bulb will gradually dim until it goes out as it reaches GND.

On the face of it, you can think of capacitors as being a bit like batteries. Both store charge. However, there are some very important differences.

- Capacitors only store a tiny fraction of the charge that a battery of the same size can store.
- Batteries use a chemical reaction to store electrical energy. This means their voltage remains relatively constant until they are spent, at which time it falls off rapidly. Capacitors, however, drop evenly in voltage as they discharge, just like the level of water decreasing in a tank.

Design

Figure 4-33 shows the schematic diagram for this modification.

The headlights (D2 and D3) are powered all the time from the slot car's connection to the track, so whenever the motor is running the LEDs will light.

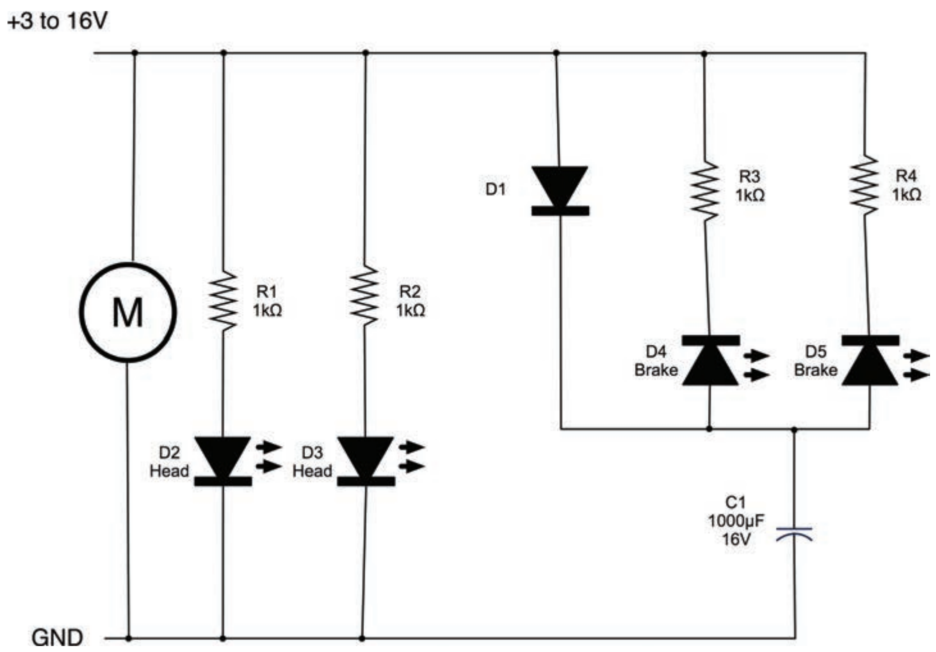


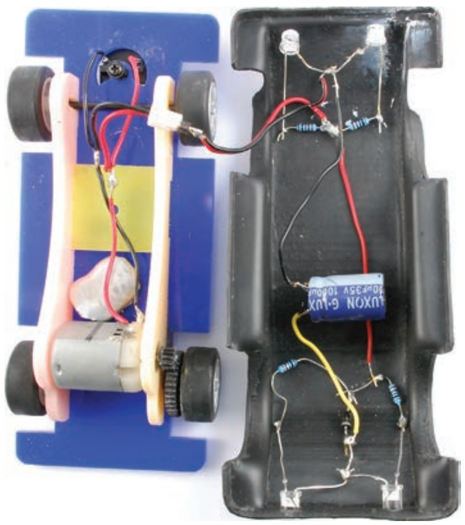
FIGURE 4-33 A schematic diagram for the slot car modification

The brake lights are more interesting. These will automatically come on when the car stops, and then go off after a few seconds. To do this, we make use of a capacitor C1.

When the car is powered, C1 will be charged through D1. At this point, the brake lights D4 and D5 will not be lit because they will be reverse-biased—that is, the voltage going in from the car tracks will be higher than the voltage at the top of the capacitor.

When you release the trigger on the controller, there will be no voltage coming in. Now the voltage at the top of the capacitor will be higher than the voltage coming in, so the capacitor will discharge through D4 and D5, making them light.

FIGURE 4-34 The components inside the car



Construction

Figure 4-34 shows how the components are laid out in the two halves of the car.

How you lay these out in your vehicle may vary depending on how much space you have.

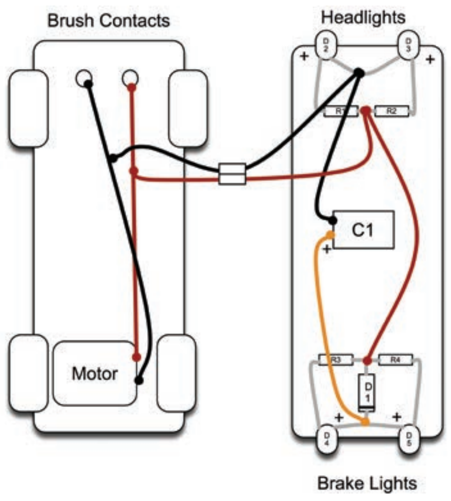


FIGURE 4-35 The wiring diagram for the modified car

Holes were drilled in the case to take the 5mm LEDs. The LEDs are a snug fit in the holes and stay in place without any glue.

Figure 4-35 shows a wiring diagram for the arrangement that makes it easier to see what is going on.

Use your multimeter of the 20V range to identify which is the positive power connector on the contacts at the front of the car. This contact is connected to the red lead.

The longer leads of the LEDs are the positive connections, and the capacitor's negative lead should be marked with a "-".

The optional connector makes it easier to work on the two halves of the car separately.

Testing

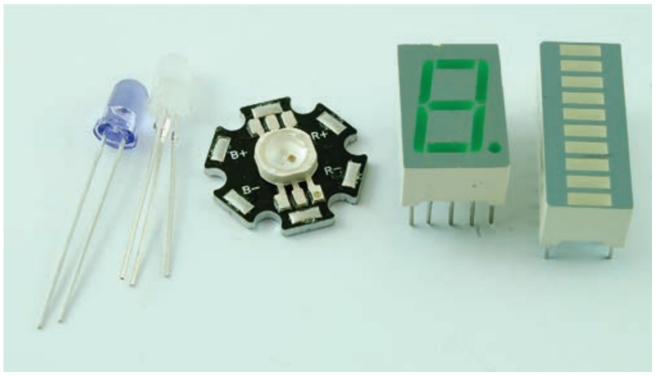
Testing really just involves trying out the car on the track. If the headlight LEDs are not on as soon as you squeeze the trigger on the controller, check the wiring, paying special attention to the polarity of the LEDs.

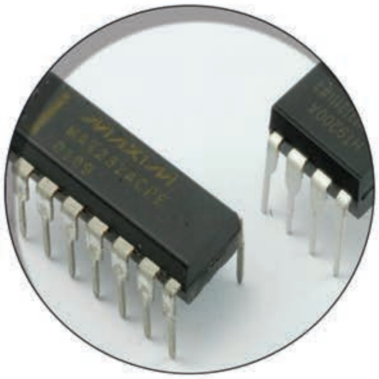
Summary

We have learned how to use LEDs in this chapter, as well as picked up some good building skills so we can make our creations a bit more permanent using stripboard.

In the next chapter, we will examine sources of power, including batteries, power supplies, and solar panels. We will also look at how to select the right kind of battery, repurpose old rechargeable batteries, and use them in our projects.

ARDUINO
UNO
DIGITAL (PWM~)


B+
R+
R-
B-



5

Batteries and Power

Everything that you make or adapt is going to need to get its power from somewhere. This might be from a household electricity adapter, solar panels, rechargeable batteries of various sorts, or just standard AA batteries.

In this chapter, you will find out all about batteries and power, starting with batteries.

Note I use the word "battery" to describe both batteries and cells. Strictly speaking, a battery is a collection of cells wired one after the other to give the desired voltage.

Selecting the Right Battery

There are many types of battery on the market. So, to simplify things, in this chapter we will just look at the most common types of battery, those that are readily available and that will be used in most of the projects in this book.

Battery Capacity

Whether single-use or rechargeable, batteries have a capacity—that is, they hold a certain amount of electricity. Manufacturers of single-use batteries often don't specify this capacity in the batteries you buy from a supermarket. They just label them heavy duty/light duty, and so on. This is a little like having to buy milk and being given the choice of "big bottle" or "small bottle" without being able to see how big the bottle is or be told how many pints or liters it contains. One can speculate as to the reasons for this. One reason might be that battery producers think the public isn't intelligent enough to understand a stated battery capacity. Another might be that the longer a battery is on the shelf, the more its capacity shrinks. Still another is that the capacity actually varies a lot with the current drawn from the battery.

Anyway, if a battery manufacturer is kind enough to tell you what you are buying, the capacity figure will be stated in Ah or mAh. So a battery that claims to have a capacity of 3000mAh (typical of a single-use alkaline AA cell) can supply 3000mA for one hour. Or, alternatively, 3A for an hour. But it doesn't have to draw 3A. If your project only uses 30mA, you can expect the battery to last 100 hours (3000/30). In truth, the relationship is not quite that simple, because as you draw more current, the capacity decreases. Nevertheless, this will do as a rule of thumb.

Maximum Discharge Rate

You cannot take a tiny battery like a CR2032 with a capacity of 200mAh and expect to power a big electric motor at 20A for 1/100 of an hour (six minutes). There are two reasons for this. First, all batteries actually have an internal resistance. So, it is as if there is a resistor connected to one of the terminals. This varies depending on the current being drawn from the battery, but may be as high as a few tens of ohms. This will naturally limit the current.

Second, when a battery is discharged too quickly, by too high a current, it gets hot—sometimes very hot, sometimes "on fire" hot. This will damage the battery.

Batteries therefore also have a safe discharge rate, which is the maximum current you can safely draw from it.

Single-Use Batteries

Although somewhat wasteful, sometimes it makes sense to use single-use batteries that cannot be recharged. You should consider single-use batteries if:

- The project uses very little power, so they will last a long time anyway.
- The project will never be close to someplace where it can be charged up.

Table 5-1 shows some common single-use batteries. These figures are typical values and will vary a lot between actual devices.

Especially when it comes to the maximum discharge rate, you may get away with a lot more, or the battery may fail or get very hot with considerably less. It will also depend on how well ventilated a box they are in, as heating under high currents is a big problem.

So in the spirit of hacking electronics, spend less time planning and more time trying.

Type		Typical Capacity	Voltage	Max. Discharge Current	Features	Common Uses
Lithium button cell (e.g., CR2032)		200mAh	3V	4mA with pulses up to 12mA	High temperature range (–30 to 80°C); small	Low-power devices; RF remote controls; LED key ring lights; etc.
Alkaline PP3 battery		500mAh	9V	800mA	Low cost; readily available	Small portable electronic devices; smoke alarms; guitar pedals
Lithium PP3		1200mAh	9V	400mA pulses up to 800mA	Expensive; light; high-capacity	Radio receivers
AAA cell		800mAh	1.5V	1.5A continuous	Low-cost; readily available	Small motorized toys; remote controls
AA cell		3000mAh	1.5V	2A continuous	Low cost; readily available	Motorized toys
C cell		6000mAh	1.5V	Probably get away with 4A	High-capacity	Motorized toys; high-powered flashlights
D cell		15,000mAh	1.5V	Probably get away with 6A	High-capacity	Motorized toys; high-powered flashlights

TABLE 5-1 Single-Use Battery Types

Note Some of the photographs in the table are of branded batteries. The figures shown are for batteries of that type, not specifically the batteries listed.

Roll Your Own Battery

A single-cell battery with a voltage of just 1.5V is probably not going to be of any use. You will normally need to put a number of these cells in series (end to end) to produce a battery of the desired voltage.

When you do this, you do not increase the capacity. If each cell was 2000mAh, then if you put four 1.5V batteries in series, the capacity would still be 2000mAh, but at 6V rather than 1.5V.

Battery holders such as the one shown in Figure 5-1 are a great way of doing this. Look closely at how the battery holder is constructed and you'll notice how the positive of one battery is connected to the negative of the next, and so on.

This holder is designed to take six AA batteries so as to produce an overall voltage of 9V. Battery holders like this are available to take two, four, six, eight, or ten cells, both in AA and AAA.

Another advantage of using a battery holder is that you can use rechargeable batteries instead of single-use batteries. However, rechargeable cells normally have a lower voltage, so you have to take this into account when calculating the overall voltage of your battery pack.

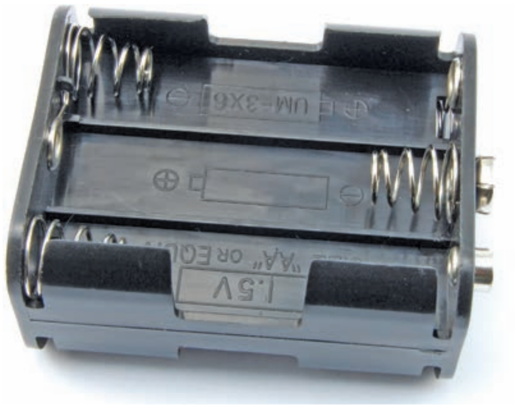


FIGURE 5-1 A battery holder

Selecting a Battery

Table 5-2 should help you decide on a suitable battery for your project. There is not always a best answer to the question "Which battery should I use?" and this table is definitely in the territory of rules of thumb.

You should also do the math and include how frequently the battery will need replacing.

	Voltage			
Power	**3V**	**6V**	**9V**	**12V**
Less than 4mA (short bursts) or 12mA continuous	Lithium button cell (e.g., CR2032)	2 × Lithium button cell (e.g., CR2032)	PP3	Unlikely
Less than 3A (short bursts) or 1.5A continuous	2 × AAA battery pack	4 × AAA battery pack	6 × AAA battery pack	8 × AAA battery pack
Less than 5A (short bursts) or 2A continuous	2 × AAA battery pack	4 × AAA battery pack	6 × AAA battery pack	8 × AAA battery pack
Even more	2 × C or D battery pack	4 × C or D battery pack	6 × C or D battery pack	8 × C or D battery pack

TABLE 5-2 Selecting a Single-Use Battery

Rechargeable Batteries

Rechargeable batteries can provide both cost and green benefits over single-use batteries. They are available in different types and in different capacities. Some, such as rechargeable AA or

AAA batteries, are designed as replacements for single-use batteries, and you remove them to charge in a separate charger. Other batteries are intended to be built into your project so all you have to do is plug a power adapter into your project to charge the batteries without removing them. The advent of cheap, high-capacity, low-weight lithium polymer (LiPo) batteries has made this a common approach for many items of consumer electronics.

Table 5-3 shows some commonly used types of rechargeable batteries.

Although there are many more types than this, these are the most commonly used batteries. Each type of battery has its own needs when it comes to charging, and we will look at each in later sections.

Type		Typical Capacity	Voltage	Features	Common Uses
NiMH button cell pack		80mAh	2.4 or 3.6V	Small	Battery backup
NiMH AAA cell		750mAh	1.25V	Low cost	Replacement for single-use AAA cell
NiMH AA cell		2000mAh	1.25V	Low cost	Replacement for single-use AA cell
NiMH C cell		4000mAh	1.25V	High capacity	Replacement for single-use C cell
Small LiPo cell		50mAh	3.7V	Low cost; high capacity for weight and size	Micro-helicopters
LC18650 LiPo cell		2200mAh	3.7V	Low cost; high capacity for weight and size; slightly bigger than AA	High-power flashlights; Tesla Roadster (yes, really—about 6800 of them)
LiPo pack		900mAh	7.4V	Low cost; high capacity for weight and size	Cell phones, iPods, etc.
Sealed lead–acid battery		1200mAh	6/12V	Easy to charge and use; heavy	Intruder alarms; small electric vehicles/wheelchairs

TABLE 5-3 Rechargeable Batteries

	NiMH	LiPo	Lead–Acid
Cost per mAh	Medium	Medium	Low
Weight per mAh	Medium	Low	High
Self-discharge	High (flat in 2–3 months)	Low (6% per month)	Low (4% per month)
Handling of full charge/ discharge cycles	Good	Good	Good
Handling of shallow discharge/charge	Medium (regular full discharge prolongs battery life)	Medium (not well-suited to trickle charging)	Good

TABLE 5-4 Characteristics of Different Battery Technologies

Table 5-4 summarizes the features of NiMH, LiPo, and lead–acid battery technologies.

If you want your project to charge a battery in place, then a LiPo or sealed lead–acid battery is probably the best choice. However, if you want the option to remove the battery and/or use single-use batteries, then a AA battery pack is a good compromise between capacity and size.

For ultra-high-power projects, lead–acid batteries, despite being an ancient technology, still perform pretty well, just as long as you don't have to carry them around! They are also easy to charge and are the most robust of the technologies, offering the least chance of fire or explosion.

Charging Batteries (in General)

Certain principals apply no matter what kind of battery you are charging. So read this section before reading those that follow it concerning specific battery types.

C

The letter C is used to denote the capacity of a battery in Ah or mAh. So, when people talk about charging a battery, they often talk about charging at 0.1C or C/10. Charging a battery at 0.1C means charging it at 1/10 of its capacity per hour. For example, if a battery has a capacity of 2000mAh, then charging it at 0.1C means charging it with a constant current of 200mA.

Over-Charging

Most batteries do not respond well to being over-charged. If you keep supplying them with a high charging current, you will damage them.

For this reason, chargers often charge at a low rate (called trickle charging), so that the low current will not damage the battery. Clearly this makes charging slow. Or, they will use a timer or other circuitry to detect when a battery is full and either stop charging altogether, or switch to trickle charging, which keeps the battery topped up until you are ready to use it.

With some kinds of battery, notably LiPos and the lead–acid variety, if you charge the battery with a constant voltage, then as the battery becomes charged, its voltage rises to match the charging voltage and the current naturally levels off.

Many LiPo batteries now come with a little built-in chip that prevents over-charging automatically. Always look for batteries with such protection.

Over-Discharging

You are probably starting to get the impression that rechargeable batteries are fussy. If so, you're right. Most types of battery are equally unhappy if you over-discharge them and let them go completely flat.

Battery Life

Anyone with a laptop more than a few years old will notice that the capacity of the battery gradually decreases until the laptop only works when plugged in, since the battery has become completely useless. Rechargeable batteries (whatever the technology) can only be recharged a few hundred (perhaps 500) times before needing to be replaced.

Many manufacturers of consumer electronics now build the battery into the device in such a way that it is not "user serviceable," with the rationale that the life of the battery is probably longer than the attention span of the consumer.

Charging a NiMH Battery

If you are going to remove your batteries to charge them, this section is pretty trivial. You take them out and put them in a commercial NiMH battery charger that will charge them until they are full and then stop. You can then put them back into your project and you are done.

If, on the other hand, you want to leave the batteries in place while you charge them, then you need to understand a little more about the best way to charge your NiMH batteries.

Simple Charging

The easiest way to charge a NiMH battery pack is to trickle charge it, limiting the current with a resistor. Figure 5-2 shows the schematic for charging a battery pack of four NiMH batteries using a 12V DC adaptor like the one we used back in Chapter 1 to make our fume extractor.

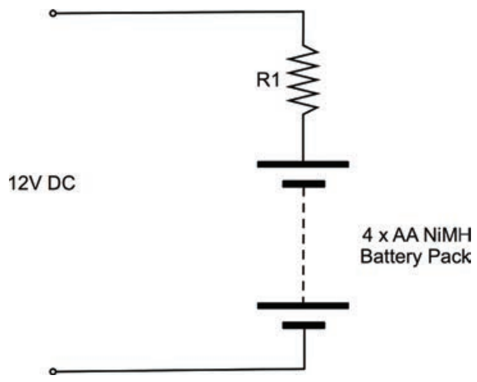


Figure 5-2 Schematic for trickle charging a NiMH battery pack

To calculate the value of R1, we first have to decide what current we want to charge our battery with. Generally, a NiMH battery can be safely trickle charged with less than 0.1C indefinitely. If the AA batteries we have each hold a C of 2000mAh, then we can charge them at up to 200mA. To be on the safe side, and if we planned to allow the batteries to "trickle" charge most of the time—for, say, a battery backup project—I would probably use a lower current of 0.05C or more conveniently C/20, which is 100mA.

Typically, the charge time for NiMH batteries is about 3C times the charging current, so at 100mA, we could expect our batteries to take 3 × 2000mAh / 100mA = 60 hours.

Back to calculating R1. When the batteries are discharged, each will be at a voltage of about 1.0V, so the voltage across the resistor will be 12V – 4V = 8V.

Using Ohm's law, R = V / I = 8V / 0.1A = 80Ω.

Let's be conservative and choose the convenient resistor value of 100Ω. Feeding this back in, the actual current will be I = V / R = 8V / 100Ω = 80mA.

When the batteries are fully charged, their voltage will rise to about 1.3V so the current will reduce to: I = V / R = (12V – 1.3V × 4) / 100Ω = 68mA.

That all sounds just fine, our 100Ω will be great. Now we just need to find out what maximum power rating we need for R1.

$$P = I\,V = 0.08A \times 8 = 0.64W = 640\ mW$$

So, we should probably use a 1W resistor.

Fast Charging

If you want to charge the batteries faster than that, then it is probably best to use a commercial charger, which will monitor the batteries and turn itself off or reduce the charge to a trickle when the batteries are full.

Charging a Sealed Lead–Acid Battery

These batteries are the least delicate of the battery types and could easily be trickle charged using the same approach as for NiMH batteries.

Charging with a Variable Power Supply

However, if you want to charge them faster, then it is best to charge them with a fixed voltage, with some current limiting (a resistor again). For a 12V battery (halve this for a 6V battery) until a discharged battery gets to around 14.4V, you can charge it with almost as much current as your power supply can take. It's only when it gets to this voltage that you need to slow down the charging to a trickle to prevent the battery from getting hot.

The reason we need to limit the current when the battery first starts to charge is that even if the battery doesn't get hot, the wires to it might get hot and whatever is supplying the voltage will only be able to supply a certain amount of current.

Figure 5-3 shows an adjustable power supply. Once you get into electronics, this is one of the first pieces of test equipment you should buy. You can use it in place of batteries while you are working on a project, and also use it to charge up pretty much any type of rechargeable battery.

A variable power supply lets you set both an output voltage and a maximum current. So the power supply will try and supply the specified voltage until the current limit is reached, at which point the voltage will drop until the current falls back below the set current.

Figure 5-3a shows the power supply set to 14.4V and we have attached the power leads to an empty 12V 1.3Ah sealed lead–acid battery. We will start by adjusting the current setting of the power supply to minimum, so as to prevent any nasty surprises. The voltage immediately drops to 11.4V (Figure 5-3b), so we can

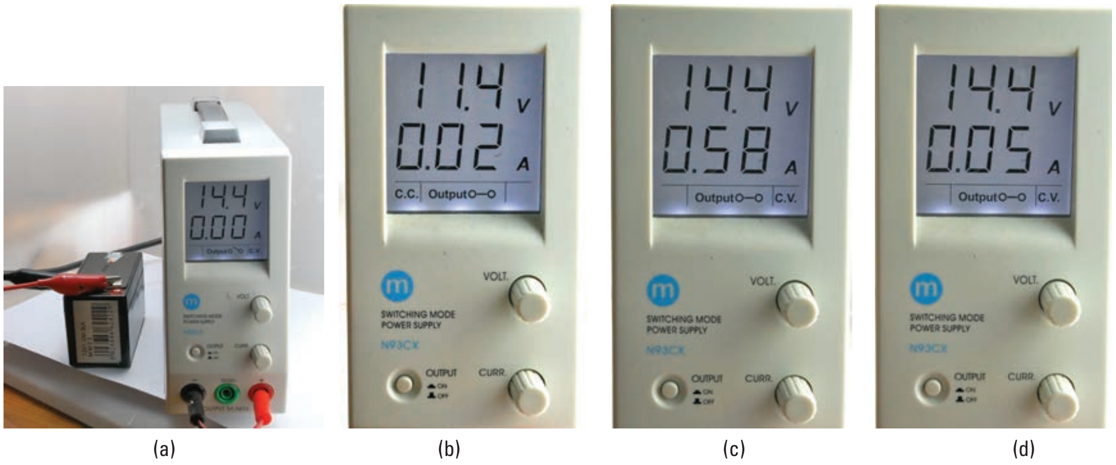


FIGURE 5-3 Using a variable power supply to charge a lead–acid battery

gradually increase the maximum current. In actual fact, even with no current limiting (turning the current knob to maximum), the current only rose to 580mA and the voltage increases to 14.4V (Figure 5-3c). After about two hours, the current has dropped to just 200mA, indicating that our battery is getting full. Finally, after four hours, the current is just 50mA and the battery is now fully charged (Figure 5-3d).

Charging a LiPo Battery

The technique we have just used on a lead–acid battery using a variable power supply will work just as well on a LiPo battery if we adjust the voltage and current accordingly.

For a LiPo cell, the voltage should be set to 4.2V and the current limited (usually to 0.5A) for a smallish cell, but currents up to C are sometimes used in radio-controlled vehicles.

However, unlike lead–acid and NiMH batteries, you cannot put a number of cells in series and charge the whole lot as one battery. Instead, you have to charge them separately, or use a "balanced charger" that monitors the voltage at each cell separately and controls the power to each.

The safest and most reliable way to charge a LiPo is to use one of the chips that exist just for that purpose. These chips are cheap, but generally only available as surface-mounted

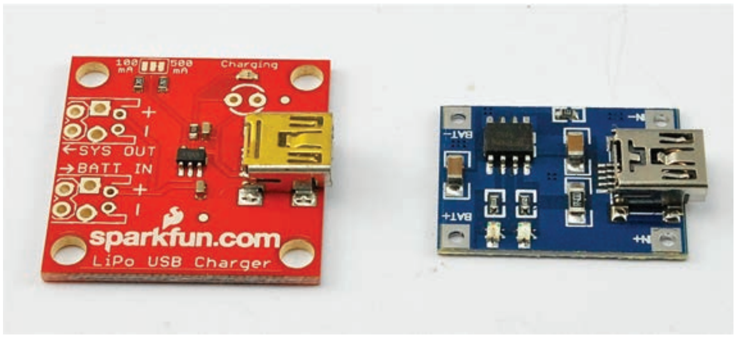


FIGURE 5-4 SparkFun and generic LiPo chargers

components. However, there are plenty of ready-made modules available, many of which use the MCP73831 IC. Figure 5-4 shows two of these—one from SparkFun (see the Appendix, M16) and one for just a few dollars from eBay.

Both are used in the same manner. They will charge a single LiPo cell (3.7V) from a USB input of 5V. The SparkFun board has space on the PCB for two other connectors, one to which the battery is connected and the other for a second connection to the battery—the intention is that you connect the electronics that will use the battery to the second socket. The sockets can be either JST connectors as are often found on the end of the leads of a LiPo batter, or just screw terminals. The SparkFun module allows you to select the charging current, using a connection pad.

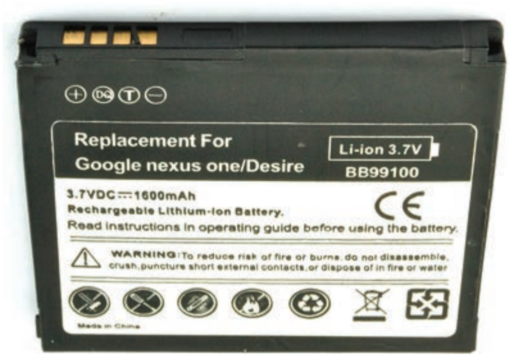
The generic module on the right has a fixed charge rate of 500mA and just a single pair of connections for the battery.

It is not a good idea to trickle charge a LiPo. If you want to keep them topped up for, say, a battery backup solution, then leave them attached to the charger.

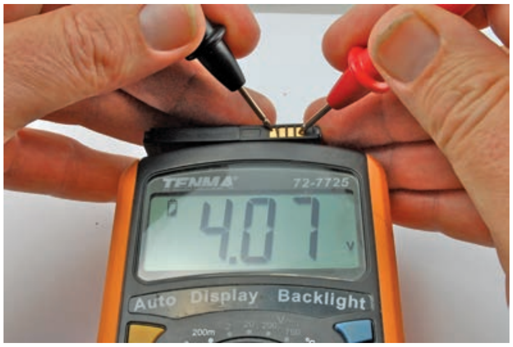
Hacking a Cell Phone Battery

Most of us have a cell phone or two languishing in a drawer somewhere, and one of the useful components that can usually be scavenged (assuming it's not the reason the phone is in the drawer) is the battery. The power supply is another useful component often found in people's drawers.

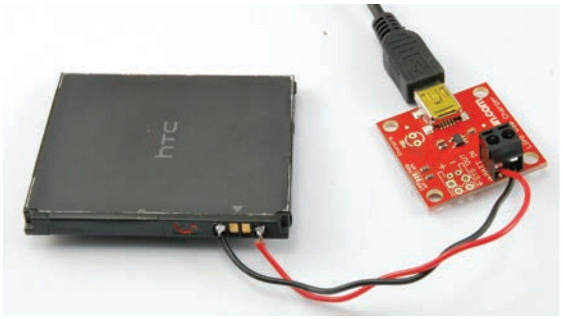
Figure 5-5a shows a fairly typical vintage cell phone battery. The battery is 3.7V (a single cell) and is 1600mAh (pretty good). Cell phone batteries normally have more than just the usual two


Replacement For
Google nexus one/Desire
Li-ion 3.7V
BB99100
3.7VDC 1600mAh
Rechargeable Lithium-Ion Battery.
Read instructions in operating guide before using the battery.

(a)



(b)



(c)

Figure 5-5 Hacking a cell phone battery

connections for positive and negative. So the first task must be to identify the connections on the battery.

To identify the positive and negative connections to the cell, just put your multimeter into the 20V DC range and test each combination of pairs until you get the meter to read something over 3.5V, depending on how well charged the battery is (Figure 5-5b).

The batteries often have gold-plated contacts that make them very easy to solder leads to. Once they have leads attached, you can use a charger like the one described in the previous section. Figure 5-5c shows the SparkFun charger module being used for just that purpose.

Caution When using a LiPo battery, remember that if you discharge them too far (below about 3V per cell), you can permanently damage them. Most new LiPo batteries will include an automatic cut-off circuit, built into the battery package, to prevent over-discharging, but this may not be the case for a scavenged battery.

Controlling the Voltage from a Battery

The thing with batteries is that even though they may say 1.5V, 3.7V, or 9V on the package, their voltage will drop as they discharge—often by quite a high percentage.

For example, a 1.5V alkaline AA battery when brand new will be about 1.5V and will quickly fall to about 1.3V under load but still deliver useful amounts of power down to about 1V. This means that in a pack of four AA batteries, the voltage could be anything between 6V and 4V. Most types of battery, whether single-use or rechargeable, exhibit a similar voltage drop.

This may not matter much; it just depends on what the battery is powering. If it is powering a motor or an LED, then the motor will just go a bit more slowly, or the LED will be a little dimmer as the battery discharges. However, some ICs have a very narrow voltage tolerance. There are ICs designed to work at 3.3V that specify a maximum working voltage of 3.6V. Similarly, if the voltage drops too low, the device will also stop working.

In fact, many digital chips such as microcontrollers are designed to work at a standard voltage of 3.3V or 5V.

To ensure a steady voltage, we need to use something called a voltage regulator. Fortunately for us, voltage regulators come as convenient three-pin, low-cost chips that are very easy to use. In fact, the packages just look like transistors, and the bigger the package, the more current they can control.

Figure 5-6 shows how you would use the most common of voltage regulators, called the 7805.

Using just a voltage regulator IC and two capacitors, any input voltage between 7V and 25V can be regulated to a constant 5V. The capacitors provide little reservoirs of charge that keep the regulator IC operating in a stable manner.

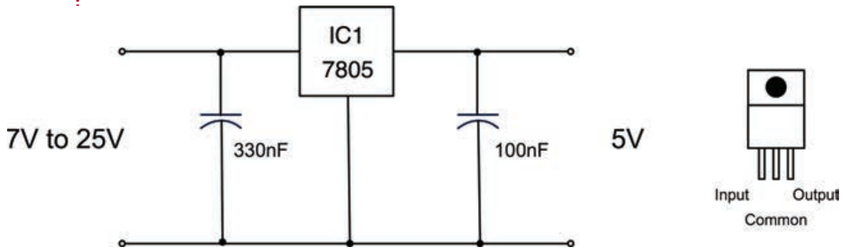


FIGURE 5-6 A voltage regulator schematic

In the following experiment with a 7805, we will omit the capacitors, as the supply voltage is a steady 9V battery and the load on the output is just a resistor (Figure 5-7).

The capacitors become much more necessary when the load varies (in other words, in the amount of current that it draws), something that is true of most circuits.

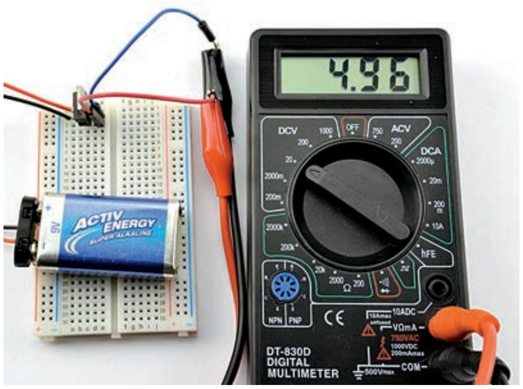


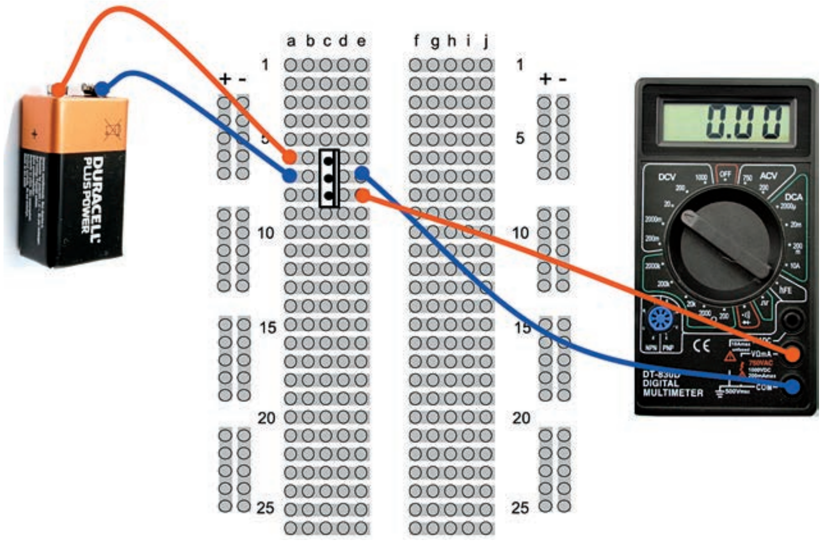
FIGURE 5-7 Experimenting with the 7805

You Will Need

Quantity	Names	Item	Appendix Code
1		Solderless breadboard	K1, T5
1	IC1	7805 voltage regulator	K1, S4
1		Battery clip	K1, H2
1		9V PP3 battery	

Wire up the breadboard as shown in Figure 5-8.

FIGURE 5-8 The 7805 breadboard layout



Breadboard

With the battery connected, the multimeter should display a voltage of close to 5V.

Although 5V is a very common voltage, there are voltage regulators for most common voltages, as well as the LM317 voltage regulator that we discussed in Chapter 4, that as well as providing constant current, can also be configured as a voltage regulator.

Table 5-5 lists some common voltage regulators that provide different output voltages and different current handling capabilities.

Output Voltage	100mA	1–2A
3.3V	78L33	LF33CV
5V	78L05	7805 (App A S4) (7–25V in)
9V	78L09	7809
12V	78L12	7812

TABLE 5-5 Voltage Regulators

Linear voltage regulators like the 78 series of ICs are ideal for low-power circuits of a few hundred mA; however, above that, they start to get hot and may need a heatsink attachment. To avoid this problem of the regulator IC getting hot, you can use a “switching” regulator module, like the one shown in Figure 5-9, that can regulate a voltage at currents of well over 1A without getting hot at all.

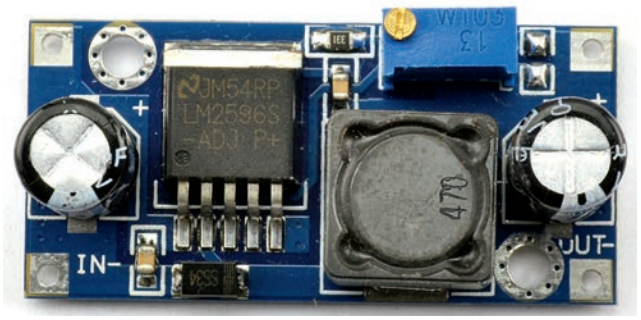


FIGURE 5-9 A cheap switching regulator from eBay

The module shown in Figure 5-9 has four terminals volts in + and – and volts out + and –. The input voltage can be up to 37V and by twiddling the variable resistor you can set the output voltage to anything from 0V to the input minus 1.2V. The module can supply up to 3A.

Boosting Voltage

The voltage regulator ICs in the section titled “Controlling the Voltage from a Battery” only work if the input voltage is greater than the output voltage. In fact, it normally has to be a couple of volts higher, but some more expensive voltage regulators called LDO (low drop out) regulators are available that only require about half a volt more on the input than the output.

Sometimes, however—and cellular phones are a good example of this—it is very convenient to use a single-cell LiPo battery of 3.7V when we require a higher voltage (often 5V) for the circuit.

In these situations, you can employ a very useful circuit called a boost converter. These use an IC and a small inductor (coil of wire) and, by applying pulses to the inductor, produce a higher voltage. Actually, it's more complex than that, but you get the idea.

Buck-boost converters are readily available as modules on well-known auction sites. You can find 1A adjustable modules that will provide an adjustable output of 5V to 25V from 3.7V for a few dollars. Try searching for "Boost Step-Up 3.7V." The main module suppliers also provide such boards for around USD 5.

SparkFun sells an interesting module (see the Appendix, M17) that combines a LiPo battery charger with a buck-boost, so you can both charge your LiPo from an external USB 5V input and use the 3.7V LiPo cell to provide an output of 5V using the buck-boost (Figure 5-10).

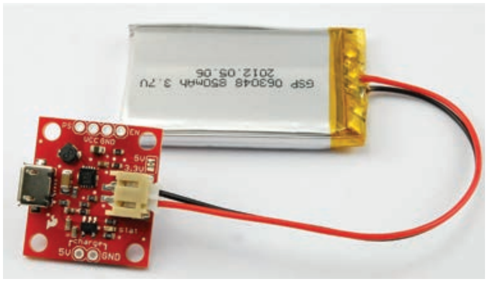


FIGURE 5-10 Combined LiPo charger and booster

This actually takes all the difficulties away from using a LiPo in a situation where you want to charge the LiPo battery in situ. Your 5V microcontroller circuit or whatever you are using is just attached to the VCC and GND connections, the battery is clipped into the socket, and to charge the device, you just plug in a USB cable.

Calculating How Long a Battery Will Last

We have already touched on the capacity of a battery—that is, the number of mAh it can supply. However, other factors come into play that we should think about when deciding if the batteries we are considering for a project are going to last long enough.

It's really just a matter of common sense, but nevertheless it's easy to make false assumptions about what you need.

As an example, I recently built an automated door for my chicken house. It opens at dawn and closes when it gets dark. It uses an electric motor, and electric motors use a lot of current, so I needed to decide what kind of batteries to use. My first thought was to use big D cells or a lead–acid battery. But when it came to do the math, I found this wasn't really necessary.

Although the motor uses 1A each time it is in operation, it is only in operation twice a day, and each time only for about three seconds. I measured the control circuit as using 1mA all the time. So, let's work out how many mAh the control circuit and motors each use in a day, and then see how many days various types of battery will last.

Let's start with the motors:

1A × 3 seconds × 2 = 6As = 6/3600Ah = 0.0016 Ah = 1.6mAh per day

On the other hand, the controller, which I had assumed was the low-power part of the project, would consume

1mA × 24 hours = 24mAh per day

This means we can pretty much ignore the power consumed by the motor since it is less than a tenth of the juice required by the controller. Let's say the total requirement is 25mAh/day.

AA batteries are typically 3000mAh, so if we powered the project from AA batteries, we could expect them to last 3000mAh / 25mAh per day = 120 days.

So we do not really need to look much further, AAs will be fine. In the end, I used solar power for this project, which we will visit again in the section titled "Using Solar Cells," later in this chapter.

Battery Backup

Replacing batteries is a nuisance and expensive, so it is often cheaper and more convenient to power things from a wall-wart power supply. However, this brings its own disadvantages:

- The device is now tethered to a wire.
- If the household electricity fails, the device will stop working.

The best of both worlds can be achieved by arranging for automatic battery backup of a device that is powered by your household electricity. So, both batteries and a power supply are used, but the batteries are only used if the power supply is not available.

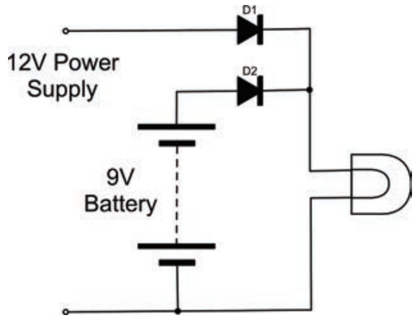
Diodes

What we do not want to happen is for both the batteries and the voltage from the power supply to conflict with each other when both are available. For instance, if the power supply is at a higher voltage than the batteries, it would charge them. But without anything to limit the current, this could be disastrous, even if the batteries were of the rechargeable sort.

Figure 5-11 shows the basic schematic for this. The power supply always needs to be a higher voltage than the battery, so in this case it is 12V and the battery 9V. The schematic also assumes that the battery backup is being used to drive a light bulb.

Recall that diodes act rather like one-way valves. They only allow current to flow in the direction of the arrow. So, let's look at the three possible cases of how power could be supplied here. This is simply the power supply: just batteries and both the battery and power supply (Figure 5-12).

FIGURE 5-11 Battery backup schematic

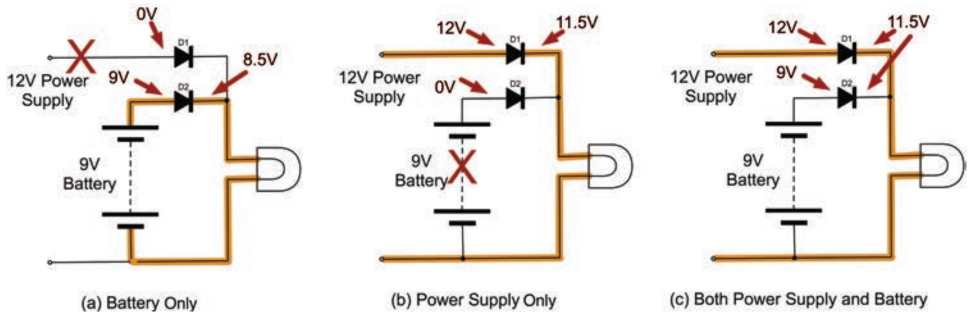


Just the Battery

If only the battery has a voltage greater than zero (in other words, the power supply is not plugged in), then the situation is as shown in Figure 5-12a. The 9V from the battery will be at the anode of D2, and the cathode of D2 will be pulled toward ground by the load of the light bulb. This will cause D2 to be forward-biased and conduct the current through the light bulb. A forward-biased diode will have an almost constant voltage of 0.5V across it, which is why we can say that the voltage after the diode is 8.5V.

On the other hand, D1 will have a higher voltage (8.5V) on its cathode (right-hand side in the diagram) than its anode (0V), so no current will flow through D1.

FIGURE 5-12 Diodes for battery backup



Just the Power Supply

If just the power supply is connected (Figure 5-12b), then the role of the diodes is reversed and now the current flows through D1 to the light bulb.

Both the Power Supply and the Battery

Figure 5-12c shows the situation where both the power supply and the battery are connected. The 12V of the power supply will ensure that the cathode of D2 is at 11.5V. Since the anode of D2 is at 9V from the battery, the diode will remain reverse-biased and no current will flow through it.

Trickle Charging

As we already have a battery and a power supply, we have most of the ingredients we need to charge the battery. We could for example use six AA rechargeable batteries in a battery box and arrange to charge them at C/20 (assuming C = 2000mAh) or 100mA from the power supply.

That way, the batteries would always be charged, and provide light whenever the power failed. Figure 5-13 shows the schematic for this.

You may not have been expecting the extra diode D3. This is really just to account for the fact that we do not know exactly how the power supply is designed, so we do not know what would happen if the battery was connected to its output (via R1) when it was turned off. This may discharge the battery or damage the power supply. The diode D3 just protects it and makes sure no current can flow back into it.

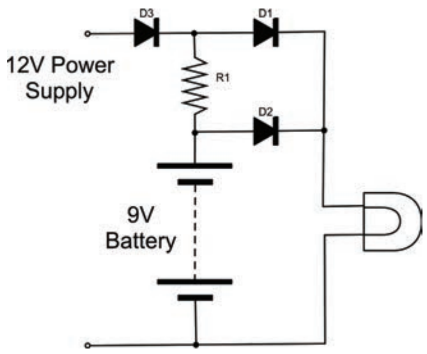


FIGURE 5-13 Battery backup and charging

We want a charging current of 100mA to flow through R1, and we know that when both the power supply and battery are connected, there will be a voltage across R1 of 12V – 0.5V – 9V or 2.5V. So, using Ohm's law, the value of the resistor should be:

$$R = V / I = 2.5 / 0.1A = 25\Omega$$

The nearest standard value to this is probably 27Ω.

Its power requirement: $P = V^2 / R = 2.5^2 / 27 = 0.23W$

This means a standard half- or quarter-watt resistor will be fine.

Using Solar Cells

On the face of it, solar cells seem like the perfect power source. They convert light into electricity, and so in theory you need never change a battery or be plugged into a wall outlet again!

However, as always, the reality is not quite so simple. Solar cells, unless they are very large, produce fairly small amounts of electricity and so are most suited to low-power devices and projects that are outdoors away from household electricity.

If you are thinking of trying a solar project that will be installed indoors, unless it will be sited against a south-facing window, I really wouldn't try it. Solar cells do not require direct sunlight, but to produce any useful amounts of electricity, they really need a good unobstructed view of the sky.

Two solar projects I have developed are a solar-powered radio (the solar panel is as big as the radio and, yes, it needs to be next to the window), and a solar-operated chicken house door. If you are lucky enough to live somewhere sunny, then solar power is obviously a lot easier.

Figure 5-14 shows a typical solar panel. This one was scavenged from a security light installation. It is about six inches by four inches and has a swivel mount that allows it to be angled toward the sun. It is the panel I used for the chicken house door.

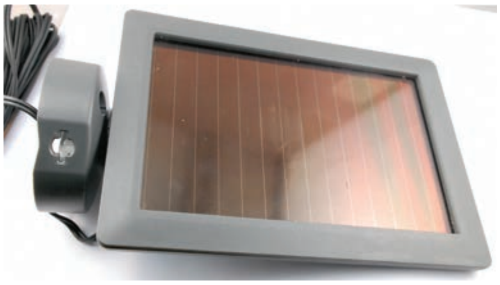


Figure 5-14 A solar panel

Projects that use a solar panel to provide power nearly always also use a rechargeable battery. So the panel charges the battery and the project draws its power from the battery.

Small solar cells generally only produce around half a volt, so they are usually combined into panels of many cells that increase the voltage to a level that is high enough to charge a battery.

The voltage you find on a solar panel normally refers to the voltage of battery that the solar panel is capable of charging. So, it is quite common to find 6V or 12V solar panels. When you measure the voltage from these in bright sunlight, the reading will be much higher, possibly 20V for a 12V panel. But, under the load of charging a battery, this drops rapidly.

Testing a Solar Panel

A solar panel will have a certain number of watts and a nominal voltage specified for it. These tend to be for ideal conditions, so when I get a solar panel that I want to use in a project, I like to test it to find what it is really capable of. Without knowing how much power it can provide in a real situation where it's installed, it is hard to make safe assumptions about battery capacities and how low you need to keep the current consumption.

When testing out a solar panel, you should use a resistor as a "dummy load," and then try out the solar panel in various locations and levels of brightness, measuring the voltage across the resistor. From this, you can calculate the current being provided by the panel.

Figure 5-15 shows such an arrangement for my "chicken house" solar panel. The meter is showing just 0.18V with a 100Ω load resistor inside the light box that I use for my photography. That equates to just 1.8 mA.

I find a spreadsheet a useful way of recording how the solar panel performs. Figure 5-16 shows an excerpt from the spreadsheet, complete with graph. You can then file this away until the next time you wish to use a solar panel in a project.

FIGURE 5-15 Testing a solar panel

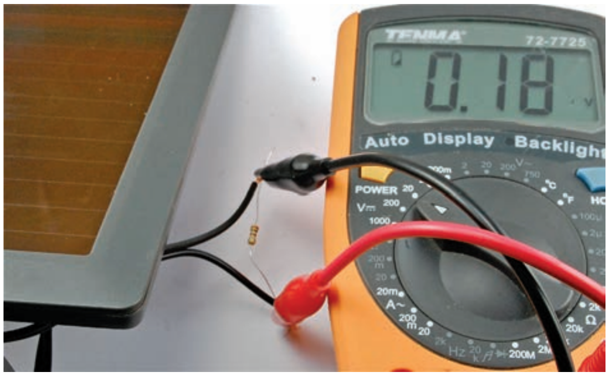
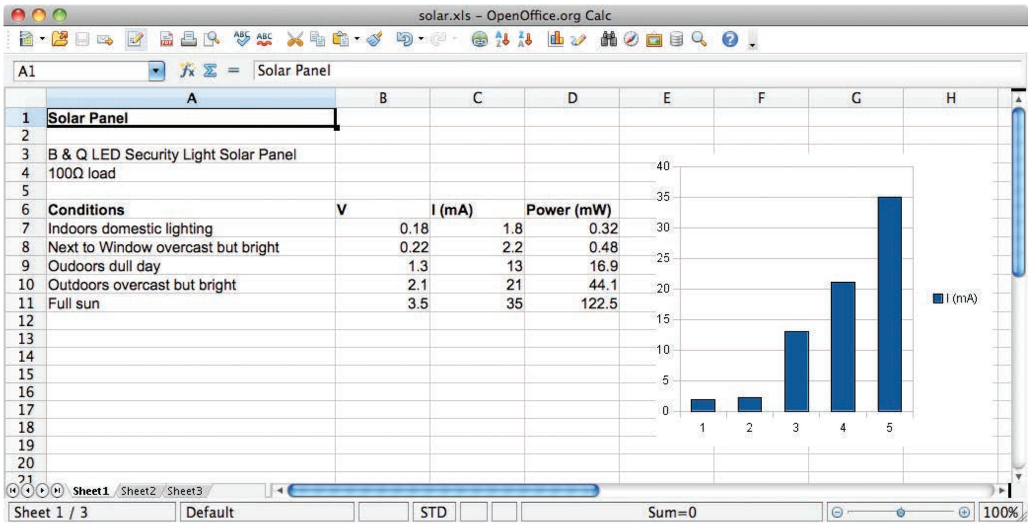


FIGURE 5-16 Solar panel data



solar.xls – OpenOffice.org Calc

A1 Solar Panel

	A	B	C	D
1	**Solar Panel**			
2				
3	B & Q LED Security Light Solar Panel			
4	100Ω load			
5				
6	**Conditions**	**V**	**I (mA)**	**Power (mW)**
7	Indoors domestic lighting	0.18	1.8	0.32
8	Next to Window overcast but bright	0.22	2.2	0.48
9	Oudoors dull day	1.3	13	16.9
10	Outdoors overcast but bright	2.1	21	44.1
11	Full sun	3.5	35	122.5

The spreadsheet can be downloaded from www.hackingelectronics.com, but there is really nothing complex about the math.

As you can see, the solar panel produces only 1 or 2mA indoors even under bright artificial lighting. The results outdoors with a clear view of the sky are better, but it really only produces quite high power in direct sunlight.

Trickle Charging with a Solar Panel

Since the solar panels produce a reasonable voltage, even in relatively low light conditions, they can easily be used to trickle charge a battery. However, you should always use a diode to protect the solar panel from the situation where the battery is at a higher voltage than the panel (say at night), since such a reverse flow will damage the solar panel.

A typical simple trickle charge schematic is shown in Figure 5-17.

Lead–acid batteries are still a very popular choice for trickle charging from solar. This is mainly because they are very forgiving of gentle over-charging and have a lower self-discharge rate than, say, NiMH batteries.

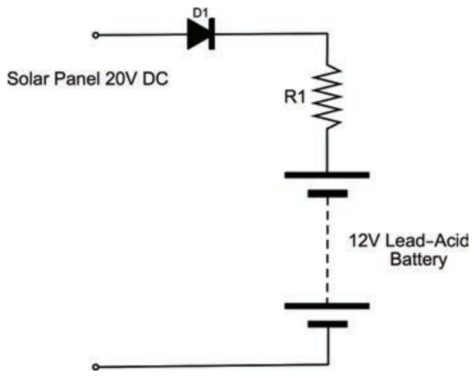


Figure 5-17 Schematic for solar trickle charging

Minimizing Power Consumption

When planning solar power for some small outdoor project, you need to make sure the solar panel charging the battery can keep up with demand.

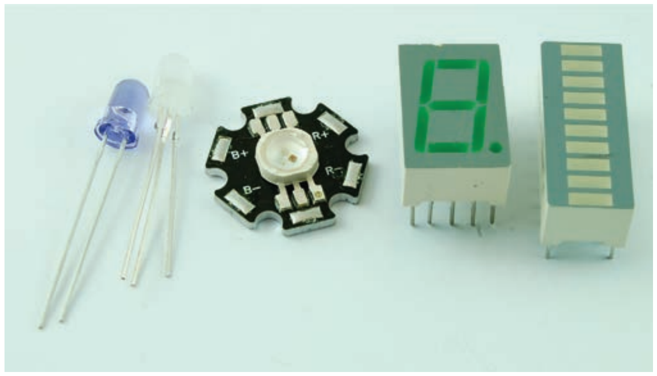
If you live in southern California, the design for using solar panels is pretty easy. You can count on quite a lot of sun all year long. However, if you live a long way from the equator, say, in a maritime climate where it's often quite dull during the day, then you will have short winter days. You may get weeks of dull weather with short days. If your system is to work all year long, you either need to have a large battery that will last for a few weeks of dull weather, or use a larger solar panel.

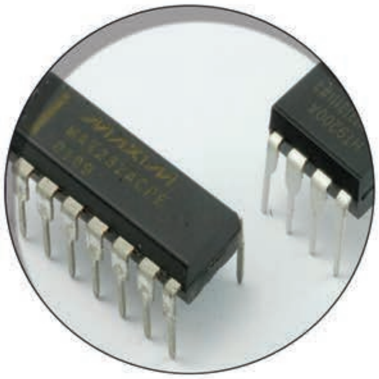
The sums are pretty easy. There are mAh going into the battery from the solar panel, and mAh coming out for the device it is powering. The device might be running all the time, but the solar panel is only active half the time (daylight). So, you need to work out what you think the worst case for solar input might be for a week or two, and then design it accordingly.

It will probably be easier and cheaper to put your efforts into minimizing the current consumed by the system rather than increasing the size of the solar panel and battery.

Summary

In this chapter, we have learned about how to power our projects. In the next chapter, you will learn how to use the very popular Arduino microcontroller board.

ARDUINO
UNO
DIGITAL (PWM~)

B+
R+
R-
B-



6 Hacking with Arduino

Microcontrollers are essentially low-powered computers on a chip. They have input/output pins to which you can attach electronics so the microcontroller can, well, control things. Using a microcontroller used to be quite a complex process, largely because the microcontroller needed to be programmed. This was often done in assembler or complex C. But there was a lot to learn before you could do anything useful. Because of this, it discouraged their use in casual projects where you just wanted to hack something together.

Enter the Arduino (Figure 6-1). The Arduino is a simple-to-use, low-cost, readymade board that lets you use a microcontroller in your projects with a minimum of fuss.

The Arduino sells in vast quantities and has become the platform of choice for makers and hackers in need of microcontrollers.

The popularity of Arduino is due to many factors, including its:

- Low cost
- Open-source hardware design
- Easy-to-use integrated development environment (IDE) to program it with
- Plug-in shields that add features like displays and motor drivers that clip onto the top of the Arduino

All the programs for the Arduino used in this and later chapters are available for download from the book's accompanying Github repository https://github.com/simonmonk/hacking2.

The examples in this book are designed and tested with an Arduino Uno R3. However, two of the projects, in the sections "Typing Passwords Automatically" in this chapter and "Making a USB Music Controller" (see Chapter 10), only work with Arduino Leonardo.

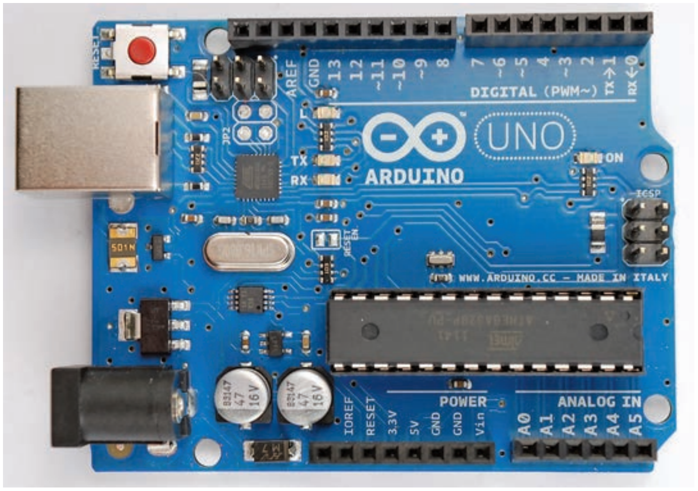


Figure 6-1 An Arduino Uno R3 board

Blinking an LED

To be able to program an Arduino, we first have to install the Arduino integrated development environment (IDE) on our computer. Arduino is available for Windows, Mac, and Linux.

You Will Need

Quantity	Item	Appendix Code
1	Arduino Uno R3	M2
1	Type B USB lead	

Setting Up Arduino

The first step is to download the software for your computer type from the official Arduino web site here: http://arduino.cc/en/Main/Software.

Once this is downloaded, you can find detailed instructions for the installation of each platform here: http://arduino.cc/en/Guide/HomePage.

One of the nice things about the Arduino is that to get started with it, all you need is an Arduino, a computer, and a USB lead to connect the two together. The Arduino can even be powered over the USB connection to the computer. Figure 6-2 shows an

Installing the Book's Software

All the example programs, both for Arduino and Raspberry Pi, are available for download from the GitHub software repository.

Now is probably a good time to download these programs onto your computer. So open the browser on your computer and go to https://github.com/simonmonk/hacking2.

On the right of the page, you will see a green button that says "Clone or Download." Click on this and select the option "Download ZIP." Unzip the archive file when it has downloaded and you will find two folders within it: "pi" and "arduino." Within the "arduino" directory you will find a directory for each of the Arduino programs (or "sketches" as they are called in the Arduino world). Within the directory for a particular project, there will be just one file with the same name as the directory, but with the extension of .ino. If the Arduino IDE is installed correctly, then clicking on one of these .ino files will open the Arduino IDE on the sketch so that you can see the code and edit it.

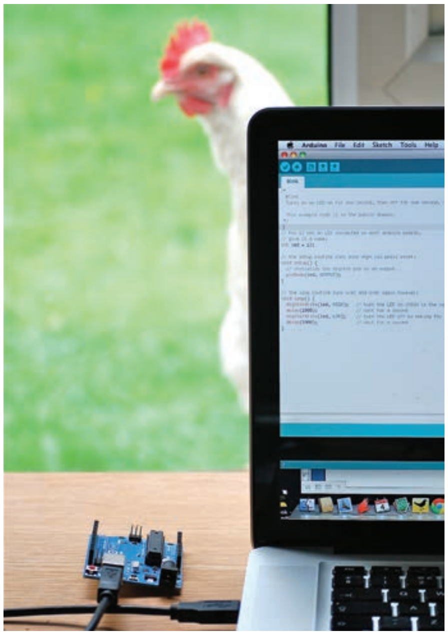


FIGURE 6-2 The Arduino, laptop, and chicken

Arduino Uno (the most common type of Arduino) connected to a laptop running the Arduino IDE.

To prove that the Arduino is working, we will program it to flash an LED that is on the Arduino board labeled "L" and hence known as the "L" LED.

Start by launching the Arduino IDE on your computer. Then, from the File menu (Figure 6-3), select Examples | 01.Basics | Blink.

In an attempt to make programming the Arduino sound less daunting to non-programmers, programs on the Arduino are referred to as "sketches." Before we can send the Blink sketch to your Arduino, we need to tell the Arduino IDE what type of Arduino we are using. The most common type is the Arduino Uno, and in this chapter we will assume that is what you have. So from the Tools | Board menu, select "Arduino Uno/Genuino" (Figure 6-4).

As well as selecting the board type, we also need to select the port it is connected to. In Windows, this will be COM followed by a number and if the board is connected, the type of board (in this case Arduino/Genuino Uno) will be displayed after the port name in parentheses (see Figure 6-5). However, on a Mac or Linux, there

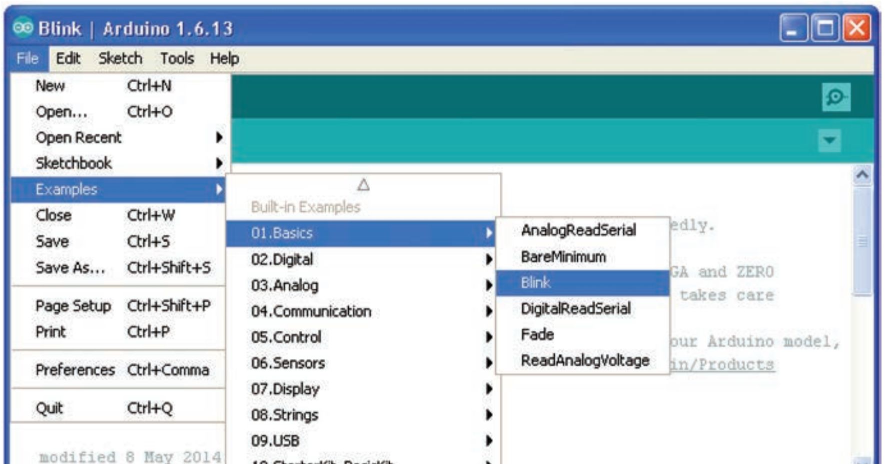


FIGURE 6-3 Loading the "Blink" sketch

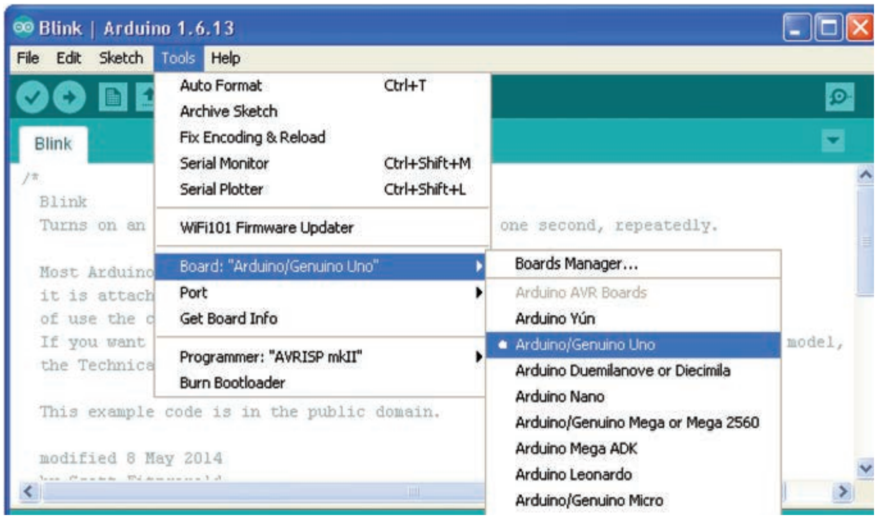


FIGURE 6-4 Selecting the board type

will generally be more serial devices. However, the Arduino IDE will indicate the type of any board connected, making it easy to identify the right port.

To actually upload the sketch onto the Arduino board, click the Upload button on the tool bar. This is the second button on the toolbar, highlighted in Figure 6-6.

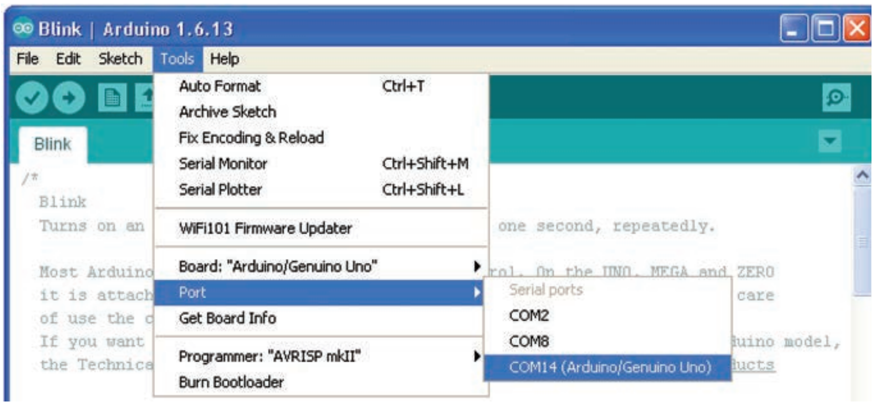
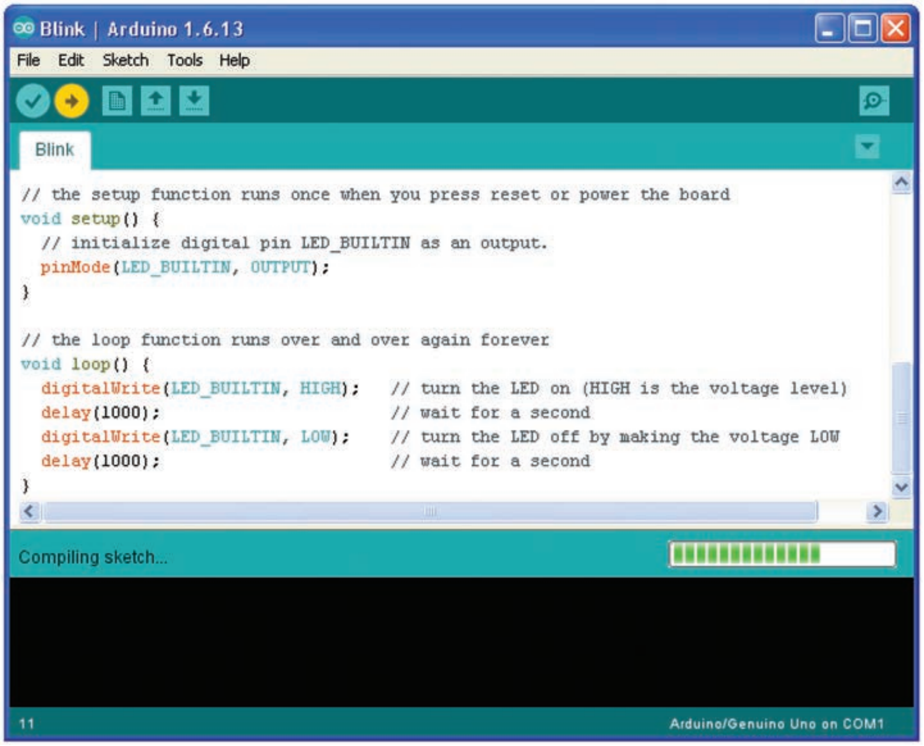

Blink | Arduino 1.6.13
File Edit Sketch Tools Help
Auto Format Ctrl+T
Archive Sketch
Fix Encoding & Reload
Serial Monitor Ctrl+Shift+M
Serial Plotter Ctrl+Shift+L
WiFi101 Firmware Updater
Board: "Arduino/Genuino Uno"
Port
Get Board Info
Programmer: "AVRISP mkII"
Burn Bootloader
Serial ports
COM2
COM8
COM14 (Arduino/Genuino Uno)
Blink

FIGURE 6-5 Selecting the serial port

FIGURE 6-6 Uploading the Blink sketch



```
// the setup function runs once when you press reset or power the board
void setup() {
  // initialize digital pin LED_BUILTIN as an output.
  pinMode(LED_BUILTIN, OUTPUT);
}

// the loop function runs over and over again forever
void loop() {
  digitalWrite(LED_BUILTIN, HIGH);   // turn the LED on (HIGH is the voltage level)
  delay(1000);                       // wait for a second
  digitalWrite(LED_BUILTIN, LOW);    // turn the LED off by making the voltage LOW
  delay(1000);                       // wait for a second
}
```

```
Compiling sketch...
```

Once, you press the Upload button, a few things should happen. First, a progress bar will appear as the Arduino IDE first compiles the sketch (converts it into a suitable form for uploading). Then, the LEDs on the Arduino—labeled Rx and Tx—should flicker for a while.

Finally, the LED labeled L should start to blink. The Arduino IDE will also show a message that looks something like "Binary sketch size: 1,084 bytes (of a 32,256 byte maximum)." This means the sketch has used about 1kB of the 32k of flash memory available for programs on the Arduino.

Note that if you are using a Leonardo, you may have to keep the Reset button depressed until you see the message "Uploading…" in the Arduino software.

Modifying the Blink Sketch

It may be that your Arduino was already blinking when you first plugged it in. That is because the Arduino is often shipped with the Blink sketch already installed.

If this is the case, you might like to prove to yourself that you have actually done something by changing the blink rate. We will now examine the Blink sketch and see how we could change it to make it blink faster.

The first part of the sketch is just a comment, to tell someone looking at the sketch what it is supposed to do. This is not actual program code.

```
/*
  Blink
  Turns on an LED on for one second, then off for one
second, repeatedly.

  ....
 */
```

The next part of the sketch is the "setup" function. Every Arduino sketch must have a "setup" function, and this function runs every time the Arduino is reset, either because (as the comment says) the reset button is pressed, or the Arduino is powered up.

```
// the setup function runs once when you press reset
// or power the board
void setup() {
  // initialize digital pin LED_BUILTIN as an output.
  pinMode(LED_BUILTIN, OUTPUT);
}
```

The structure of this text is a little confusing if you are new to programming. A function is a section of code that has been given a name (in this case, the name is "setup"). For now, simply use the text just cited as a template and understand that it must start with the first line "void setup() {". Afterward, place each of the commands you want to issue on a line ending with ";", and then mark the end of the function with a "}" symbol.

In this case, the only command we expect the Arduino to carry out is to issue the "pinMode(LED_BUILTIN, OUTPUT)" command that not unsurprisingly sets the Arduino pin connected to its built-in LED (13 on an Arduino Uno) to be an output.

Next comes the juicy part of the sketch: the "loop" function.

Like the "setup" function, every Arduino sketch must have a "loop" function. Unlike "setup," which only runs once after a reset, the "loop" function runs continuously. That is, as soon as all its instructions have been done, it starts again.

In the "loop" function, we first turn on the LED by issuing the "digitalWrite(LED_BUILTIN, HIGH)" instruction. We then pause for a second by using the command "delay(1000)". The value is 1000 for 1000 milliseconds, or one second. We then turn the LED back off again, and delay for another second before the whole process starts over.

```
// the loop routine runs over and over again forever:

void loop() {
  digitalWrite(LED_BUILTIN, HIGH); // turn the LED on (HIGH is the voltage level)
  delay(1000);                     // wait for a second
  digitalWrite(LED_BUILTIN, LOW);  // turn the LED off by making the voltage LOW
  delay(1000);                     // wait for a second
}
```

To modify this sketch to make the LED blink faster, change both occurrences of 1000 to 200. These changes are both in the "loop" function, so your function will now look like this:

```
void loop() {
  digitalWrite(LED_BUILTIN, HIGH); // turn the LED on (HIGH is the voltage level)
  delay(200);                      // wait for a second
  digitalWrite(LED_BUILTIN, LOW);  // turn the LED off by making the voltage LOW
  delay(200);                      // wait for a second
}
```

If you try to save the sketch before uploading it, you will be reminded that it is a "read-only" example sketch, but the Arduino IDE will offer you the option to save it as a copy, which you can then modify to your heart's content.

You do not have to do this, of course. You can just upload the sketch unsaved, but if you do decide to save this or any other sketch, you will find that it then appears in the File | Sketchbook menu on the Arduino IDE.

So, either way, click the Upload button again. When the uploading is complete, the Arduino will reset itself and the LED should start to blink much faster.

Controlling a Relay Using an Arduino

The USB connection of an Arduino can be used for more than just programming the Arduino. You can also use it to send data between the Arduino and your computer. If we attach a relay to the Arduino, we could send a command from our computer to turn the relay on and off.

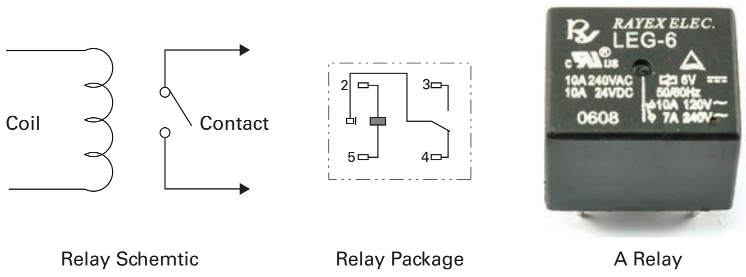
You'll start by learning how to solder up a relay module of your own, but you will also learn how to use a ready-made relay module if you prefer.

Relays

A relay (Figure 6-7) is an electromechanical switch. It's very old technology, but relays are cheap and very easy to use.

A relay is basically an electromagnet that closes switch contacts. The fact that the coil and the contacts are electrically isolated from one another makes relays great for things like switching home-powered devices on and off from something like an Arduino.

FIGURE 6-7 A relay



Whereas the coil of a relay is often energized by between 5V and 12V, the switch contacts can control high-power, high-voltage loads. For example, the relay photographed in Figure 6-7 claims a maximum current of 10A at 120V AC (household power) as well as 10A at 24V DC.

Arduino Outputs

Arduino outputs, and for that matter inputs, are referred to as "pins," even though if you look at the connectors along the sides of the Arduino, they are most definitely sockets rather than pins. The name harkens back to the pins on the microcontroller IC at the heart of the Arduino that were connected to the sockets.

Each of these "pins" can be configured to act as either an input or an output. When they are acting as an output, each pin can provide up to 40mA. This is more than enough to light an LED, but not enough to energize a relay coil, which typically requires more like 100mA.

This is a problem we have already discussed. Since we want to use a small current to control a larger one, a good way to do this is by using a transistor.

Figure 6-8 shows a schematic diagram of what we are going to build.

We are using a transistor just like we did when we were controlling an LED. One difference in the schematic is that there is a diode across the relay coil. This is required because

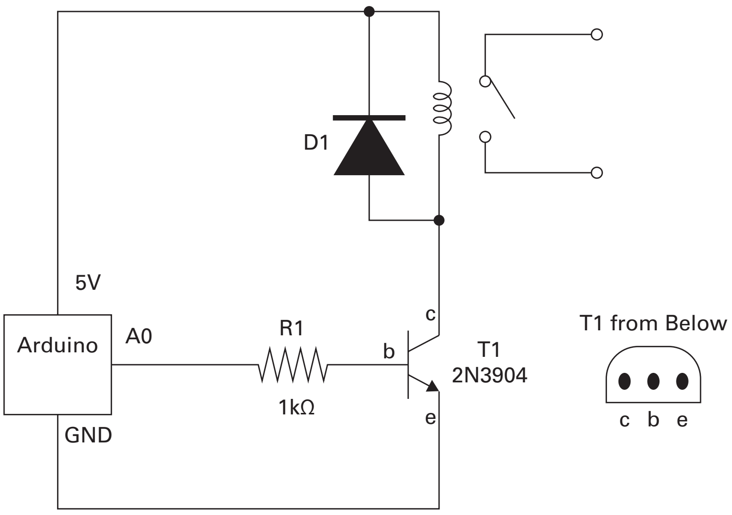


FIGURE 6-8 Schematic diagram of an Arduino-controlled relay

when you turn the relay off and the magnetic field in the coil collapses, you get a spike of voltage. The diode prevents this from damaging anything.

We are going to solder the components to the relay and then attach the necessary leads to a header strip that will plug into the Arduino (Figure 6-9). The header strip has 15 pins and spans both of the connector sockets on the side of the Arduino closer to the microcontroller chip. There is a gap between the two connector strips, so one of the header pins will not actually be fitted into a socket.

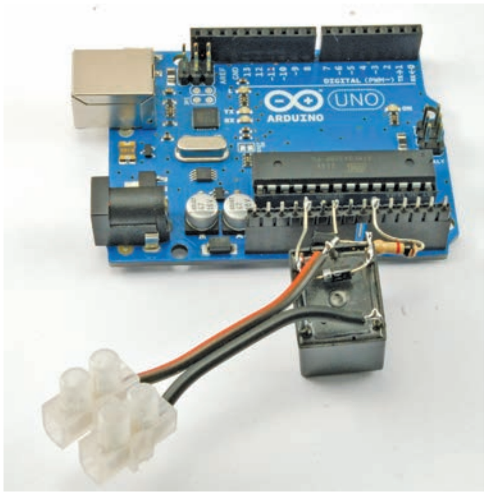


FIGURE 6-9 The Arduino relay interface

You Will Need

Quantity	Names	Item	Appendix Code
1		Arduino Uno R3	M2
1		Transistor 2N3904	K1, S1
1	R1	1kΩ resistor	K1
1	D1	1N4001 diode	K1, S5
1	Relay	5V Relay	K1, H16
1		* Pin header 15-way	K1, H4
1		Two-way screw terminal	K1

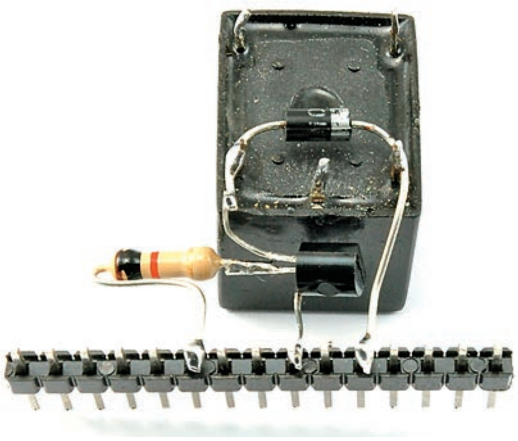
* Pin headers are usually supplied in long lengths designed to be snapped into whatever number of connections you need.

Construction

Figure 6-10 shows how the components are attached. First, solder the diode to the relay coil contacts. These are the two pins on the far side of the relay that have three pins more or less in a row. The stripe on the diode should be to the right, as shown in Figure 6-10.

After soldering the diode across the relay coils, bend out the leads of the transistor and position it as shown in Figure 6-10 with the flat side against the relay. Shorten the base (middle) lead of the transistor, shorten the leads on the resistor and attach it to the base lead.

FIGURE 6-10 Wiring the relay interface



Finally, solder the three leads to the connector strip. The resistor lead should go to the 6th lead from the left, the emitter of the transistor to the 9th from the left, and the diode lead to the 11th from the left.

Before we attach leads to the relay contacts, we can test our work using a multimeter in Continuity mode, so attach the header pins to the Arduino as shown in Figure 6-9 and clip one lead of the multimeter (on Continuity mode) to the middle contact of the relay (in between the diode leads). Attach the other lead of the multimeter to each of the two unconnected contacts on the relay. One will buzz and the other will not. Attach the lead to the one that does not cause the multimeter to beep—this is the n.o. (normally open contact).

Load the sketch "ch06_relay_test" (see the sidebar "Installing the Book's Software" on page 107) into the Arduino IDE and upload it to the Arduino board. When the Arduino restarts, you should find that every two seconds the relay will flip from being open to being closed.

Software

The sketch for this is much the same as the Blink sketch.

```
const int relayPin = A0;

void setup()
{
  pinMode(relayPin, OUTPUT);
}

void loop()
{
  digitalWrite(relayPin, HIGH);
  delay(2000);
  digitalWrite(relayPin, LOW);
  delay(2000);
}
```

The only real difference is that we are using pin A0 rather than LED_BUILTIN (pin 13). Arduino has a feature that allows you to use the analog input pins A0 to A5 as digital inputs or outputs as well as analog inputs, but you have to put the letter A in front of them when using them as digital pins. The pin A0 has been given the name relayPin.

If all is well, then to make it easier to attach things to the relay contacts, we can solder some wires to them and use a two-way screw terminal block (Figure 6-11).

The relay module can be used to control all sorts of things.

In the next section, you will hack an electrical toy so it can be turned on and off using the Arduino and relay module you have just built, or a ready-made relay module.

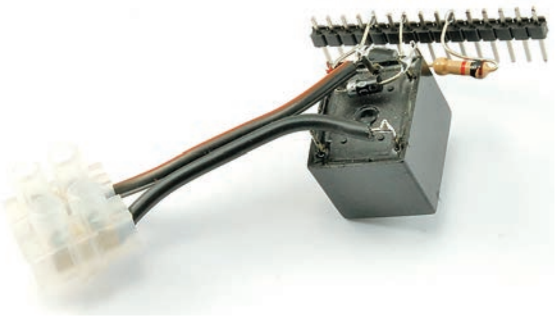


Figure 6-11 Attaching leads to the relay contacts

Figure 6-12 shows how you can connect a ready-made relay module bought for a few dollars to an Arduino. The module's VCC connection is connected to 5V on the Arduino, GND to GND and IN1 of the relay module to A0 of the Arduino.

Some relay modules (including the one shown in Figure 6-12) have inverted logic for the inputs. You can tell if you have this kind of relay module because when you connect up the relay and run ch6_relay_test the relay will click on as soon as the sketch runs but then not turn off. These relay modules switch the relay coil using a PNP transistor that requires the input to be 0V to turn the relay on and the controlling pin to actually be set to be an input to turn the relay off. If you have such a relay, then run the sketch ch6_relay_test_inverted rather than ch6_relay_test.

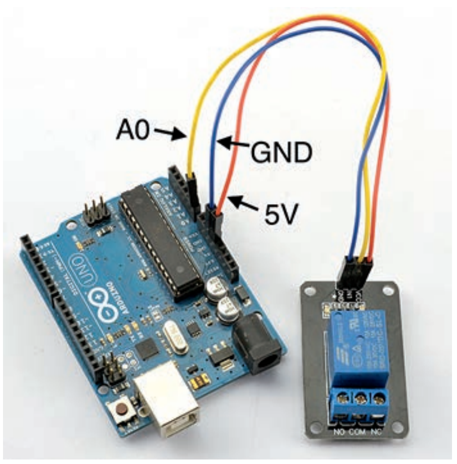


Figure 6-12 Connecting a ready-made relay module to an Arduino

Hacking a Toy for Arduino Control

The great thing about a relay is that it behaves just like a switch. This means that if you have something you want to turn on and off from your Arduino and that item has a switch, then all you need to do is solder some wires to the switch and attach them to the relay. This would allow both the relay and the switch to turn the device on and off. But if you do not want to keep the original switch, it can be removed, as it will be in this case.

The toy that the author chose to hack is a little electric bug (Figure 6-13).

Figure 6-13 The hapless electric bug awaiting dissection

You Will Need

As well as the relay module built in the section "Controlling a Relay Using an Arduino," or a ready-made relay module, you will also need the following items.

Quantity	Item	Appendix Code
1	Arduino Uno R3	M2
1	An electric toy (battery-powered) with an on–off switch	
1	Twin multi-core wire	

Construction

Taking the toy apart, you can see the connections to the switch (Figure 6-14a). De-solder the switch and attach wires to the leads that used to go to the switch (Figure 6-14b). You should always put insulating tape around the bare wires to prevent accidental shorts (Figure 6-14c).

The toy can then be assembled with the wire leaving through a gap in the case (Figure 6-14d). If there is no suitable gap, you will probably need to drill a hole.

Finally, the toy is ready to use, so plug the relay interface into the Arduino and connect the wires to its screw terminals (Figure 6-14e). If the test sketch is still installed, you should find that the toy repeatedly turns on for a couple of seconds, and then turns back off again.

This is okay, but not terribly useful. We will use another sketch that will allow us to send commands to the Arduino from your computer. The sketch is called "ch_06_relay_remote," but if you are using a ready-made relay module that has an inverting input, run "ch06_relay_remote_inverted" instead.

Upload this sketch to the Arduino. Then, open the Serial Monitor by clicking the button on the right-hand side of the Arduino IDE (circled in Figure 6-15).

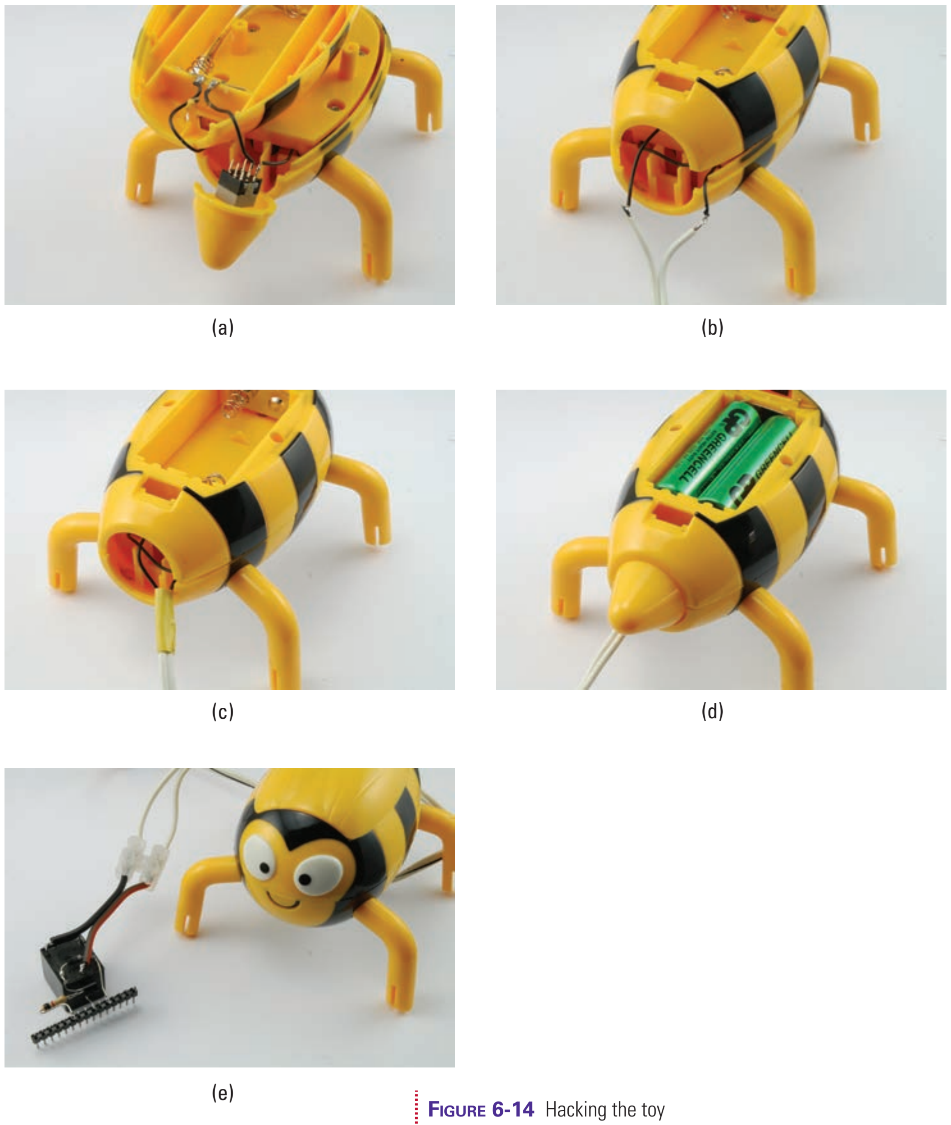


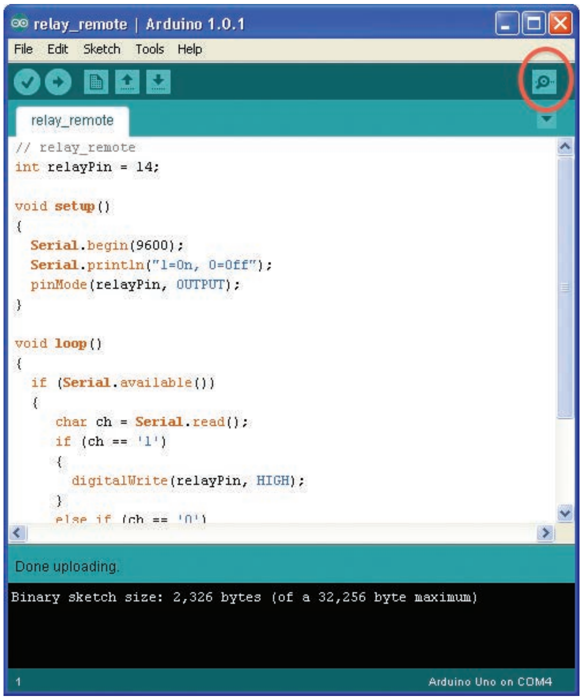
FIGURE 6-14 Hacking the toy

The Serial Monitor

The Serial Monitor is part of the Arduino IDE that allows you to send and receive data between your computer and the Arduino board (Figure 6-16).

At the top of the Serial Monitor is an area where we can type commands. When we click the Send button, these are

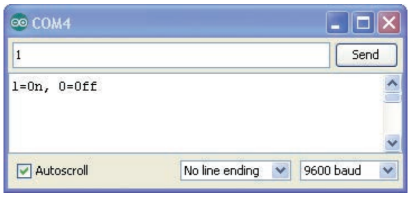
FIGURE 6-15 Opening the Serial Monitor



```
// relay_remote
int relayPin = 14;

void setup()
{
  Serial.begin(9600);
  Serial.println("1=On, 0=Off");
  pinMode(relayPin, OUTPUT);
}

void loop()
{
  if (Serial.available())
  {
     char ch = Serial.read();
     if (ch == '1')
     {
       digitalWrite(relayPin, HIGH);
     }
     else if (ch == '0')
```

Done uploading.

```
Binary sketch size: 2,326 bytes (of a 32,256 byte maximum)
```



```
1
```

```
1=On, 0=Off
```

FIGURE 6-16 The Serial Monitor

sent to the Arduino. We can see any messages that the Arduino has sent in the area below this.

Try this out by typing in the number 1 and clicking Send. This should start your toy. Entering “0” should turn it off again.

Software

Let’s now look at the sketch.

```
const int relayPin = A0;

void setup() {
  Serial.begin(9600);
  Serial.println("1=On, 0=Off");
  pinMode(relayPin, OUTPUT);
}

void loop() {
  if (Serial.available()) {
     char ch = Serial.read();
```

```
      if (ch == '1') {
        digitalWrite(relayPin, HIGH);
      }
      else if (ch == '0') {
        digitalWrite(relayPin, LOW);
      }
    }
  }
```

Notice that the "setup" function now has two new commands in it.

```
  Serial.begin(9600);
  Serial.println("1=On, 0=Off");
```

The first of these starts serial communications over the serial port at 9600 baud. The second sends the prompt message, so that we know what to do when the Serial Monitor opens.

The "loop" function first uses the function "Serial.available()" to check if there is any communication from the computer waiting to be processed. If there is, then this is read into a character variable.

We then have two if statements. The first checks to see if the character is a "1", and if it is, it turns on the toy. If, on the other hand, the character read is a "0", it turns it off.

We have made a little bit of a leap from our first flashing sketch, and if you need more help understanding how the sketch works, you might consider buying the book *Programming Arduino: Getting Started with Sketches* by this author.

Measuring Voltage with an Arduino

The pins labeled A0 to A5 on an Arduino can be used as analog inputs. That is, you can use them to measure voltage. To demonstrate this, you will use the variable resistor (trimpot) as a voltage divider connected to A3 (Figure 6-17).

If you skipped the section on voltage dividers in Chapter 3

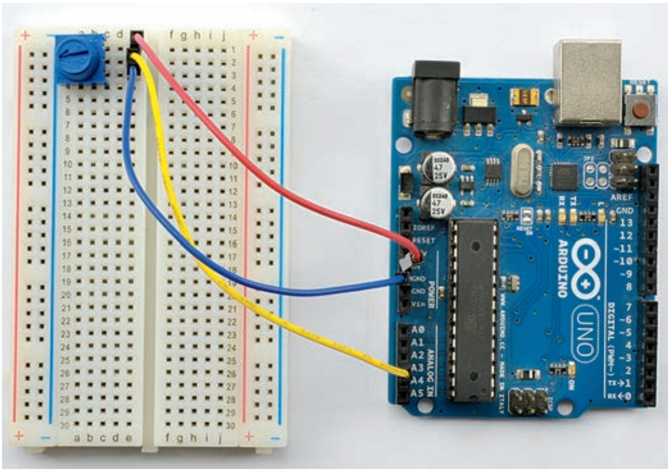


Figure 6-17 A variable resistor and Arduino

titled “Using Resistors to Divide a Voltage,” you probably should go back and have a quick look now.

You Will Need

To try this example, you will need the following items.

Quantity	Names	Item	Appendix Code
1		Arduino Uno R3	M2
1		Solderless breadboard	T5, K1
3		Male-to-male jumper wire or solid core wire	T6
1	R1	10kΩ trimpot variable resistor	K1, R1

Construction

The construction of this project is very simple. There is no actual soldering involved; we will just push the three pins of the variable resistor into the breadboard and connect the “top” of the pot to 5V, the bottom to GND and the slider to A3 as shown in Figure 6-17. Figure 6-18 shows the schematic diagram for this.

As you turn the pot’s knob from the GND (0V) end towards the 5V end of its travel, the voltage at the slider will vary between 0 and 5V. In the Arduino sketch that follows, you will be able to see the voltage reading at A3 displayed in the Serial Monitor.

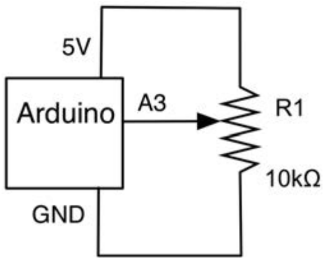


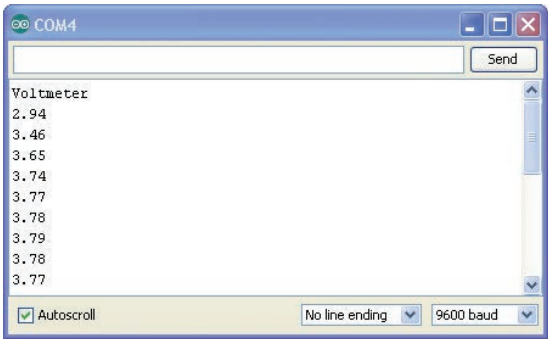
FIGURE 6-18 Schematic diagram—measuring voltage with an Arduino

Software

Load the sketch “ch6_voltmeter” into the Arduino IDE and then program your Arduino board with it. Open the Serial Monitor and you should see something like Figure 6-19.

Try twiddling the knob from one end of the range to the other. You should find that you can set the voltage to anything between 0 and 5V.

FIGURE 6-19 The Serial Monitor showing voltage at A3



```
Voltmeter
2.94
3.46
3.65
3.74
3.77
3.78
3.79
3.78
3.77
```

```
const int voltsInPin = A3;
void setup() {
  Serial.begin(9600);
  Serial.println("Voltmeter");
}
void loop() {
  int rawReading = analogRead(voltsInPin);
  float volts = rawReading / 204.6;
  Serial.println(volts);
  delay(200);
}
```

The sketch defines A3 to be voltsInPin. Note that when referring to the analog input pins for use as analog inputs (as with "voltsInPin"), you can just use the pin number. So for A3 you could have used just 3. However, to distinguish the "A" pins from the other digital pins, it is a good idea to add the letter A to the front.

The "setup" function starts serial communication and sends a welcome message.

Inside the loop, we use "analogRead" to give us a raw value of between 0 and 1023, where 0 means 0V and 1023 means 5V. To convert this into an actual voltage, we need to divide it by 204.6 (1023/5). In dividing a raw reading that is an integer (whole number) by a decimal number of 204.6 (called floats in Arduino), the result will be a float, and so we specify the type of the "volts" variable to be "float".

Finally, we print out the voltage and then wait 200 milliseconds before we take the next reading. We don't have to wait before taking the next reading, it just stops the readings from flying up the screen too fast to read.

In the next section, we will use the same hardware with the addition of an external LED and a slightly different sketch to change the rate at which an external LED flashes.

Controlling an LED with an Arduino

There are three useful things to be learned here. The first is how to make an Arduino drive an LED. The second is how to control the rate of flashing using a reading from a variable resistor, and finally we will show how to use the Arduino to control the power going to the LED and thus determine its brightness (Figure 6-20).

You Will Need

To try this example, you will need the following items.

Quantity	Names	Item	Appendix Code
1		Arduino Uno R3	M2
1		Solderless breadboard	T5, K1
3		Male-to-male jumper wire or solid core wire	T6
1	R1	10kΩ trimpot variable resistor	K1, R1
1	R2	270Ω Resistor	K1
1	D1	LED	K1

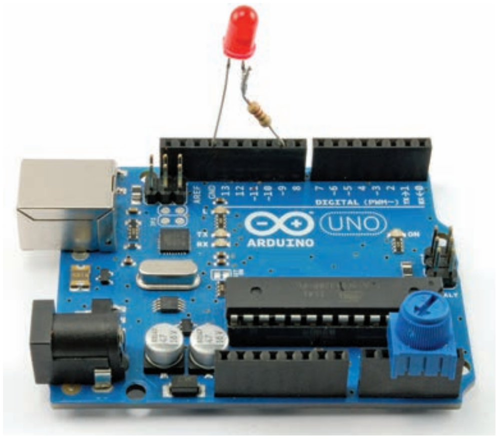


FIGURE 6-20 An Arduino, variable resistor, and LED

Construction

As we discussed in Chapter 4, LEDs need a resistor to stop them drawing too much current. This means we cannot just connect one directly to the output pin of an Arduino. Push the LED leads into the breadboard remembering to place the shorter negative lead on the same row as the ground connection to the pot. One end of the resistor should go to the other positive lead of the LED and the other should be linked by a jumper wire to Arduino pin 9. The connections to the pot remain unchanged. Note that you don't have to use exactly the same rows of the breadboard as Figure 6-20, just so long as things that should be connected together are on the same row as each other.

The schematic diagram for the arrangement is shown in Figure 6-21.

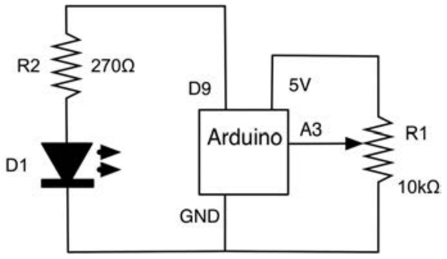


FIGURE 6-21 Schematic for an LED, Arduino, and a variable resistor

Software (Flashing)

You will use two different sketches with this arrangement of hardware. The first uses the variable resistor to control the speed of the flashing, while the second will control the brightness of the LED.

Load the sketch “ch06_variable_led_flash” onto your Arduino board. You should find that turning the knob controls the rate at which the LED flashes.

```
const int voltsInPin = A3;
const int ledPin = 9;

void setup() {
  pinMode(ledPin, OUTPUT);
}

void loop() {
  int rawReading = analogRead(voltsInPin);
  int period = map(rawReading, 0, 1023, 100, 500);
  digitalWrite(ledPin, HIGH);
  delay(period);
  digitalWrite(ledPin, LOW);
  delay(period);
}
```

The sketch is quite similar to that in the previous section; however, we no longer use the Serial Monitor, so all that code is gone. We do need to define a new pin "ledPin" to use for the LED.

The "loop" function still reads the raw value from the analog pin A3, but it then uses the "map" function to convert the "rawReading" value of between 0 to 1023 to a range of 100 to 500.

The "map" function is a standard Arduino command that adjusts the range of the value passed in as the first parameter. The second and third parameters are the range of the raw value, while the fourth and fifth are the desired range you want to compress or expand the value into.

We then flash the LED using this number (100 to 500) as the delay between turning the LED on and off. The end result of this is that the LED will flash faster the closer A3 is to 0V.

Software (Brightness)

We can use exactly the same hardware, but with different software to control the brightness of the LED instead of its rate of flashing. This will use the Arduino "analogWrite" function to vary the power going to the pin. This feature is only available for those pins marked with a "~" on the Arduino board. Fortunately, we thought ahead and chose such a pin to connect the LED to.

These pins can use a technique called pulse-width modulation (PWM) to control how much power goes to the output. This works by sending out a series of pulses, around 500 times per second. These pulses may be high for only a short time, in which case little power is delivered, or high until it's nearly time for the next pulse, in which case lots of power is delivered.

In the case of the LED, this means that in each cycle, the LED is either off, on for some of the time, or on the whole time. Our eyes cannot keep up with such a fast-changing event, so it just appears that the brightness of the LED varies.

Load the sketch “ch06_led_brightness” onto your Arduino. You should find that, now, the variable resistor controls the brightness of the LED rather than its rate of flashing.

Most of the sketch is the same as the previous one; the difference lies in the “loop” function.

```
void loop()
{
  int rawReading = analogRead(voltsInPin);
  int brightness = rawReading / 4;
  analogWrite(ledPin, brightness);
}
```

The function “analogWrite” expects a value between 0 and 255, so we can take our raw analog reading of between 0 and 1023 and divide it by 4 to put it into roughly the right range.

Playing a Sound with an Arduino

The first Arduino sketch we tried at the start of this chapter flashed an LED on and off. If we turn a digital output pin on and off much faster than this, we can drive a sounder to create a sound. Figure 6-22 shows a simple sound generator that plays one of two notes when a button is pressed.

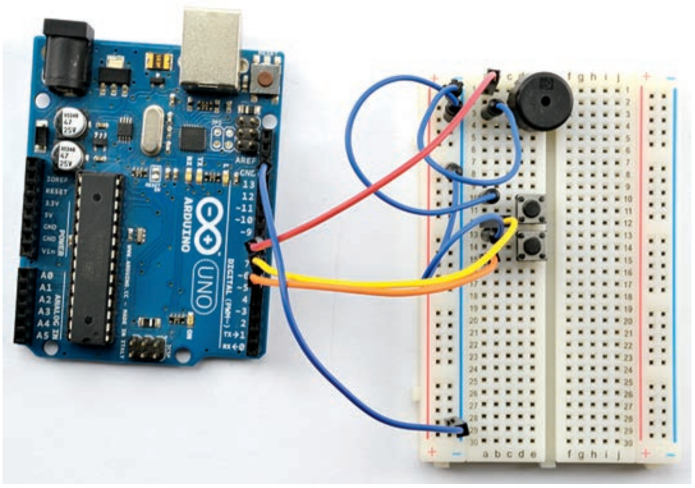


FIGURE 6-22 A simple Arduino tone generator

You Will Need

To have your Arduino make sounds, you will need the following items.

Quantity	Names	Item	Appendix Code
1		Arduino Uno R3	M2
2	S1, S2	Miniature push switches	K1
1	Sounder	Small piezo sounder	M3
1		Solderless breadboard	K1, T5
7		Male-to-male jumper wire or solid core wire	K1, T6

Construction

Figure 6-23 shows the schematic for the tone generator, while Figure 6-24 displays the breadboard layout.

Make sure the push switches are the right way around. If the switches have 4 pins, they should be positioned so the pins extend out of the sides rather than the top and bottom. The piezo sounder may have between 2 and 4 holes between the pins, so make sure that the jumper wires line up with the rows into which the buzzer pins are inserted.

Attach the components as shown and connect the jumper leads to the Arduino.

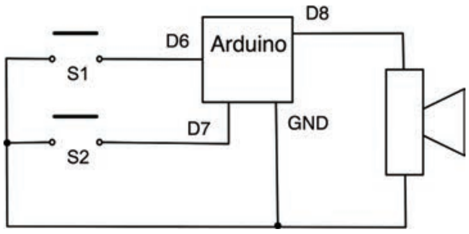


FIGURE 6-23 Schematic diagram for the tone generator

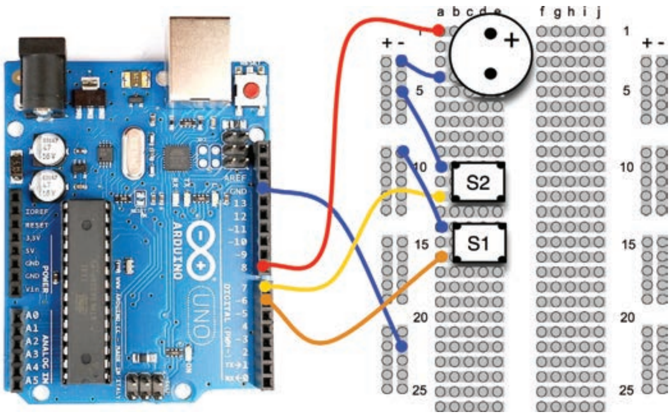


FIGURE 6-24 Breadboard layout for the tone generator

Software

The sketch, “ch06_sounds”, is quite straightforward and follows what should be a familiar pattern by now.

```
const int sw1pin = 6;
const int sw2pin = 7;
const int soundPin = 8;

void setup() {
  pinMode(sw1pin, INPUT_PULLUP);
  pinMode(sw2pin, INPUT_PULLUP);
  pinMode(soundPin, OUTPUT);
}

void loop() {
  if (! digitalRead(sw1pin)) {
    tone(soundPin, 220);
  }
  else if (! digitalRead(sw2pin)) {
    tone(soundPin, 300);
  }
  else {
    noTone(soundPin);
  }
}
```

First, we define the variables for the pins. The switches will be connected to "sw1pin" and "sw2pin". These will be digital inputs, while the "soundPin" will be a digital output.

Note that in the setup function for the switch pins, we use the command "pinMode" with the parameter INPUT_PULLUP. This sets the pin to be an input, but also enables a "pull-up" resistor built into the Arduino, which keeps the input pin HIGH, unless we pull it LOW by pressing the button.

It is because the input pins are normally high that in the "loop" function, when we are checking to see if a button is pressed, we have to use the "!" (logical not). In other words, the following will only sound the tone if the digital input pin "sw1pin" is LOW.

```
if (! digitalRead(sw1pin))
{
  tone(soundPin, 220);
}
```

The "tone" function is a useful built-in Arduino function that plays a tone on a particular pin. The second parameter is the frequency of the tone in Hertz (cycles per second).

If no key is pressed, then the function "noTone" is called and stops any tone that is playing.

Using Arduino Shields

The success of Arduino had been in no small part due to the wide range of plug-in shields that add useful features to a basic Arduino board. A shield is designed to fit into the header sockets of the main Arduino board. Most shields will then pass through these connections in another row of header sockets, making it possible to construct stacks of shields with an Arduino at the bottom. Shields that have, say, a display on them, will not normally pass through in this way. You also need to be aware that if you stack shields like this, you need to make sure there are no incompatibilities, such as two of the shields using the same pin. Some shields get around this problem by providing jumpers to add some flexibility to pin assignments.

There are shields available for almost anything you could want an Arduino to do. They range from relay control to LED displays and audio file players.

Most of these are designed with the Arduino Uno in mind, but are also usually compatible with the bigger Arduino Mega and the newer Arduino Leonardo.

An encyclopedic list, that includes useful technical details about the pin usage of these shields can be found at http://shieldlist.org/.

Some of the author's favorite shields are listed in Table 6-1.

Controlling a Relay from a Web Page

The Arduino Uno does not have a WiFi interface and although there are versions of the Arduino that do have WiFi, as well as a WiFi shield for the Arduino, these are very expensive and writing software for them is by no means straightforward.

Fortunately, a low-cost, easy-to-use solution is at hand in the form of a NodeMCU board as shown in Figure 6-25, where it is attached to a ready-made relay module. You can also see

Shield	Description	URL
Motor	Ardumoto shield. Dual H-bridge bidirectional motor control at up to 2A per channel.	www.sparkfun.com/products/9815
Ethernet	Ethernet and SD card shield.	http://arduino.cc/en/Main/ArduinoEthernetShield
Relay	Controls four relays. Screw terminals for relay contacts.	www.robotshop.com/seeedstudio-arduino-relay-shield.html
LCD	16 × 2 character alphanumeric LCD shield with joystick.	www.freetronics.com/products/lcd-keypad-shield

TABLE 6-1 Some Commonly Used Shields

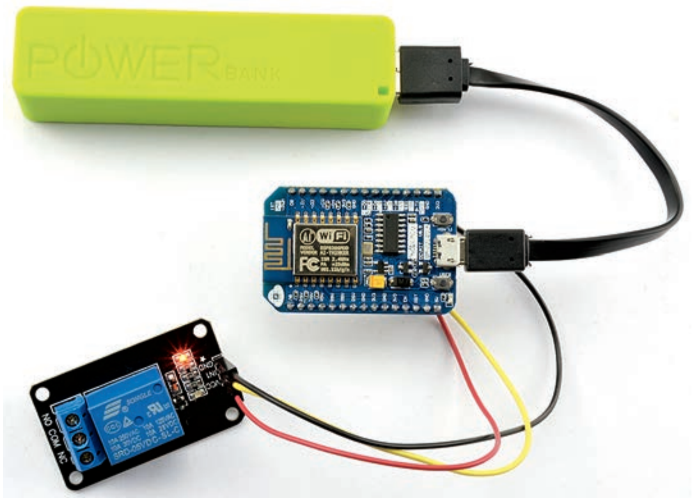


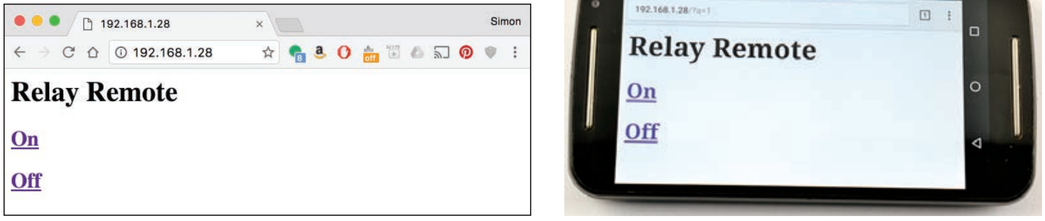
FIGURE 6-25 A NodeMCU board connected to a relay module

that the NodeMCU board is powered by USB cable to a 5V backup battery.

Figure 6-26 shows the browser window used to control the relay, first on a desktop computer (a) and then on a smartphone (b).

The ESP8266 at the center of the NodeMCU board is a WiFi System on a Chip. That is it's a single chip that pretty much does everything that an Arduino Uno equipped with a WiFi shield could do. It includes a few GPIO pins and an analog input and can be programmed from the Arduino IDE as if it were an official Arduino board.

The NodeMCU board actually comes supplied with its own firmware that uses the Lua programming language rather than Arduino C, however, thanks to the efforts of the ESP8266



Relay Remote

On

Off

(a) (b)

FIGURE 6-26 A web interface to control a relay (a) computer browser and (b) smartphone browser

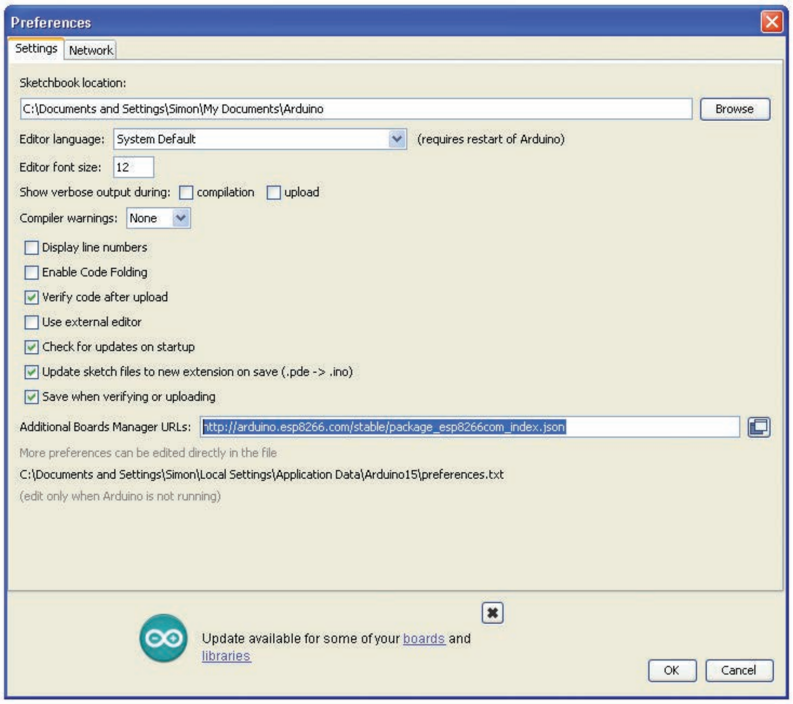
community, this firmware can be replaced and the board programmed almost like any other Arduino board. We can program this board to act as a web server and connect a relay module to one of the NodeMCU's GPIO pins.

The first step in using a NodeMCU is to update your Arduino IDE so that it is aware of this new type of board. You must be using Arduino 1.6 or later for the following instructions to work.

First open the Arduino IDE Preferences window from the File menu and add the address http://arduino.esp8266.com/stable/package_esp8266com_index.json to the Additional Board Manager URLs field (Figure 6-27).

Open the Arduino IDE's Boards Manager window, which you will find under the Tools –> Boards menu option. Scroll down to the bottom of the list and click on the Install button next to the "esp8266 by ESP8266 Community" item.

Close the Boards Manager and now, when you look at the list of possible boards you will find some new board options for ESP8266 related boards, in particular "NodeMCU0.9" and


Preferences

Settings | Network

Sketchbook location:
C:\Documents and Settings\Simon\My Documents\Arduino
Browse

Editor language: System Default (requires restart of Arduino)

Editor font size: 12

Show verbose output during: ☐ compilation ☐ upload

Compiler warnings: None

☐ Display line numbers
☐ Enable Code Folding
✓ Verify code after upload
☐ Use external editor
✓ Check for updates on startup
✓ Update sketch files to new extension on save (.pde -> .ino)
✓ Save when verifying or uploading

Additional Boards Manager URLs: http://arduino.esp8266.com/stable/package_esp8266com_index.json

More preferences can be edited directly in the file
C:\Documents and Settings\Simon\Local Settings\Application Data\Arduino15\preferences.txt
(edit only when Arduino is not running)

Update available for some of your boards and libraries

OK Cancel

FIGURE 6-27 Adding a board manager URL for ESP8266 boards

"NodeMCU 1.0". When you buy your ESP8266 board you will need to check which of these two types it is.

Before you can start programming your NodeMCU board, you will need to install drivers for the USB to serial chip. This is not the same chip as is used for an Arduino Uno's USB to serial interface, so you will need to download and install the drivers for your platform from https://github.com/nodemcu/nodemcu-devkit/tree/master/Drivers and then run the installer.

With the board connected to your computer with a USB lead, you should now see a new port in the Arduino IDE's Ports menu option.

Select the board type (NodeMCU0.9 or NodeMCU 1.0) and port as you would any other Arduino and you should now be able to program the ESP8266. There are, however, a few differences from standard Arduino C that you should be aware of:

- Sometimes when programming the board (but not always) you will need to hold down the Flash button on the board before you power it up, and only release the button a few seconds after the board has been powered up. Do this if your sketch fails to upload.
- All digital inputs and outputs to the NodeMCU board are 3.3V, NOT 5V. Connecting 5V to a NodeMCU pin is likely to damage it.
- When referring to the pins D0 to D8 in a sketch, these pins must always be used with the D in front of them, for example, pinMode(D0, OUTPUT). This "D" is optional for the official Arduino boards.
- The NodeMCU has a built-in LED like the Arduino Uno's L LED, but it is on pin D0 rather than pin D13, so you will need to modify your blink sketch to use pin D0.

You Will Need

To make this project, you will need the following items:

Quantity	Item	Appendix Code
1	NodeMCU board	M4
1	Relay module	M6
3	Female-to-female jumper wires	K1, T14

Connect up your relay module to the NodeMCU board using the female-to-female jumper wires as follows:

- GND to GND
- VCC on the relay module to 5V on the NodeMCU board
- IN1 on the relay module to D0 on the ModeMCU board

Open the sketch "ch06_web_relay" in the Arduino IDE. If you have a relay module with an inverting input, then use "ch06_web_relay_inverting". Modify the values of ssid (network name) and password to match the login credentials for your WiFi router. Make sure that you still have the right board type and port for your NodeMCU board and upload the sketch. This will take a lot longer than for an Arduino Uno, and during the process of uploading, you will see a row of dots appear in the console of the IDE to show the upload progress (see Figure 6-28).

Once the sketch is running you need to be able to find out what internal IP address your network has allocated to the NodeMCU. To do this, open the Serial Monitor. As well as allowing you to find the IP address allocated to the NodeMCU, this will also allow you to check that the NodeMCU is correctly connecting to your network. The output from the Serial Monitor should look something like this:

```
Connecting to My Network
......
WiFi connected
IP address: 192.168.1.28
HTTP server started
```

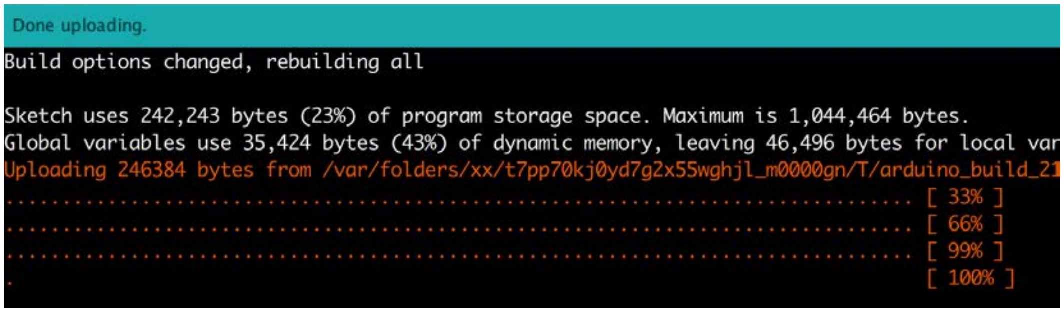

```
Done uploading.
Build options changed, rebuilding all

Sketch uses 242,243 bytes (23%) of program storage space. Maximum is 1,044,464 bytes.
Global variables use 35,424 bytes (43%) of dynamic memory, leaving 46,496 bytes for local var
Uploading 246384 bytes from /var/folders/xx/t7pp70kj0yd7g2x55wghjl_m0000gn/T/arduino_build_21
................................................................................ [ 33% ]
................................................................................ [ 66% ]
................................................................................ [ 99% ]
.                                                                                [ 100% ]
```

FIGURE 6-28 Uploading a sketch to NodeMCU

You can now see that the IP Address for the NodeMCU allocated by my router is 192.168.1.28, so if I type that into the address field of a browser on any computer, smartphone, or tablet on my network, the NodeMCU will serve up a web page for me, as shown in Figure 6-26, that allows me to turn the relay on and off.

The code for "ch06_web_relay" is listed below:

```
#include <ESP8266WiFi.h>
#include <WiFiClient.h>
#include <ESP8266WebServer.h>
#include <ESP8266mDNS.h>

const char* ssid = "Linda-and-Simon";
const char* password = "EP8KQG9D";
const int relayPin = D0;

ESP8266WebServer server(80);

void setup() {
  pinMode(relayPin, OUTPUT);
  Serial.begin(9600);
  connectToWiFi();
  server.on("/", handleRoot);
  server.begin();
  Serial.println("HTTP server started");
}

void loop() {
  server.handleClient();
}

void connectToWiFi() {
  Serial.print("\n\nConnecting to ");
  Serial.println(ssid);
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  Serial.println("\nWiFi connected");
  Serial.print("IP address: ");
  Serial.println(WiFi.localIP());
}


void handleRoot() {
  Serial.println("Got a Request");
  if (server.arg(0)[0] == '1') {
    digitalWrite(relayPin, HIGH);
  }
  else {
    digitalWrite(relayPin, LOW);
  }
  String msg = "";
  msg += "<html><body>\n";
  msg += "<h1>Relay Remote</h1>";
  msg += "<h2><a href='?a=1'/>On</a></h2>";
  msg += "<h2><a href='?a=0'/>Off</a></h2>";
  msg += "</body></html>";

  server.send(200, "text/html", msg);
}
```

The *setup* function sets the relay pin to be an output, starts serial communication and then calls the function *connectToWiFi*. It also specifies a handler function *handleRoot* to be called any time anyone makes a web request to the server.

The loop function calls *handleClient* on the server, which waits for incoming requests and then services them.

The process of creating the WiFi connection, along with code to display the IP address of the server in the Serial Monitor, is all contained in the *connectToWiFi* function. This reports the progress in connecting to the WiFi, which will take a few seconds.

The *handleRoot* function demonstrates a nice feature of the ESP8266WiFi library. That is, the ability to define handlers for different pages that the server is serving. Remember that in the *setup* function, there is the command *server.on("/", handleRoot)*. This tells the server that whenever there is a request for the root page (/) the function *handleRoot* should be called to generate the necessary HTML for that page and send it back to the browser. The *handleRoot* function reads the first letter of the first request parameter (*server.arg(0)[0]*) and if it equals '1' the relay is turned on; otherwise, it is set off.

This function uses the Arduino *String* class to construct the HTML a line at a time. The HTML that it generates includes web requests back to the same page but with an extra request parameter called “a” that has a value of either 1 or 0 to turn the relay on or off respectively.

Switching with an Arduino and Transistor

In this chapter when it comes to switching things on and off, we have been using a relay. Electromechanical relays are very old-fashioned components and have the advantage of being easy to use and from the point of view of the thing being controlled, they behave just like a simple switch. The only real down-side of using a relay is that because they have moving parts, they will eventually break. This makes them unsuitable for rapid on/off switching.

Back in Chapter 3, you saw how a MOSFET transistor could be used to switch a motor on and off by controlling the voltage to the transistor’s gate. If you connect the gate of the MOSFET to an Arduino digital output, you can use the Arduino to switch the MOSFET on and off.

You Will Need

To experiment with using a MOSFET with an Arduino, you will need the following parts.

Quantity	Item	Appendix Code
1	Solderless breadboard	K1, T5
2	Male-to-male jumper wire or solid core wire	K1, T6
1	4 × AA battery holder	K1, H1
1	4 × AA batteries	
1	Battery clip	K1, H2
1	FQP30N06L MOSFET	K1, S6
1	6V DC motor	K1, H6

Construction

Using the breadboard layout of Figure 6-29, connect everything together. Figure 6-30 shows the complete project.

Software

The test sketch for this experiment, "ch06_", uses the Arduino IDE's Serial Monitor to send commands to the Arduino to control the speed of the motor. Figure 6-31 shows the Serial Monitor. Setting a value between 0 and 255 will set the motor to different speeds.

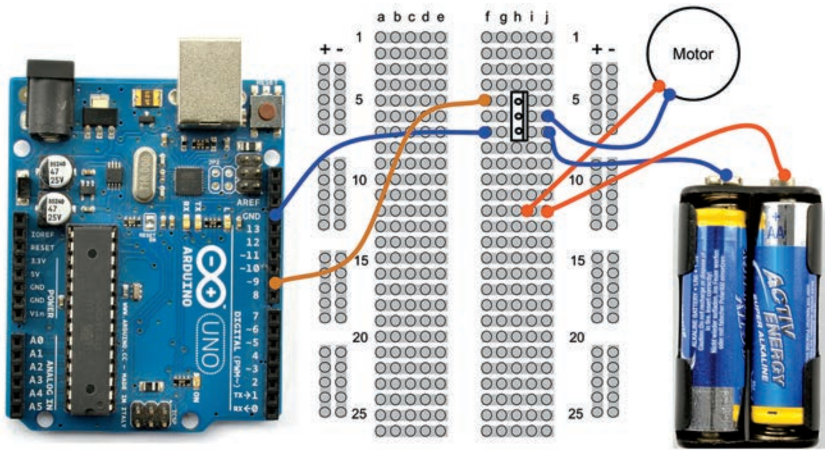


FIGURE 6-29 Breadboard layout for Arduino control of a motor

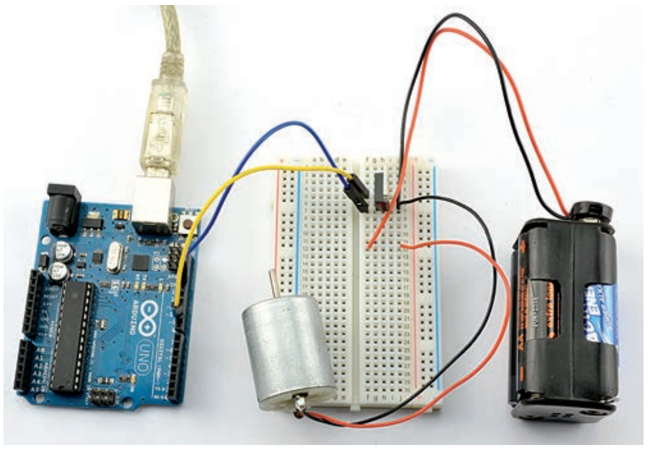


FIGURE 6-30 Arduino control of a motor

For this sketch to work correctly, you need to set the “Line ending” drop-down list on the Serial Monitor to be “No Line Ending”.

The sketch is listed below:

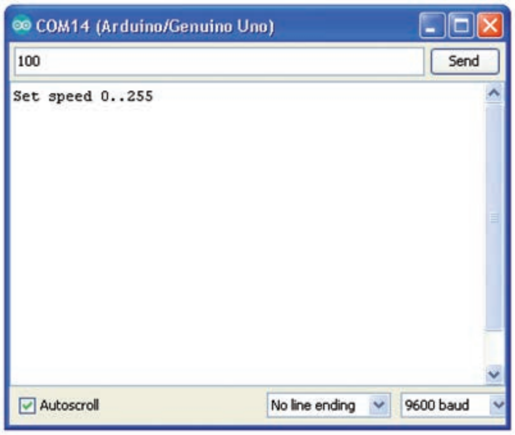
```
const int motorPin = 9;

void setup() {
  pinMode(motorPin, OUTPUT);
  Serial.begin(9600);
  Serial.println("Set speed 0..255");
}

void loop() {
  if (Serial.available()) {
    int speed = Serial.parseInt();
    analogWrite(motorPin, speed);
  }
}
```

The *setup* function sets the motor pin (pin 9) to be an output and then starts serial communication to the Serial Monitor with a welcome message that tells you the valid range of numbers for controlling the speed (0 to 255).

If a message has been sent from the Serial Monitor (indicated by *Serial.available* being true, then the speed is read and the motor pin set to the speed that was read from the Serial Monitor using *Serial.parseInt*, which converts the text of the number into an *int*.

You can use a MOSFET in this arrangement to control all sorts of devices, not just motors. However, unlike a relay that can switch either to plus volts or ground, the “source” pin (the bottom pin in Figure 6-29) of the MOSFET must be connected to ground and so the switching action must be to ground.

The arrangement of Figure 6-29 works equally well with a Raspberry Pi.



```
100
Set speed 0..255
```

No line ending
9600 baud

FIGURE 6-31 Controlling the motor speed using the Arduino Serial Monitor

Using an Alphanumeric LCD Shield with Arduino

Another commonly used Arduino shield is the LCD shield (Figure 6-32).

There are many different shields available and most use an LCD module based on the HD44780 LCD driver chip. The model used here is DFRobot LCD Shield—widely available on eBay. Most other LCD boards will work with this example code, but you may have to change the pin allocations (discussed later).

This project lets you send a short message (the display is only two lines of 16 characters) using the Serial Monitor (Figure 6-33).

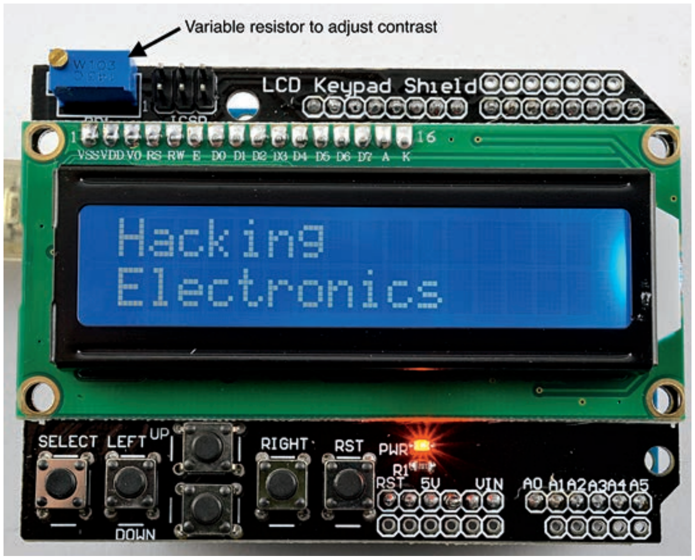
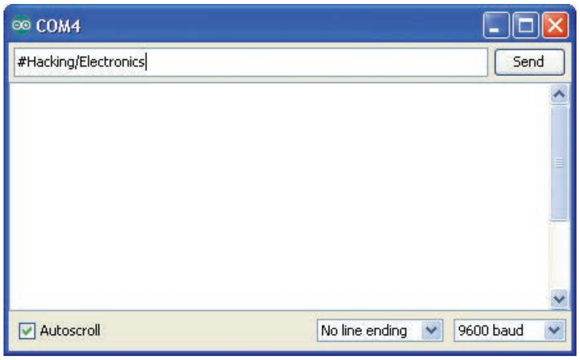


Figure 6-32 An LCD shield



#Hacking/Electronics

No line ending

9600 baud

Figure 6-33 Sending a message with the Serial Monitor

You Will Need

To experiment with an LCD display, you will need the following items.

Quantity	Item	Appendix Code
1	Arduino Uno R3	M2
1	USB Type A to Type B (as commonly used for USB printers)	
1	DFRobot LCD shield	M18

Construction

There is really not very much to construct here. Just plug the LCD shield onto the Arduino and plug in the Arduino to your computer via a USB port.

Software

The software is pretty straightforward too. Again, most of the work is done in the library.

```
#include <LiquidCrystal.h>

// LiquidCrystal display with:
// rs on pin 8
// rw on pin 11
// enable on pin 9
// d4-7 on pins 4-7
LiquidCrystal lcd(8, 11, 9, 4, 5, 6, 7);


void setup() {
  Serial.begin(9600);
  lcd.begin(2, 16);
  lcd.clear();
  lcd.setCursor(0,0);
  lcd.print("Hacking");
  lcd.setCursor(0,1);
  lcd.print("Electronics");
}
```

```
void loop() {
  if (Serial.available()) {
    char ch = Serial.read();
    if (ch == '#') {
      lcd.clear();
    }
    else if (ch == '/') {
      lcd.setCursor(0,1);
    }
    else {
      lcd.write(ch);
    }
  }
}
```

If you are using a different LCD shield, then check the specification to see which pins it uses for what. You may need to modify the line:

```
LiquidCrystal lcd(8, 11, 9, 4, 5, 6, 7);
```

The parameters to this are the pins that the shield uses for (rs, rw, e, d4, d5, d6, d7). Note that not all shields use the rw pin. If this is the case, just pick the number of a pin not being used for anything else.

The loop reads any input, and if it is a # character, it clears the display. If it is a "/" character, it moves to the second row; otherwise, it just displays the character that was sent.

For example, to send the text displayed in Figure 6-32, you would enter the following into the Serial Monitor:

```
#Hacking/Electronics
```

Notice that the LCD library provides you with the "lcd .setCursor" function to set the position for the next text to be written. The text is then written using the "lcd.write" function.

Controlling a Servo Motor with an Arduino

Servo motors are a combination of motor, gearbox, and sensor that are often found in remote-controlled vehicles to control steering or the angles of surfaces on remote-controlled airplanes and helicopters.

Unless they are special-purpose servo motors, servo motors do not rotate continuously. They usually only rotate through about 180 degrees, but can be accurately set to any position by sending a stream of pulses.

Figure 6-34 displays a servo motor and shows how the length of the pulses determines the position of the servo.

A servo will have three connections: GND, a positive power supply (5 to 6V), and a control connection. The GND connection is usually connected to a brown or black lead, the positive connection to a red lead, and the control connection to an orange or yellow lead.

The control connection draws very little current. The server expects to receive a pulse every 20 ms or so. If the pulse is 1.5 ms in duration, then the servo will sit at its middle position. If the pulse is shorter, it will settle in a position to one side, and if the pulse is longer, it will move to a position on the other side of the center position.

You Will Need

To experiment with a servo and Arduino, you will need the following items.

Quantity	Item	Appendix Code
1	Arduino Uno R3	M2
1	9g servo motor	K1, H10
3	Male-to-male jumper wire or solid core wire	K1, T6

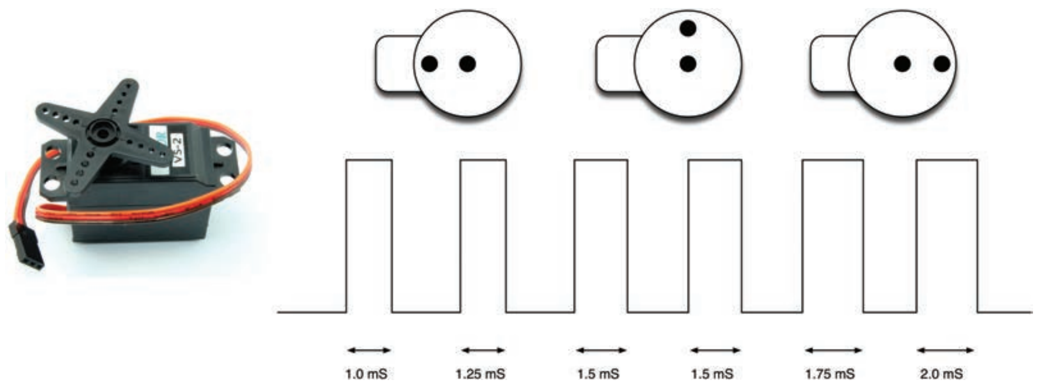


FIGURE 6-34 Controlling a servo motor with pulses

Construction

Figure 6-35 shows a servo connected to an Arduino using male-to-male jumper wires to the servo motor's connector.

Before you power the servo motor from the 5V supply of an Arduino, first check that the Arduino can supply the current requirement. Most small servos will be just fine, such as the tiny 9g servo shown in Figure 6-35.

In Figure 6-35, you can also see the little blue trimpot used to set the position of the servo. This is connected to A1, but uses A0 and A2 to provide GND and +5V to the track ends of the variable resistor.

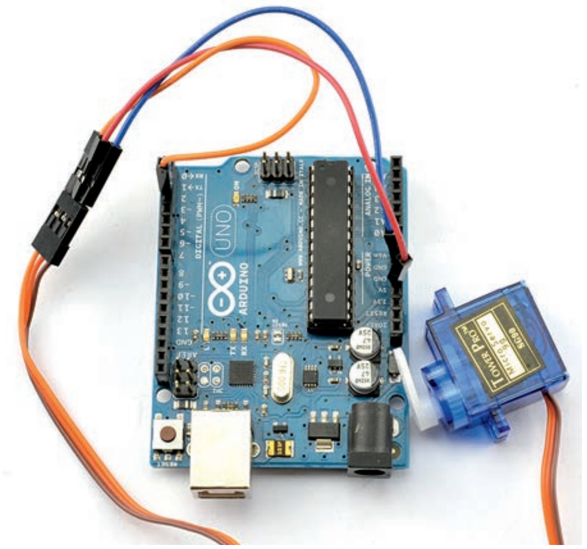


FIGURE 6-35 Connecting a servo to an Arduino

Software

The Arduino has a library specifically designed for generating the pulses that the servo needs. The following example sketch (called "ch06_servo") will use this library to set the position of the servo arm to an angle between 0 and 180 degrees sent to the Arduino over the Serial Monitor.

The code for "ch06_servo" is listed below:

```
#include <Servo.h>

const int servoControlPin = 2;

Servo servo;

void setup() {
  servo.attach(servoControlPin);
  Serial.begin(9600);
  Serial.println("Angle in degrees (0 to 180)");
}

void loop() {
  if (Serial.available()) {
    int angle = Serial.parseInt();
    servo.write(angle);
  }
}
```

The “setup” function sets up the pin to be used with the servo motor and then writes a message to the Serial Monitor explaining that an angle between 0 and 180 degreees needs to be sent to set the servo motor's position.

The “loop” function works much like the motor control sketch earlier in this chapter. It checks for messages coming from the Serial Monitor, and uses the value received to set the angle of the servo.

Typing Passwords Automatically

The Arduino Leonardo can be used to impersonate a USB keyboard. Unfortunately, this is not true of the Arduino Uno, so in this section you will need an Arduino Leonardo.

Figure 6-36 shows the device we are going to construct.

All that happens when you press the button is that the Arduino Leonardo pretends to be a keyboard and types the password set in the sketch, wherever the cursor happens to be.

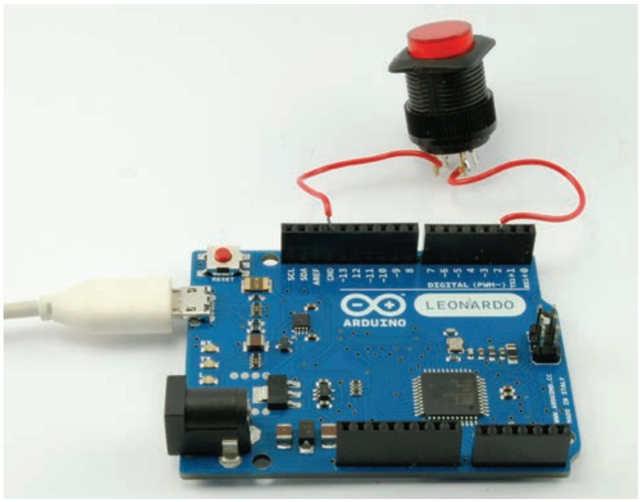


FIGURE 6-36 Entering passwords automatically with Arduino Leonardo

You Will Need

To build this, you will need the following items.

Quantity	Item	Appendix Code
1	Arduino Leonardo	M21
1	Micro USB lead for the Leonardo	
1	Impressive switch	H15
	Hookup wire	K1, T7

Construction

Solder leads to the switch, and tin the ends so they can be pushed directly into the sockets on the Arduino. One lead from the switch should go to digital pin 2 and the other to GND.

Program the Arduino Leonardo with the sketch "password". Note that when programming the Leonardo, you may have to hold down the reset button until the message "uploading…" appears in the Arduino software.

Software

To use the project, just position your mouse over a password field and press the button. Please note that this project is really just to illustrate what you can do with an Arduino Leonardo. To find your password, all someone would have to do is press the button while in a word processor. So, in terms of security, it is about as secure as writing your password on a sticky note and attaching it to your computer monitor!

The sketch, "ch06_password", is very simple. The first step is to define a variable to contain your password. You will need to change this to your password. We then define the pin to use for the switch.

```
// Arduino Leonardo Only

char* password = "mysecretpassword";

const int buttonPin = 2;
```

The Leonardo has access to special keyboard and mouse features not available to other types of Arduino. So, in the "setup" function, the Keyboard feature is started with the line "Keyboard.begin()".

```
void setup() {
  pinMode(buttonPin, INPUT_PULLUP);
  Keyboard.begin();
}
```

In the main loop, the button is checked with a digital read. If the button is pressed, then the Leonardo uses "Keyboard.print" to send the password. It then waits two seconds to prevent the password being sent multiple times.

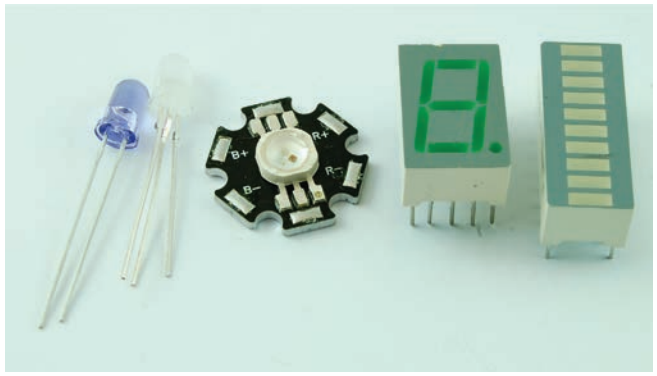
```
void loop() {
  if (! digitalRead(buttonPin)) {
    Keyboard.print(password);
    delay(2000);
  }
}
```

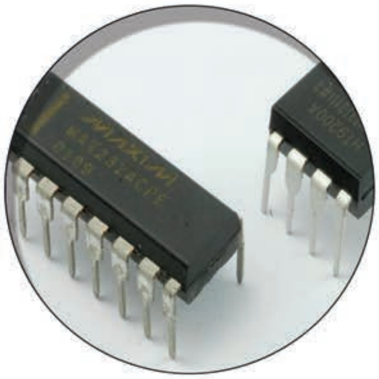
Summary

This chapter should have got you started using the Arduino and given you some food for thought for clever hacks using it. It has, however, only scratched the surface of what is possible with this versatile board.

For more information on programming the Arduino, you may wish to look at some of the author's other books on this topic. *Programming Arduino: Getting Started with Sketches* assumes no prior programming experience and will show you how to program the Arduino from first principals. *30 Arduino Projects for the Evil Genius* is a project-based book that explains both the hardware and programming side of Arduino, and is illustrated with example projects, nearly all of which are built on breadboard.

The official Arduino web site, www.arduino.cc, has a wealth of information on using the Arduino, as well as the official documentation for the Arduino commands and libraries.





7

Hacking with Raspberry Pi

Whereas the Arduino Uno is based around a 16MHz 8-bit microcontroller that just runs the program currently uploaded to its flash memory, the Raspberry Pi 3 (Figure 7-1) is a 900MHz single board computer running its own version of the Linux operating system.

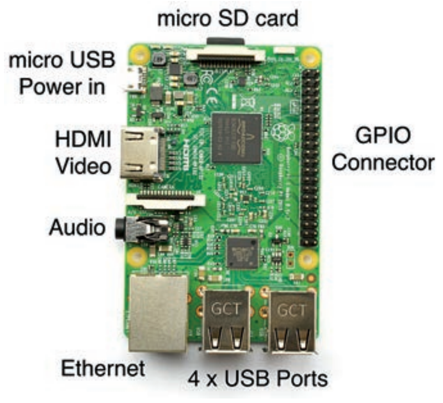
To use an Arduino, you have to have a second computer from which to program the Arduino. However, you can attach a keyboard, mouse and monitor to a Raspberry Pi and use it as a regular computer if you like. What sets a Raspberry Pi apart from a PC is that it:

- Is small—tiny in fact
- Low cost—around $40
- Low power—uses around 2W
- Has GPIO (general purpose IO pins)—like the Arduino's pins, these are used to connect external electronics

The decision as to whether to use a Raspberry Pi or an Arduino as the basis for a project can be difficult. The Arduino is a simpler device that you don't have to wait to boot-up, so most simple control applications are best made with an Arduino, unless you need some feature of the Raspberry Pi such as:

- Internet connection
- Video output
- High performance processor (for things like image processing)
- Interfaces to USB peripherals such as barcode scanners, printers, etc.

FIGURE 7-1 A Raspberry Pi 3 Model B Single Board Computer



Setting Up a Raspberry Pi

You can use a Raspberry Pi without a keyboard, mouse and monitor, but to be able to do that you first need them to set up the Raspberry Pi so that it can be accessed from a second computer over Wifi.

You do not have to use a computer monitor, the Raspberry Pi will also connect to any TV that has an HDMI lead.

Preparation

So, start by making sure that you have the following items, along with your Raspberry Pi:

- USB keyboard and mouse
- Monitor or TV with HDMI socket and HDMI lead
- A micro SD card at least 8GB and ideally Class 10 (high speed)
- A USB WiFi adaptor unless you have a Raspberry Pi 3 that includes WiFi
- A second computer with Windows, Mac or Linux

Before you can boot up your Raspberry Pi, you must prepare the micro SD card by formatting it as FAT and then copying the NOOBS (New Out Of Box Software). You will find full instructions on doing this at https://www.raspberrypi.org/downloads/noobs/.

You can then put the micro SD card into the slot on the Raspberry Pi, connect up all the peripherals and boot it up.

The first time it boots up, it will boot into NOOBS, which will lead you through the process of installing the Raspbian operating system. Once this is done, the Raspberry Pi will reboot and you will see the Raspberry Pi Desktop (Figure 7-2).

Click on the WiFi icon at the top of the screen (Figure 7-3), select your WiFi network from the list, and connect to it by entering your password.

Once connected to WiFi, if you plan to connect to your Raspberry Pi remotely from a second computer, you will need to find its internal IP address. To do this, start a terminal session by clicking on the black terminal icon at the top of the screen and then type the command **hostname –I** (Figure 7-4). You can

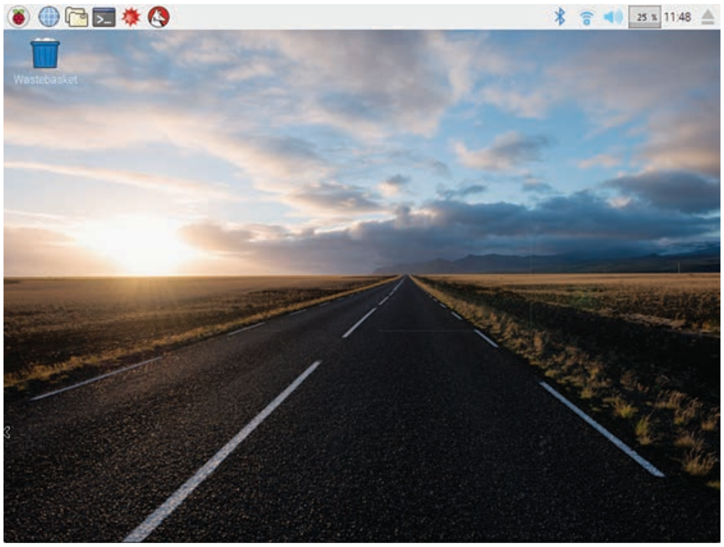
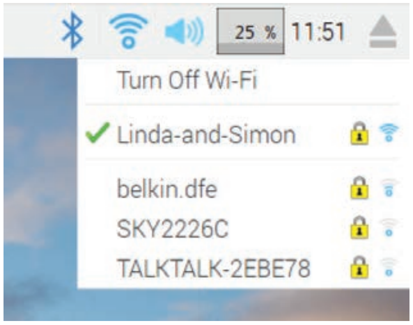


FIGURE 7-2 The Raspberry Pi desktop



25 %
11:51

Turn Off Wi-Fi

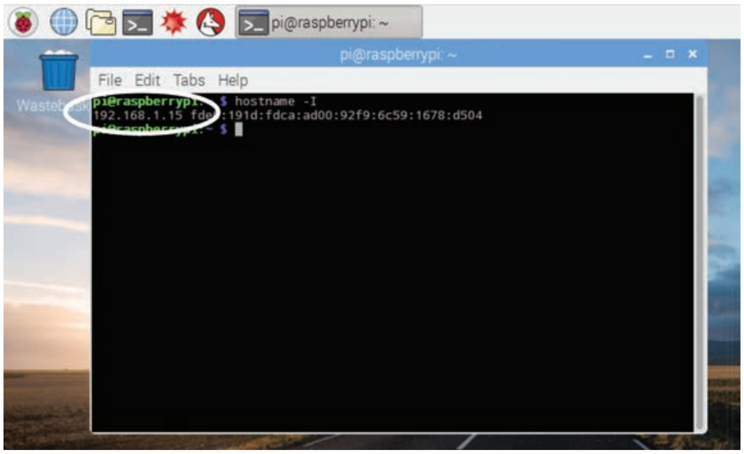
✓ Linda-and-Simon

belkin.dfe

SKY2226C

TALKTALK-2EBE78

FIGURE 7-3 Connecting to WiFi



```
pi@raspberrypi:~ $ hostname -I
192.168.1.15 fde :191d:fdca:ad00:92f9:6c59:1678:d504
pi@raspberrypi:~ $ 
```

FIGURE 7-4 Finding the IP address of your Raspberry Pi

see that in this case, the local IP address of the Raspberry Pi is 192.168.1.15. Make a note of this address.

Fetching the Example Code

While you are using the command line, you may as well fetch the example code used by the book onto your Raspberry Pi. The easiest way to do this is actually to fetch all the downloads for the book by running the command:

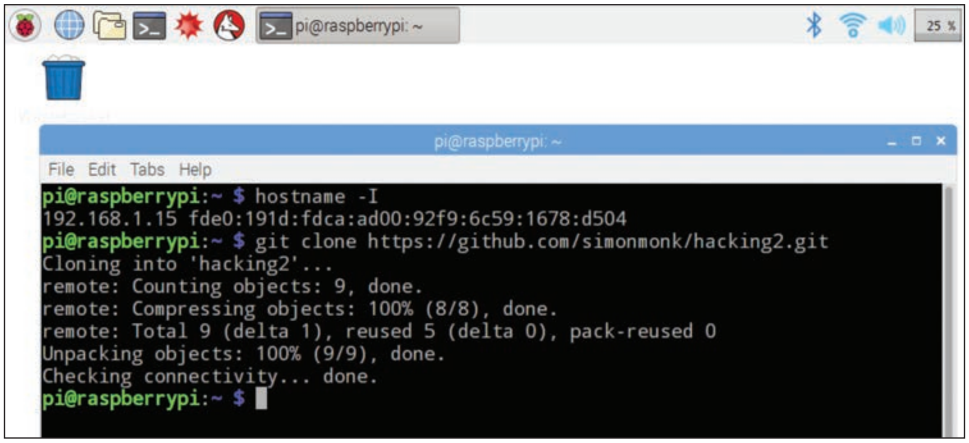
```
git clone https://github.com/simonmonk/
hacking2.git
```

You can see the result of running this command in Figure 7-5.

Connecting to Your Pi from a Second Computer

It may be that the Raspberry Pi setup described above is all you need, and if you are experimenting with electronics connected to the Pi, having a keyboard, mouse and monitor attached is not a problem, however, if you plan to make projects that are mobile (perhaps a robot), or in a location where you can't easily get to the Pi, then it's a good idea to be able to connect to the Raspberry Pi remotely from another computer.

To do this, you first need to configure Raspbian to allow access in this way. So from the desktop of your Raspberry



```
pi@raspberrypi:~ $ hostname -I
192.168.1.15 fde0:191d:fdca:ad00:92f9:6c59:1678:d504
pi@raspberrypi:~ $ git clone https://github.com/simonmonk/hacking2.git
Cloning into 'hacking2'...
remote: Counting objects: 9, done.
remote: Compressing objects: 100% (8/8), done.
remote: Total 9 (delta 1), reused 5 (delta 0), pack-reused 0
Unpacking objects: 100% (9/9), done.
Checking connectivity... done.
pi@raspberrypi:~ $
```

FIGURE 7-5 Installing the book's example code

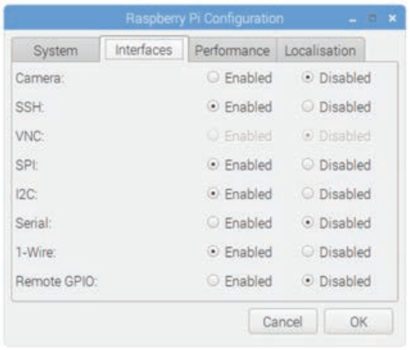


FIGURE 7-6 Turning on SSH

Raspberry Pi Configuration

System | Interfaces | Performance | Localisation

Camera:	Enabled	[✓ Disabled]
SSH:	[✓ Enabled]	Disabled
VNC:	Enabled	[✓ Disabled]
SPI:	[✓ Enabled]	Disabled
I2C:	[✓ Enabled]	Disabled
Serial:	Enabled	[✓ Disabled]
1-Wire:	[✓ Enabled]	Disabled
Remote GPIO:	Enabled	[✓ Disabled]

Pi, click on the Raspberry Menu and select *Raspberry Pi Configuration* from the *Preferences* menu. Now click on the *Interfaces* tab and check the box next to SSH as shown in Figure 7-6.

While you are here, it's probably worth turning on some of the other interfaces that you will need later in the book, so also check the boxes for SPI, I2C and 1-Wire.

If your other computer is a Mac or Linux computer, then it already has the software that you need to connect remotely to a Raspberry Pi. If you are a Windows user, then you will need to download the PuTTY software at http://www.putty.org/.

If you are using a Windows computer, run the PuTTY program and enter your Raspberry Pi's IP address in place of 192.168.1.15 in Figure 7-7. Then click Open. The first time you do this, you
will receive a security message about certificates that you should accept.

What is going on here is that you are connecting with SSH (Secure Socket Shell), which allows you to run commands on

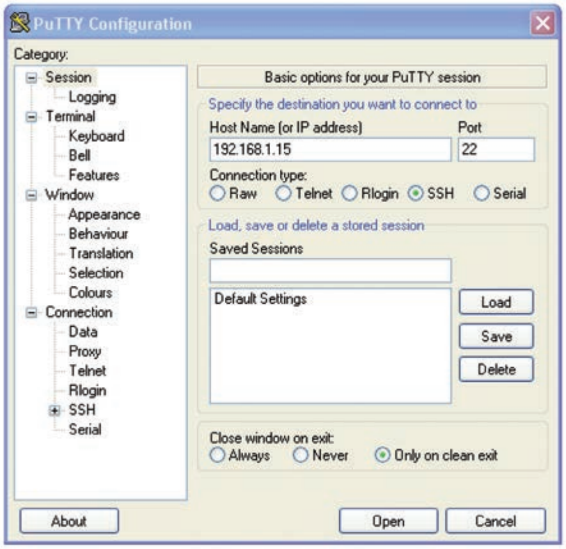


FIGURE 7-7 Connecting to your Raspberry Pi with PuTTY

PuTTY Configuration

Basic options for your PuTTY session

Specify the destination you want to connect to

Host Name (or IP address): 192.168.1.15
Port: 22

Connection type: Raw Telnet Rlogin [✓ SSH] Serial

Close window on exit: Always Never [✓ Only on clean exit] your Raspberry Pi remotely, but as if you were using a terminal session directly from the Raspberry Pi.

If you are using a Mac or Linux computer, start a terminal session and enter the following command, replacing 192.168.1.15 with the IP of your Raspberry Pi that you found earlier.

```
$ ssh pi@192.168.1.15
```

Whether using Putty or a Mac or Linux terminal, you now need to log in. The username is "pi" and the password is "raspberry".

Congratulations, you can now connect to your Raspberry Pi from another computer on your home network and you can if you want disconnect the keyboard mouse and monitor from the Raspberry Pi.

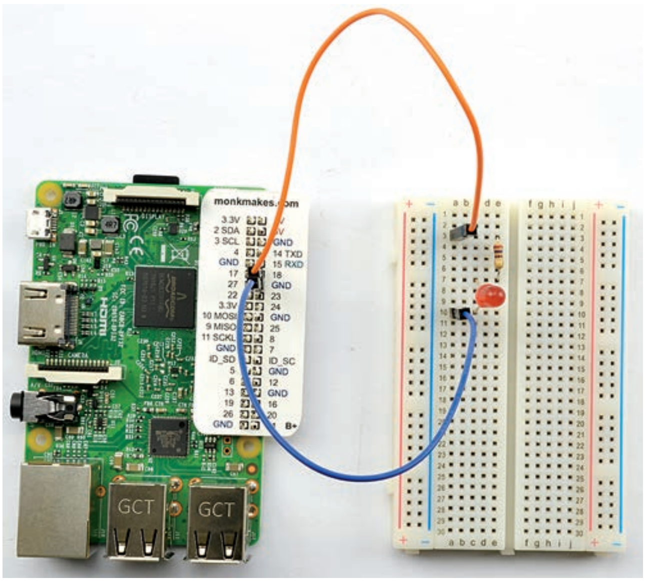
Blinking an LED

To get the hang of running a program on the Raspberry Pi, we can start in the same way as you did on an Arduino by blinking an LED.

Since there is no built-in LED that is controllable from one of the normal GPIO (general purpose input/output) pins on a Raspberry Pi, you need to attach an external LED and resistor using breadboard as shown in Figure 7-8.

The LED is going to be controlled by pin GPIO 18.

FIGURE 7-8 Attaching an LED and resistor to a Raspberry Pi



Python

Whereas Arduino is programmed in C, the most common language used to program a Raspberry Pi is Python.

Many of the features of Python are similar to those of C, but unlike C, Python uses indentation to mark off blocks of code. So in Arduino C, you might write

```
if (x > 10) {
  x = 0;
}
```

The equivalent in Python would be

```
if x > 10 :
  x = 0
```

The differences in the formatting are that in Python:

- You do not need parentheses around the condition "x > 10".
- The start of a block of code is indicated by ":" rather than "{".
- All lines of code inside the block must be indented to the same level. This is done by convention in Arduino C but is mandatory in Python.

There are lots of other differences between Python and Arduino C, and if you would like to find out more about learning Python, you might enjoy my book *Programming the Raspberry Pi: Getting Started with Python* from TAB DIY.

You Will Need

To connect your LED up, you will need the following items.

Quantity	Item	Appendix Code
1	Raspberry Pi Model B (Pi 2 or later preferred)	M11
1	Solderless breadboard	K1, T5
1	270Ω/470Ω resistor	K1
1	Red LED	K1
2	Male to female jumper wires	K1, T12

Raspberry Pi GPIO Connections

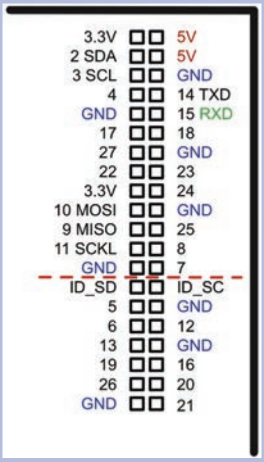
Figure 7-9 shows the GPIO pin out of a Raspberry Pi 3.

The dotted line separates the top 26 pins of the connector from the rest because in version 1 of the Raspberry Pi model B, there were only the top 26 pins rather than the full 40 pins of the newer models.

Some of the pins on the connector are to supply power to other devices attached to the connector. This includes 5V, 3.3V and GND connections. Most of the rest of the pins can be used as either digital inputs, digital outputs or analog outputs (PWM) but unlike the Arduino, the Raspberry Pi does not have any analog inputs.

Some of the GPIO pins also have a second function relating to a serial interface:

Figure 7-9 The Raspberry Pi GPIO Connector



- GPIO 2 and 3 are used as SDA and SCL if the I2C interface is enabled to allow I2C displays and sensors to be connected to the Raspberry Pi.
- GPIO 9, 10 and 11 are used in the SPI (Serial Programming Interface).
- GPIO 14 and 15 will be used by the TTL Serial Interface if that is enabled.

By default, none of these extra interfaces are enabled when you install Raspbian, so all the GPIO pins can be used as inputs or outputs unless you enable the interface (I2C, SPI or Serial) as shown in Figure 7-6. In general, if you just need a digital output, then avoid these special purpose pins.

Digital outputs on a Raspberry Pi are not capable of providing to 40mA of an Arduino. The maximum that you should use for one pin is 16mA. The inputs and outputs of the Raspberry Pi operate at 3.3V rather than the 5V of an Arduino Uno. Connecting 5V to a Raspberry Pi GPIO is likely to damage your Raspberry Pi. You must always use 3.3V logic with a Raspberry Pi.

Using a 270Ω resistor will make the LED brighter, but a 470Ω resistor will also work fine. These items (apart from the Raspberry Pi) are all included in the Electronics Starter Kit for Raspberry Pi by MonkMakes (http://monkmakes.com/rpi_esk) and the Hacking Electronics Mega Kit (http://monkmakes.com/hacking2).

Software

With the LED connected, run the example Python program "ch07_blink.py" by issuing the following commands:

```
$ cd /home/pi/hacking2/pi
$ python ch07_blink.py
```

The first line makes sure that you are in the right directory to run the program and you only need to type this the first time that you want to run the program. The second line runs the Python program. Note that if you are using an older version of Raspbian then you may have to run the command with "sudo" in front like this:

```
$ sudo python ch07_blink.py
```

If all is well, the LED will start to slowly blink. When you have had enough, hold down the CTRL key and type **c** to quit the program.

The code for ch07_blink.py is listed below:

```
import RPi.GPIO as GPIO

import time

# Configure the Pi to use the BCM (Broadcom) pin names
GPIO.setmode(GPIO.BCM)

led_pin = 18
GPIO.setup(led_pin, GPIO.OUT)

try:
    while True:
        GPIO.output(led_pin, True)  # LED on
        time.sleep(0.5)             # delay 0.5 seconds
        GPIO.output(led_pin, False) # LED off
        time.sleep(0.5)             # delay 0.5 seconds

finally:
    print("Cleaning up")
    GPIO.cleanup()
```

The code has some similarities with its Arduino equivalent. Python uses the # symbol to denote comments that are not part of the code.

First, the *RPi.GPIO* Python library is imported. This is the library that allows Python programs to control the GPIO pins. The second import is for the *time* module that is used to produce the delay between turning the LEDs on and off.

The code *GPIO.setmode(GPIO.BCM)* is needed in all your Python programs and specifies that the standard numbering system for GPIO pins should be used rather than the alternative numbering system that relies on pin positions.

The variable *led_pin* is specified as the pin that is to be connected to the LED and in this case it is pin 18. The line after that sets *led_pin* to be an output.

The *try* and *finally* commands ensure that when you press CTRL-C to quit the program the code after *finally* is run that sets all the GPIO pins back to a safe state.

The code indented in from the *try* block, first sets the GPIO pin connected to the LED high, delays for half a second, sets it low, delays for another half second and so on indefinitely because of the *while* loop enclosing it.

Controlling a Relay with Raspberry Pi

Now that you can turn a GPIO pin on and off, we could replace the LED and resistor with a relay module, opening up more possibilities such as switching a toy on and off as described in Chapter 6 in the section "Hacking a Toy for Arduino Control."

Figure 7-10 shows a relay module attached to the Raspberry Pi using female to female jumper wires.

The connections are as follows:

- GND on the relay module to GND on the Raspberry Pi
- VCC on the relay module to 5V on the Raspberry Pi
- IN on the relay module to GPIO18 on the Raspberry Pi

As with the relay module experiments with Arduino in Chapter 6, you will need to run slightly different programs depending on whether your relay module has "inverted" inputs. So run either "ch7_relay_click.py" or "ch7_relay_click_ inverted.py". Note that if you run the wrong program, it just won't work, you will not break anything.

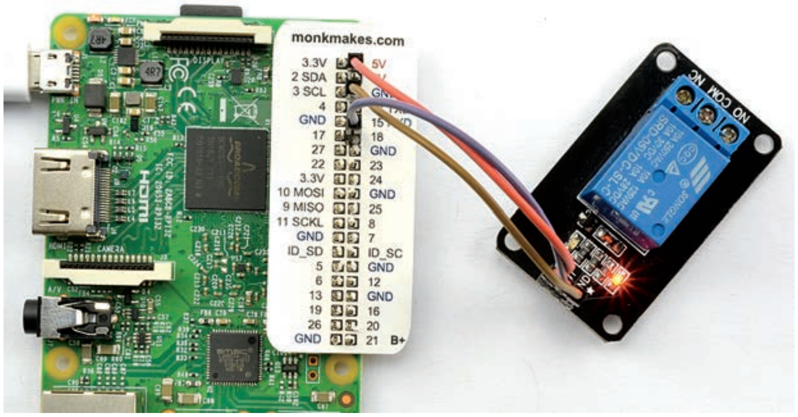


FIGURE 7-10 Controlling a relay module from a Raspberry Pi

If all is well, you will hear the relay module clicking on and off.

Controlling a Relay from a Web Page

The Raspberry Pi 3 has built-in WiFi hardware making it ideal for network projects such as the web-controlled relay that we made with a NodeMCU "Arduino" board in Chapter 6. We can replicate this, using a web framework called Bottle running on the Raspberry Pi.

The first step in doing this is to install the Bottle web framework using the commands:

```
$ sudo apt-get update
$ sudo apt-get install python-bottle
```

With the relay module still connected just as it was in the previous section, you can test out the web relay by first changing directory using:

```
$ cd /home/pi/hacking2/ch07_web_relay
```

and then either running web_relay.py or web_relay_inverted.py depending on your relay module. In this case the command to run the program has to be prefixed by "sudo" as super-user privileges are required for the Raspberry Pi to act as a web server. So, if you have a normal (non-inverted input) relay run:

```
$ sudo python web_relay.py
```

You should see a message something like this, if the web server is up and running:

```
$ sudo python web_relay.py
Bottle v0.12.7 server starting up
    (using WSGIRefServer())...
Listening on http://0.0.0.0:80/
Hit Ctrl-C to quit.
```

Remember how you found the IP address of your Raspberry Pi, back in the section “Connecting to your Pi from a Second Computer”? Well, if not go back and find the IP address, because you now need to enter it into the address bar of a browser running on a second computer on your network (Figure 7-11).

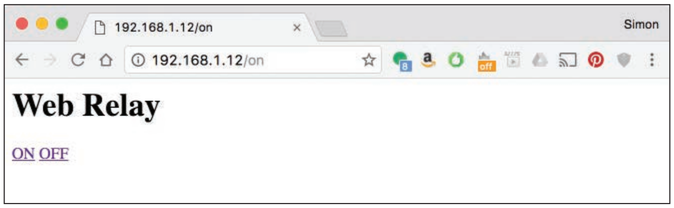


FIGURE 7-11 A web interface to control a relay on Raspberry Pi

Software

The code that the Bottle web framework uses to serve up a web interface uses two files. The first file (home.tpl) contains an HTML template. This is the HTML that when displayed in a browser will provide us with a title of “Web Relay” and a couple of hyperlinks as shown in Figure 7-11.

```
<html>
<body>

<h1>Web Relay</h1>
<a href="/on">ON</a>
<a href="/off">OFF</a>

</body>
</html>
```

The more interesting stuff lives in the Python program that runs the web server using Bottle (web_relay.py or web_relay_inverted.py). The code for web_relay.py is listed below.

```
from bottle import route, run, template,
   request
```

```
import RPi.GPIO as GPIO

GPIO.setmode(GPIO.BCM)
CONTROL_PIN = 18
GPIO.setup(CONTROL_PIN, GPIO.OUT)

@route('/')
def index():
    return template('home.tpl')

@route('/on')
def index():
    GPIO.output(CONTROL_PIN, True)
    return template('home.tpl')

@route('/off')
def index():
    GPIO.output(CONTROL_PIN, False)
    return template('home.tpl')


try:
    run(host='0.0.0.0', port=80)
finally:
    print('Cleaning up GPIO')
    GPIO.cleanup()
```

After the initial imports of the *Bottle* and *RPi.GPIO* libraries, the control pin for the relay is defined and set to be an output.

There then follow three functions, each marked with a line preceding it that starts *@route*. Each of these functions will be called when the web server receives a request for a particular resource. So, "/", the root page will just return the contents of the template "home.tpl".

If the request is for "/on" or "/off" then the template is still returned, but first the relay is switched on or off as appropriate.

The final section of the code is a *try/finally* block that starts the web server running and also clears up the GPIO pins when CTRL-C is pressed.

Summary

The Raspberry Pi is a fun little computer to have around, whether you are attaching electronics to it or just converting it into a media center or retro-games console.

8

Hacking with Modules

There are many modules available that provide a great shortcut when hacking together a project. These modules are usually a tiny PCB with a few components on them and some convenient connection points. They make it very easy to use some surface-mounted ICs that would otherwise be very difficult to solder connections to. Many of these modules are designed to be used with Arduinos or Raspberry Pis and are available at very low cost on eBay.

In this chapter, you will explore some of the more fun and useful modules available from suppliers like SparkFun and Adafruit, most of whose modules are also open-source hardware. So you'll get to see the schematics for them and even make your own modules using the design if you wish.

Access to the schematics and data sheets is very useful when trying to use a module. There are a few important things you need to know about any module before you use it:

- What is the range of supply voltage?
- How much current does it consume?
- How much current can any outputs supply?

Detecting Movement

PIR motion sensors are used in intruder alarms and for automatic security alarms. They detect movement using infrared light. They are also cheap and easy to use.

In this example, you will first experiment with a PIR module, using it to light an LED, and then look at how it could be hooked up to an Arduino to send a warning message to the Serial Console.

You Will Need (PIR and LED)

Quantity	Names	Item	Appendix Code
1		PIR module	M5
1	D1	LED	K1
1	R1	470Ω resistor	K1
1		Solderless breadboard	K1, T5
		Male-to-female jumper wires	K1, T12
1		4 × AA battery holder	K1, H1
1		4 × AA batteries	
1		Battery clip	K1, H2

Breadboard

Figure 8-1 shows the schematic diagram for this experiment.

Looking at the datasheet for this particular module, the supply voltage range is 5V to 7V, so it will work just fine with our four AA batteries.

The module is very easy to use. You just supply it with power and its output goes high (to supply voltage) when movement is detected and then back low again after a second or two.

The datasheet also says that the output can supply up to 10mA. That isn't a great deal, but is enough to light an LED. By choosing a 470Ω resistor, we will be limiting the current to:

$$I = V / R = (6V - 2V) / 470\Omega = 4 / 470 = 8.5mA$$

Figure 8-2 shows the breadboard layout, while Figure 8-3 offers a photograph of the actual breadboard.

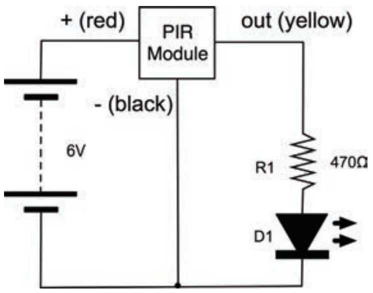


FIGURE 8-1 Schematic diagram—using a PIR module with an LED

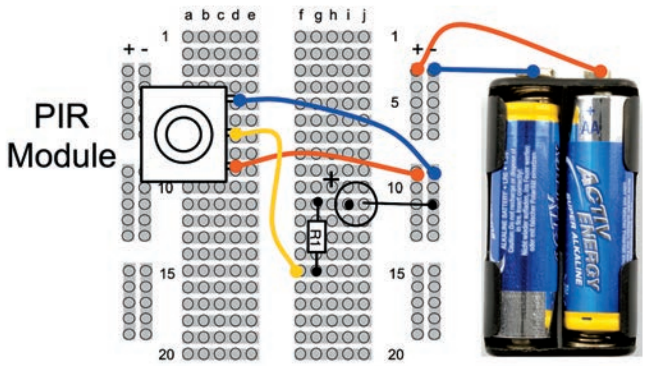


FIGURE 8-2 Breadboard layout—using a PIR module with an LED

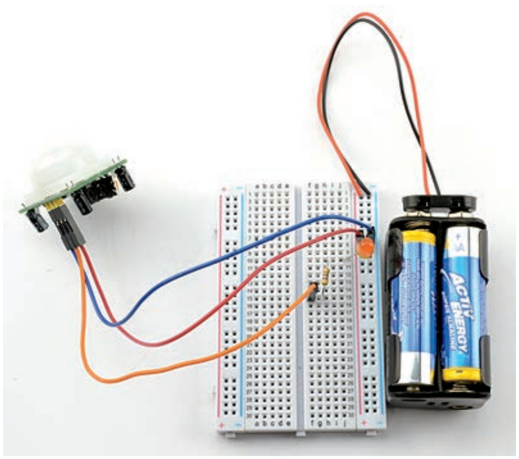


FIGURE 8-3 Using a PIR module with an LED

The PIR module has three pins labeled +5V, GND, and OUT. The supplied connector lead has red, black, and yellow leads. Hook it up so the red lead connects to the connection labeled +5V.

When it's powered up, the LED will light every time movement is detected.

Having already discussed the PIR sensor so we know what to expect of it, it's time to interface it with an Arduino.

You Will Need (PIR and Arduino)

To interface the PIR sensor with an Arduino, you really only need the PIR sensor and an Arduino. However, if your PIR module was not supplied with leads, you will also need three male-to-female jumper wires (Appendix codes K1, T12).

Quantity	Item	Appendix Code
1	PIR module	M5
1	Arduino Uno R3	M2
3	Male-to-female jumper wires	

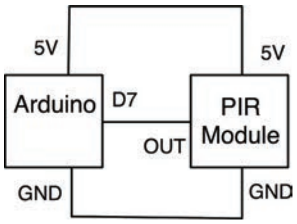


FIGURE 8-4 Schematic diagram for the Arduino and PIR sensor

Construction

Figure 8-4 shows the schematic diagram for this, while Figure 8-5 shows how the PIR module is wired to the module.

Before you move onto the next stage of programming the Arduino, temporarily remove the OUT of the PIR module from its Arduino socket. The reason for this is that you do not know what sketch was last running on the Arduino. It might have been something where pin 7 was an output, and if it was, this could easily damage the output electronics of the PIR sensor.

Up to now, you have mostly been using an Arduino's pin operating as a digital output. In this project, the pin will be acting as a digital input. If the voltage at the pin is between 0V and about 2.5V, the Arduino will read the input as LOW (False);

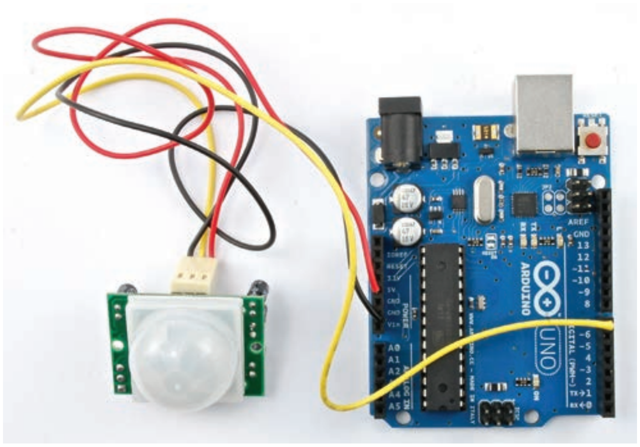


Figure 8-5 The Arduino and the PIR sensor

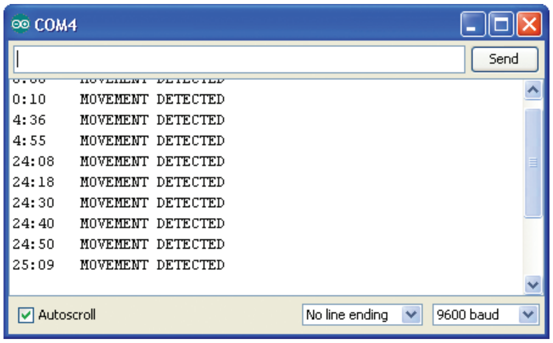
if the voltage at the pin is above 2.5V, it will count as being HIGH (True). This allows you to connect switches, and other devices like the PIR module that have a digital output, to an Arduino.

Software

Load the sketch "ch08_pir_warning" into the Arduino IDE and onto the Arduino board, and then plug the yellow "OUT" lead back into pin 7 on the Arduino.

When you launch the Serial Monitor (Figure 8-6), you will see an event appear every time movement is detected. Imagine leaving this running while away from your computer—to detect snoopers!

The sketch is very straightforward.



```
0:10    MOVEMENT DETECTED
4:36    MOVEMENT DETECTED
4:55    MOVEMENT DETECTED
24:08   MOVEMENT DETECTED
24:18   MOVEMENT DETECTED
24:30   MOVEMENT DETECTED
24:40   MOVEMENT DETECTED
24:50   MOVEMENT DETECTED
25:09   MOVEMENT DETECTED
```

Figure 8-6 The Serial Monitor showing intruder alerts

```
const int pirPin = 7;

void setup() {
  pinMode(pirPin, INPUT);
  Serial.begin(9600);
}
void loop() {
  if (digitalRead(pirPin)) {
    int totalSeconds = millis() / 1000;
    int seconds = totalSeconds % 60;
    int mins = totalSeconds / 60;
```

```
      Serial.print(mins);
      Serial.print(":");
      if (seconds < 10) Serial.print("0");
      Serial.print(seconds);
      Serial.println("\tMOVEMENT DETECTED");
      delay(10000);
    }
}
```

The only part of the code that is a bit different than the other sketches we have seen deals with displaying an elapsed time in minutes and seconds next to each event.

This code uses the Arduino "millis" function, which returns the number of milliseconds since the Arduino was last reset. This is then separated into its minute and second components and the various parts printed out as a message. The last part to be displayed uses the "println" command that adds a line feed to the end of the text so the next text starts on a new line.

The special character "\t" in this "println" is a tab character, to line the output up neatly.

PIR and Raspberry Pi

Check with a multimeter first, but most low-cost PIR sensors like this have a 3.3V output, which means that they can be connected directly to a Raspberry Pi's digital input as shown in Figure 8-7.

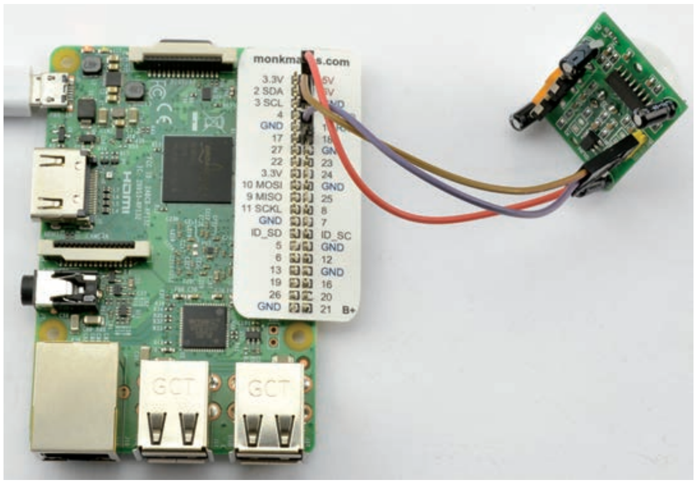


FIGURE 8-7 A PIR sensor connected to a Raspberry Pi

To use with a Raspberry Pi, you will need female-to-female jumper wires connected as follows:

- GND on the PIR module to GND on the Raspberry Pi
- VCC on the PIR module to 5V on the Raspberry Pi
- OUT on the PIR sensor to GPIO18 on the Raspberry Pi

The Python program of "ch08_pir_warning.py" will display a message each time movement is detected in a similar way to its Arduino counterpart.

```
import RPi.GPIO as GPIO
import datetime, time

# Configure the Pi to use the BCM (Broadcom)
# pin names, rather than the pin positions
GPIO.setmode(GPIO.BCM)
pir_pin = 18
GPIO.setup(pir_pin, GPIO.IN)

try:
    while True:
        if GPIO.input(pir_pin):
            print("Movement Detected " +
                  str(datetime.datetime.now()))
            time.sleep(1)

finally:
    print("Cleaning up")
    GPIO.cleanup()
```

Pin 18 is set to be an input (GPIO.IN) and the GPIO.input function used to detect when movement has occurred.

Using Ultrasonic Rangefinder Modules

Ultrasonic rangefinders use ultrasound (higher frequency than the human ear can hear) to measure the distance to a sound-reflective object. They measure the time it takes for a pulse of sound to travel to the object and back. Figure 8-8 shows a low-

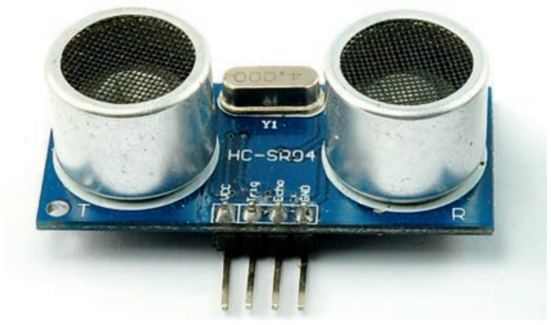


Figure 8-8 Ultrasonic rangefinders

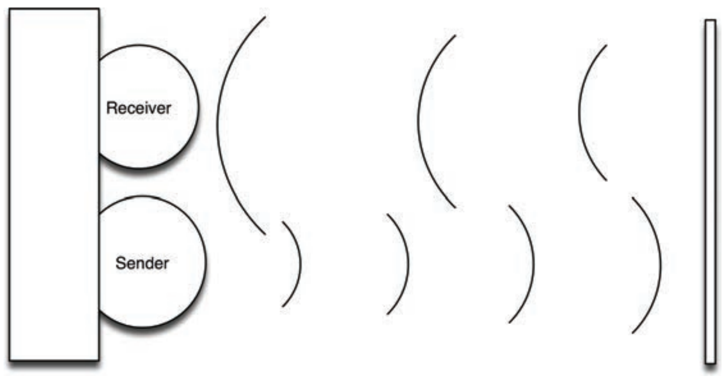


Figure 8-9 Ultrasonic rangefinding

cost sonar module (less than USD 5) with separate ultrasonic transducers for sending the pulse and receiving the echo.

These rangefinders can be used with both Arduino and Raspberry Pi.

Ultrasonic rangefinding works the same as sonar used by ships and submarines. A sound wave is sent out from a sender, hits an object, and bounces back. Since we know the speed of sound, the distance to the sound-reflecting object can be calculated from the time it takes for the sound to come back to the receiver (Figure 8-9).

The sound used is at a high frequency—hence, it is called ultrasonic. Most units operate at a frequency of about 40 kHz. Not many people can hear sounds above 20 kHz.

You Will Need

To try out a rangefinder with Arduino, you will need the following items.

Quantity	Item	Appendix Code
1	Arduino Uno R3	M2
1	HC-SR04 rangefinder	M7

The HC-SR04 Rangefinder

These modules can just fit into the side connector of an Arduino if you can spare the pins to supply them with current using two output pins (Figure 8-10).

Load the sketch "range_finder_budget" onto the Arduino and then plug the rangefinder module into the Arduino, as shown in Figure 8-10. The four HC-SR04 pins fit into D8 to D11 on the Arduino Uno.

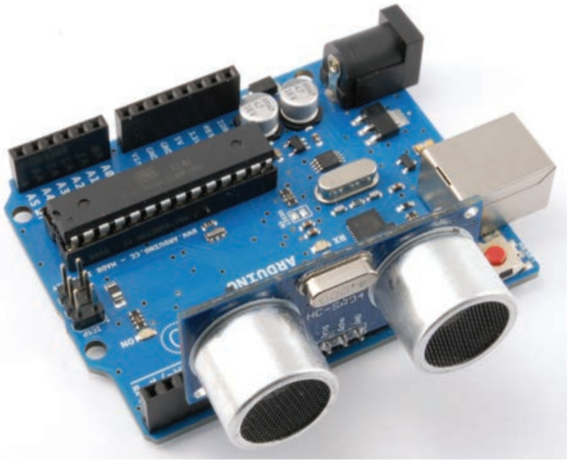
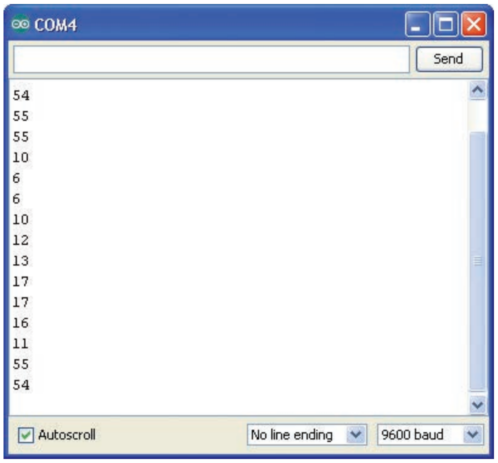


Figure 8-10 An HC-SR04 rangefinder on an Arduino

When you open the Serial Monitor, you will see a stream of distances in inches appear (Figure 8-11). Try pointing the rangefinder in different directions—say, a wall a few feet away—and confirm that the reading is reasonably accurate with a tape measure.



```
54
55
55
10
6
6
10
12
13
17
17
16
11
55
54
```

Figure 8-11 Distance readings in the Serial Monitor

The Arduino code for measuring the range is all contained within the "takeSounding_cm" function. This sends a single 10-microsecond pulse to the "trigger" pin of the ultrasonic module, which then uses the built-in Arduino function "pulseIn" to measure the time period before the echo pin goes high.

```
const int trigPin = 9;
const int echoPin = 10;
const int gndPin = 11;
const int plusPin = 8;

int lastDistance = 0;

void setup() {
  Serial.begin(9600);
  pinMode(trigPin, OUTPUT);
  pinMode(echoPin, INPUT);
  pinMode(gndPin, OUTPUT);
  digitalWrite(gndPin, LOW);
  pinMode(plusPin, OUTPUT);
  digitalWrite(plusPin, HIGH);
}
```

```
void loop() {
  Serial.println(takeSounding_in());
  delay(500);
}

int takeSounding_cm() {
  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);
  int duration = pulseIn(echoPin, HIGH);
  int distance = duration / 29 / 2;
  if (distance > 500) {
    return lastDistance;
  }
  else {
    lastDistance = distance;
    return distance;
  }
}

int takeSounding_in() {
  return takeSounding_cm() * 2 / 5;
}
```

We then need to convert that time in milliseconds into a distance in centimeters. If there is no reflection because there is no object that is close enough, or the object is reflecting the sound wave away rather than letting it bounce back to the receiver, then the time of the pulse will be very large and so the distance will also be recorded as very large.

To filter out these long readings, we disregard any measurement that is greater than 5m, returning that last sensible reading we got.

The speed of sound is roughly 343 m/s in dry air at 20 degrees C, or 34,300 cm/s.

Or, put another way, 34,300 / 1,000,000 cm / microsecond.

That is, 0.0343 cm/microsecond.

Put another way, 1/0.0343 microseconds/cm.

Or, 29.15 microseconds/cm.

Thus, a time of 291.5 microseconds would indicate a distance of 10 cm.

The “takeSounding_cm” function approximates 29.15 to 29 and then also divides the answer by 2, as we don’t want the distance of the whole return journey, just the distance to the subject.

In actual fact, many factors affect the speed of sound, so this approach will only ever give an approximate answer. The temperature and the humidity of the air will both affect the measurement.

In the later section “Making a Robot Rover with Raspberry Pi,” you will see how a rangfinder can be used with a Raspberry Pi.

Using a Wireless Remote Module

Radio frequency circuits usually are not worth making yourself when extremely useful modules like the one shown in Figure 8-12 are readily available for just a few dollars.

The module shown can be easily found on eBay and has a handy little key-fob–sized remote with four buttons on it. These buttons can toggle four digital pins on and off on the corresponding receiver module.

It is worth noting that modules like this are also available with relays instead of digital outputs, making it very easy to hack your own remote control projects.

You will first experiment with the module on breadboard, just turning on an LED, and then in the following section, you can try connecting it to an Arduino.

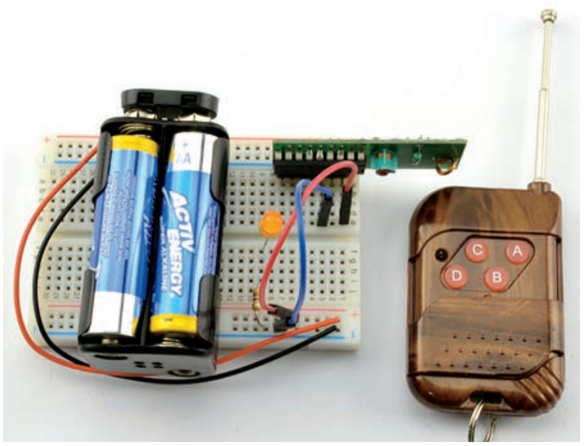


FIGURE 8-12 An RF module on breadboard

You Will Need

To try out the wireless remote on breadboard, you will need the following items.

Quantity	Names	Item	Appendix Code
1		Solderless breadboard	K1, T5
2		Male-to-male jumper wire or solid core wire	K1, T6
1		Wireless remote kit	M8

1	D1	LED	K1
1	R1	470Ω resistor	K1
1		4 ×AA battery holder	K1, H1
1		Battery clip	K1, H2
4		AA batteries	

Breadboard

Figure 8-13 shows the breadboard layout used to test the remote. You could, if you wished, add three more LEDs so there was one for each channel.

The datasheet for this module shows that the pins are as shown in Table 8-1.

Put the module on the breadboard with pin 1 at the top of the breadboard, and wire it up as shown in Figure 8-13.

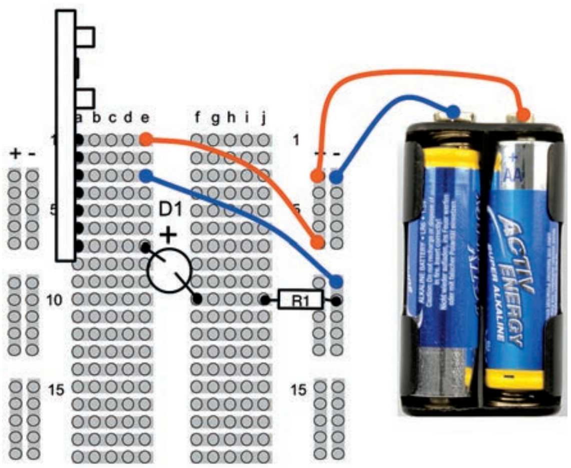


FIGURE 8-13 Breadboard layout for testing the RF module

Pin Number	Pin Name	Purpose
1	Vcc	Positive supply 4.5 to 7V
2	VT	Switch voltage—no connection needed
3	GND	Ground
4	D3	Digital output 3
5	D2	Digital output 2
6	D1	Digital output 1
7	D0	Digital output 0

TABLE 8-1 RF Receiver Pinout

That really is all there is to it. Pressing button A should toggle the LED on and off. If you wanted to, you could add more LEDs so there was one for each channel, or try moving the LED to a different output to check that they all work.

Using a Wireless Remote Module with Arduino

If we are prepared to lose one of the four channels of the remote from the previous section, then we can plug the receiver straight into the Arduino socket A0 to A5 (see Figure 8-14).

You Will Need

To try out the wireless remote with an Arduino, you will need the following items.

Quantity	Item	Appendix Code
1	Arduino Uno	M2
1	Wireless Remote Kit	M8

Before plugging the remote receiver into the Arduino, upload the sketch "ch08_rf_remote".

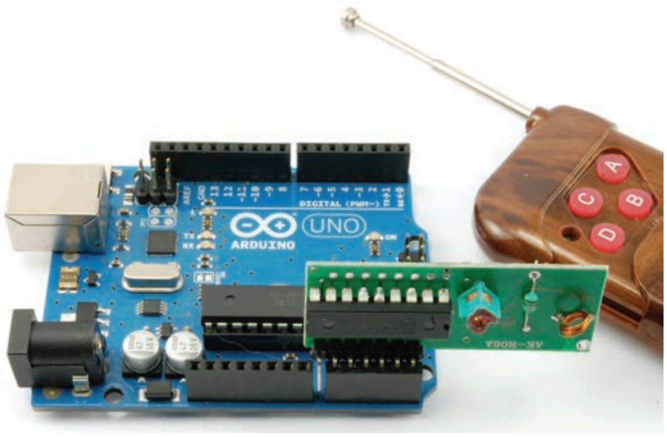


FIGURE 8-14 Using a RF remote with an Arduino

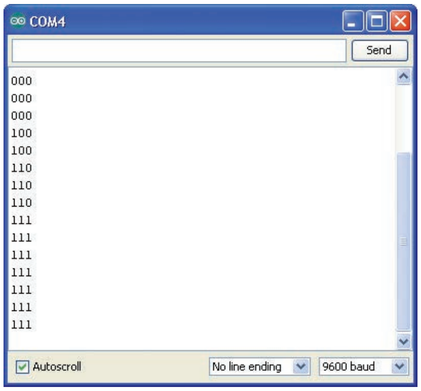
Software

With the software uploaded and the RF receiver attached, when you open the Serial Monitor you should see something like Figure 8-15.

The sketch displays as a 1 or 0 the current state of the remote control channels. So button A will not do anything (that is the button we sacrificed), but pressing the other buttons should toggle the appropriate column between 0 and 1.

```
const int gndPin = A3;
const int plusPin = A5;
const int bPin = A2;
const int cPin = A1;
const int dPin = A0;

void setup() {
  pinMode(gndPin, OUTPUT);
  digitalWrite(gndPin, LOW);
  pinMode(plusPin, OUTPUT);
  digitalWrite(plusPin, HIGH);
  pinMode(bPin, INPUT);
  pinMode(cPin, INPUT);
  pinMode(dPin, INPUT);
  Serial.begin(9600);
}

void loop() {
  Serial.print(digitalRead(bPin));
  Serial.print(digitalRead(cPin));
  Serial.println(digitalRead(dPin));
  delay(500);
}
```

FIGURE 8-15 Remote control messages to your computer



```
000
000
000
100
100
110
110
110
111
111
111
111
111
111
111
```

The RF receiver uses very little current, so there is no problem powering it from a digital output. In fact, doing so has the added benefit that we can actually turn it off to save power simply by setting the "plusPin" low.

Using a Motor Control Module

You can use a MOSFET to control the speed of a motor. This is fine as long as you always want the motor to turn in the same direction. If you want to be able to reverse the direction of the motor, you need to use something called an H-Bridge.

To change the direction in which a motor turns, you have to reverse the direction in which the current flows. To do this requires four switches or transistors. Figure 8-16 shows how this works, using switches in an arrangement. You can now see why it is called an H-Bridge.

In Figure 8-16, S1 and S4 are closed, while S2 and S3 are open. This allows current to flow through the motor with terminal "A" positive and terminal "B" negative. If we were to reverse the switches so that S2 and S3 are closed and S1 and S4 are open, then "B" will be positive and "A" will be negative, and the motor will turn in the opposite direction.

You may, however, have spotted a danger with this circuit. That is, if by some chance S1 and S2 are both closed, then the positive supply will be directly connected to the negative supply and we will have a short circuit. The same is true if S3 and S4 are both closed at the same time.

You can build an H-Bridge yourself using transistors, and Figure 8-17 shows a typical H-Bridge schematic.

This schematic requires some six transistors and a good few other components. If you wanted to control two motors, you would need some 12 transistors, which causes everything to become quite complicated.

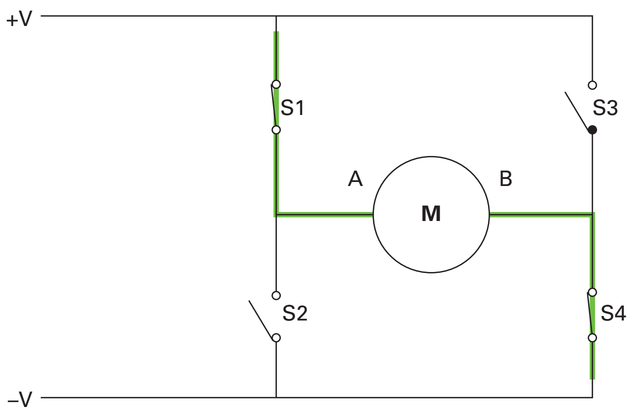


Figure 8-16 An H-Bridge using switches

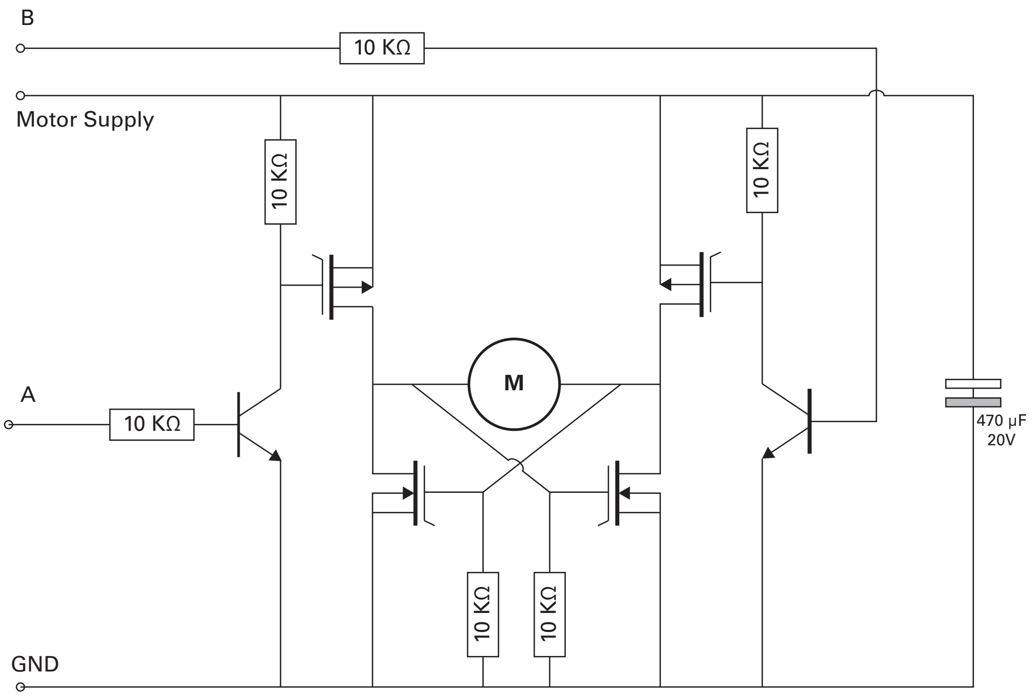


FIGURE 8-17 An example schematic for an H-Bridge

Fortunately, help is on hand as there are several H-Bridge ICs available that usually have two H-Bridges on a single chip and make controlling motors very easy. One such chip is available as a module from SparkFun (Figure 8-18). You will find similar modules available from other module suppliers.

Figure 8-18 actually shows two of these modules so you can see both sides. The modules are supplied without connectors and the module on the left has pin headers soldered to it. This makes it very easy to use with breadboard.

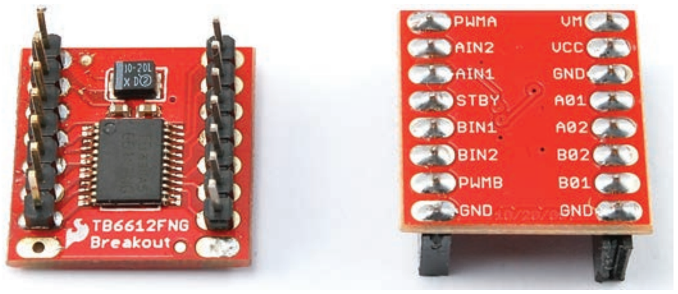


FIGURE 8-18 A SparkFun H-Bridge module

Pin Name	Purpose	Purpose	Pin Name
PWMA	PWM input for Channel A	Motor supply voltage (VCC to 15V)	VM
AIN2	Control input 2 for A; high for counter-clockwise	Logic supply (2.7 to 5.5V); only requires 2mA	VCC
AIN1	Control input 1 for A; high for clockwise		GND
STBY	To connect to GND to put the device into “standby” mode.	Motor A connection 1	A01
BIN1	Control input 1 for B; high for clockwise	Motor A connection 2	A02
BIN2	Control input 2 for B; high for counterclockwise	Motor B connection 2	B02
PWMB	PWM input for Channel A	Motor B connection 1	B01
GND			GND

TABLE 8-2 The SparkFun TB6612FNG Breakout Board Pinout

Table 8-2 shows the pins of this module and explains the purpose of each. The module has two motor channels called A and B and can drive motors with a current of 1.2A per channel with peak currents of over twice that.

We will experiment with this module using just one of its two H-Bridge channels (Figure 8-19).

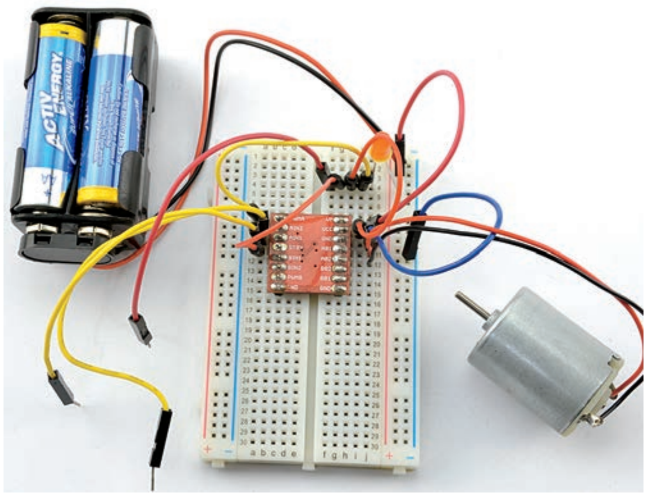


FIGURE 8-19 Experimenting with the SparkFun TB6612FNG breakout board

You Will Need

To build this, you will need the following items.

Quantity	Item	Appendix Code
1	Solderless breadboard	K1, T5
7	Male-to-male jumper wire or solid core wire	K1, T6
1	4 × AA battery holder	K1, H1
1	4 × AA batteries	
1	Battery clip	K1, H2
1	LED	K1
1	SparkFun TB6612FNG Breakout Board	M9
1	6V DC motor or gear motor	K1, H6
1	Header pins	K1, H4

The DC motor can be any small motor around 6V.

Breadboard

Before fitting the module onto the breadboard, you need to solder the header pins into place as shown in Figure 8-18. We won't use the bottom two GND connections, so you can just solder the top seven pins on each side.

Figure 8-20 shows the schematic diagram for the experiment, while Figure 8-21 displays the breadboard layout.

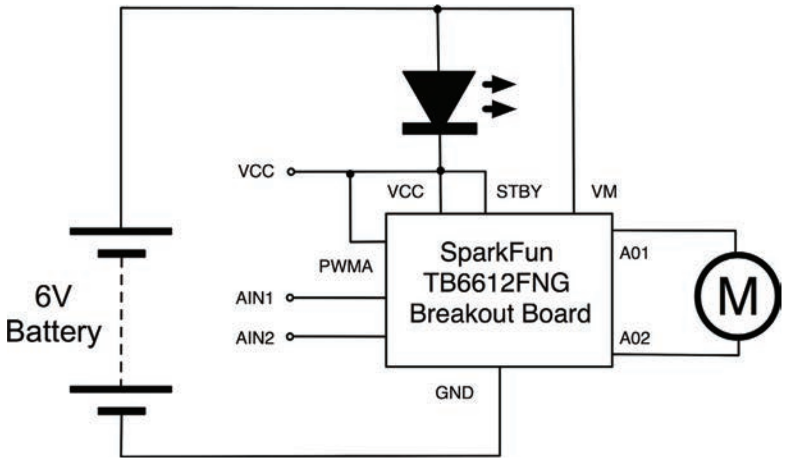


FIGURE 8-20 Schematic diagram for H-Bridge experiment

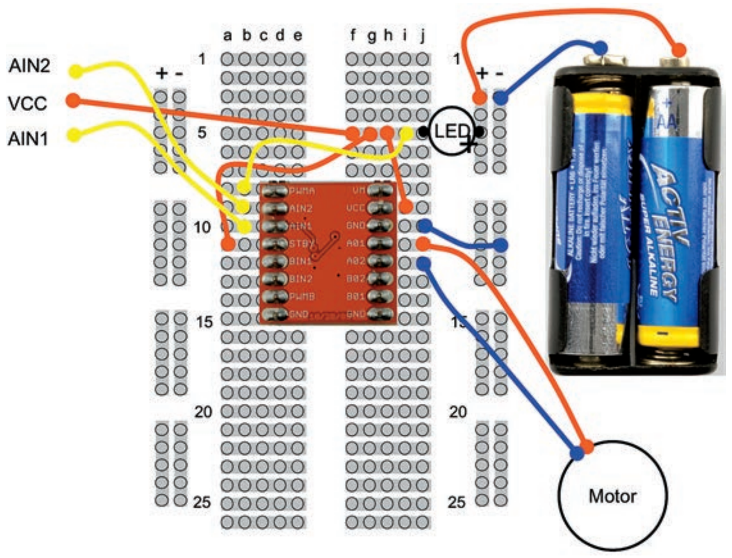


FIGURE 8-21 Breadboard layout for the H-Bridge experiment

The 6V battery pack is actually a slightly higher voltage than is (strictly speaking) allowed for VCC on the module. You would probably get away with the extra half volt above the nominal maximum voltage of 5.5V, but to play it safe, we can use an LED to drop 2V, so that VCC will be around 4V, which is well within its range.

This is a useful trick, but only use it when the current flowing is less than the maximum forward current of the LED. In fact, in this experiment, the current required for VCC is not even enough to make the LED glow.

The PWMA pin is connected to VCC, which simulates the PWM control signal being on all the time—in other words, there is full power to the motor.

Next, put everything on the breadboard as shown in Figure 8-21.

Using the Control Pins

Three of the leads from the breadboard do not actually go anywhere. You will control the motor, touching the red lead going to VCC to AIN1, and then to AIN2, in turn. Note how the motor turns first in one direction and then the other.

You might be wondering why there are two control pins, as well as the PWM pin for each motor channel. In theory, you could have one direction pin and one PWM pin, and if the PWM power was zero, then the motor would not turn at all.

The reason we have three pins to control each motor (PWM, IN1, and IN2) rather than just two is that if both IN1 and IN2 are high (connected to VCC), then the H-Bridge operates in a "braking" mode, which provides electrical braking of the motor, slowing it down. This feature is not often used, but can be useful if you want to stop the motor quickly.

Making a Robot Rover with Raspberry Pi

The same TB6612FNG motor controller IC described in the previous section is also available as an add-on board for the Raspberry Pi. This clips over the GPIO connector of the Raspberry Pi and allows you to control the speed and direction of two motors. You can use this with a chassis and motor kit, to make a roving robot like the one shown in Figure 8-22.

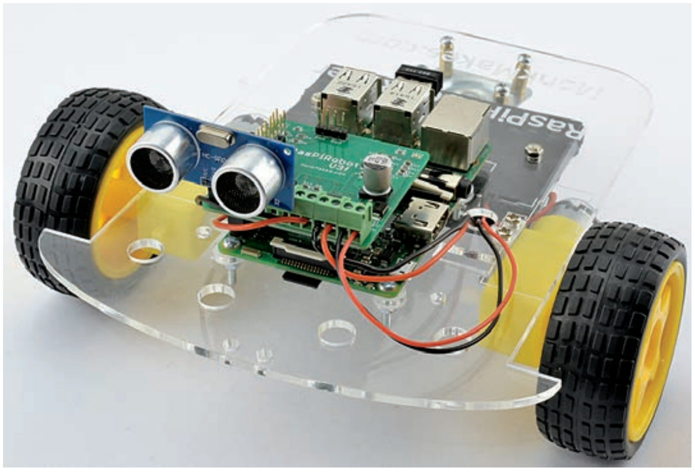


FIGURE 8-22 A Raspberry Pi robot

You Will Need

To make this project, you will need the following items.

Quantity	Item	Appendix Code
1	Raspberry Pi 1, 2 or 3	M11
1	Robot chassis kit	H7
1	MonkMakes RasPiRobot V3 motor controller	M13
1	6xAA battery holder	H8
1	HC-SR04 Ultrasonic rangefinder	M7

A kit of parts is also available from MonkMakes that provides all of these parts except for the Raspberry Pi itself (see the Appendix).

Because the rover needs to run free from any wires, you will need to have the Raspberry Pi that you intend to use with the rover set up to use WiFi and SSH as described in Chapter 7. So if you have a Raspberry Pi older than the Raspberry Pi 3 then you will need a separate USB WiFi module.

Construction

The first step in building this rover is to construct the plastic chassis and gearmotors. These chassis are usually supplied with instructions. Some are single-layer and others like the Magician chassis shown in Figure 8-23 are double layer, allowing you to fit the batteries below the Raspberry Pi.

Having assembled the chassis, you need to work out where the battery box and Raspberry Pi will fit on the chassis. Put the RasPiRobot Board V3 (Figure 8-24) on your Raspberry Pi GPIO connector so that you can be sure that the leads from the motors and battery box will reach to the RasPiRobot Board V3. The RasPiRobot Board V3 fits over the top 26 connections of the Raspberry Pi 2 and 3 and covers the whole connector of a Raspberry Pi 1.

The connections that need to be made to the screw terminals of the RasPiRobot Board V3 are shown in Figure 8-25.

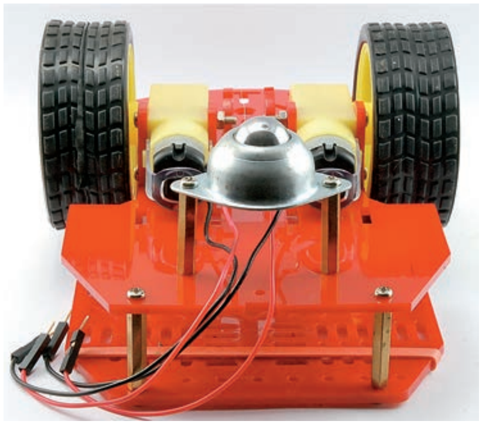


FIGURE 8-23 The magician chassis

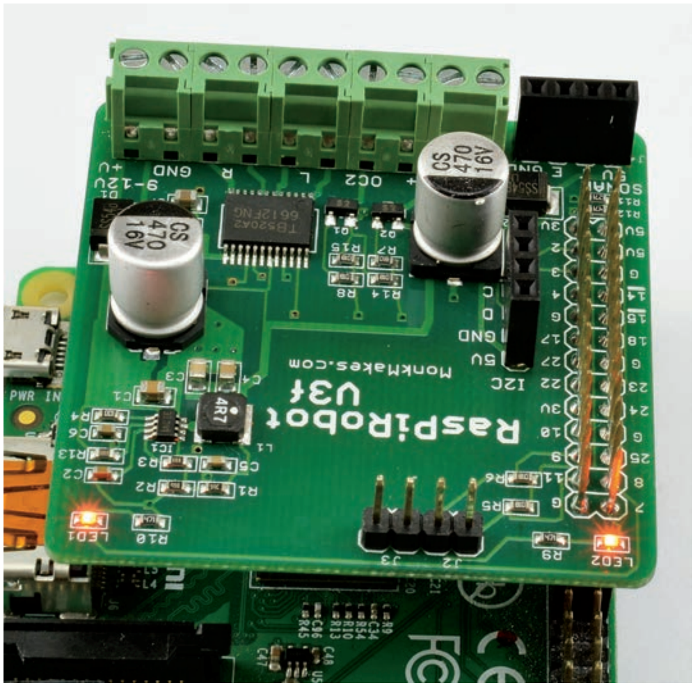


FIGURE 8-24 Fitting the RasPiRobot Board V3 onto a Raspberry Pi 3

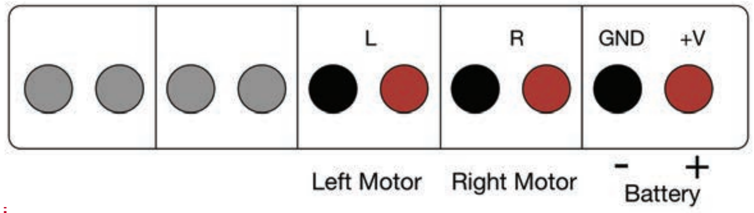


FIGURE 8-25 Connections to the RasPiRobot Board V3

As well as being able to control motors, the RasPiRobot Board V3 also provides a voltage regulator that will supply power to your Raspberry Pi from the battery pack, so that you do not need to have a separate USB power lead when powered from batteries. However, to increase the life of your batteries, you should continue to power the Raspberry Pi over USB while you get the software installed and make everything ready.

You can now also attach the HC-SR04 ultrasonic rangefinder to the socket on the RasPiRobot Board V3 labeled "Sonar" (Figure 8-26).

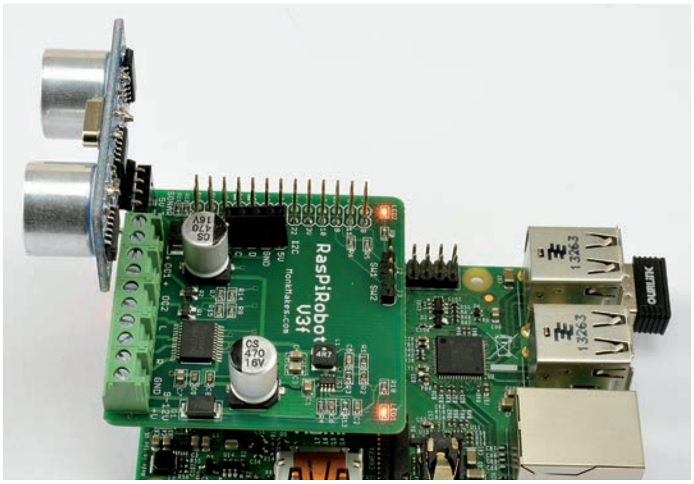


Figure 8-26 The Ultrasonic Rangefinder Attached to the RasPiRobot Board V3

Software

The RasPiRobot Board V3 has an accompanying library that you need to install by running the following commands:

```
$ cd /home/pi
$ git clone https://github.com/simonmonk/
    raspirobotboard3.git
$ cd raspirobotboard3/python
$ sudo python setup.py install
```

This library includes example files, which you will find as a directory called “examples” inside the “raspirobotboard3” directory. You can list the example programs using the command:

```
$ cd /home/pi/python/examples
$ ls
```

Once installed, you can check that the motors and battery are working ok by using one of the example programs included with the RasPiRobot Board V3. Turn the rover on its back so it can’t drive off unexpectedly and run the test program “test_motors.py”. You should find that each of the two motors turns at different seemingly random speeds and directions for a few seconds at a time. Note that the motors will only turn under battery power, so if you are just powering the Raspberry Pi over its USB lead, now is the time to put batteries in the battery box.

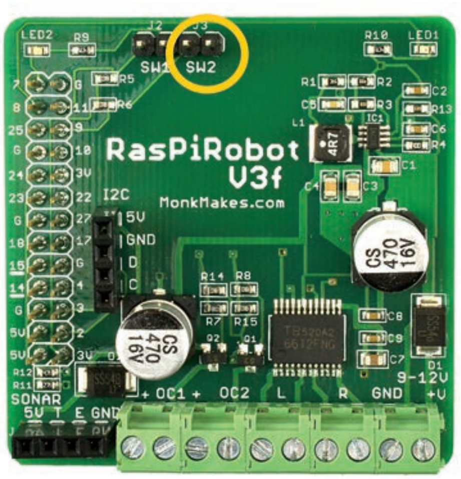


FIGURE 8-27 The SW2 contacts on the RasPiRobot Board V3

There are a number of different programs in the "examples" directory that you might like to try, but a good starting point is rover_avoiding.py. This program sets the rover moving forward until the rangefinder detects an obstacle, at which time it reverses a bit and then turns a random amount before setting off again.

Run the program using the command below using an SSH connection over WiFi:

```
$ python rover_avoiding.py
```

The rover won't actually start moving until you use a metal screw-driver to momentarily connect together the two contacts marked SW2 on the RasPiRobot Board V3 (see Figure 8-27). If you want a neater solution than poking the pins with a screwdriver, then you could connect a switch to the contacts.

When you are ready to stop the rover, pick it up and short the SW2 contacts again to stop the motors. The batteries will still be powering the Raspberry Pi, so to power down the Raspberry Pi, issue the following command and then flip one of the batteries out of the battery box.

```
$ sudo halt
```

The code for rover_avoiding.py is listed below:

```
from rrb3 import *
import time, random

BATTERY_VOLTS = 9
MOTOR_VOLTS = 6

rr = RRB3(BATTERY_VOLTS, MOTOR_VOLTS)

# if you don't have a switch, change the value
# below to True
running = False

def turn_randomly():
    turn_time = random.randint(1, 3)
    if random.randint(1, 2) == 1:
        rr.left(turn_time, 0.5) # turn at half
            speed
    else:
        rr.right(turn_time, 0.5)
    rr.stop()

try:
    while True:
        distance = rr.get_distance()
        print(distance)
        if distance < 50 and running:
            turn_randomly()
        if running:
            rr.forward(0)
        if rr.sw2_closed():
            running = not running
        if not running:
            rr.stop()
        time.sleep(0.2)
finally:
    print("Exiting")
    rr.cleanup()
```

After importing the rrb3, time, and random libraries, the code defines two variables BATTERY_VOLTS and MOTOR_VOLTS. In this case we are using a 9V battery with 6V motors and so those two values can be used by the rrb3 library to use PWM to adjust the power going to the motors so as not to damage them.

Those variables are then used in the command that initializes a variable *rr* that will be used to gain access to all the features of the RasPiRobot Board V3.

The variable *running* is used to keep track of whether the contacts of SW2 have been shorted to start the rover.

The function *turn_randomly*, as the name suggests, selects a random amount of time between 1 and 3 seconds and assigns it to the variable *turn_time*. It then chooses the direction in which to turn and calls *rr.left* or *rr.right* with *turn_time* and 0.5 (half speed) to make the rover turn.

The main loop of the program measures the distance to any obstacle using *rr.get_distance* and, if that distance is less than 50cm, calls *turn_randomly*. Otherwise, it just moves forward, also checking for a closing of the contacts of SW2 using *rr.sw2_closed*.

Finally, when the program quits, it calls *rr.cleanup*, which clears the GPIO pins back to inputs.

Using a Seven-Segment Display with Arduino

Seven-segment LED displays have a nice retro feel to them.

LED displays made up of a number of LEDs contained in a single package can be a challenge to control. Such displays will normally be controlled using a microcontroller; however, it is not necessary to connect a microcontroller output pin to each individual LED. But rather, multi-LED displays are organized as "common anode" or "common cathode," with all the LED terminals of the anode or cathode connected together and then brought out through one pin. Figure 8-28 shows how a common cathode seven-segment display might be wired internally.

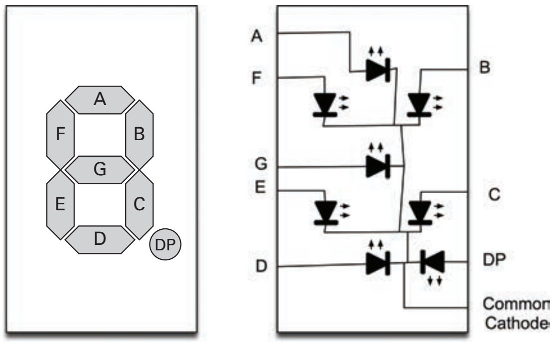
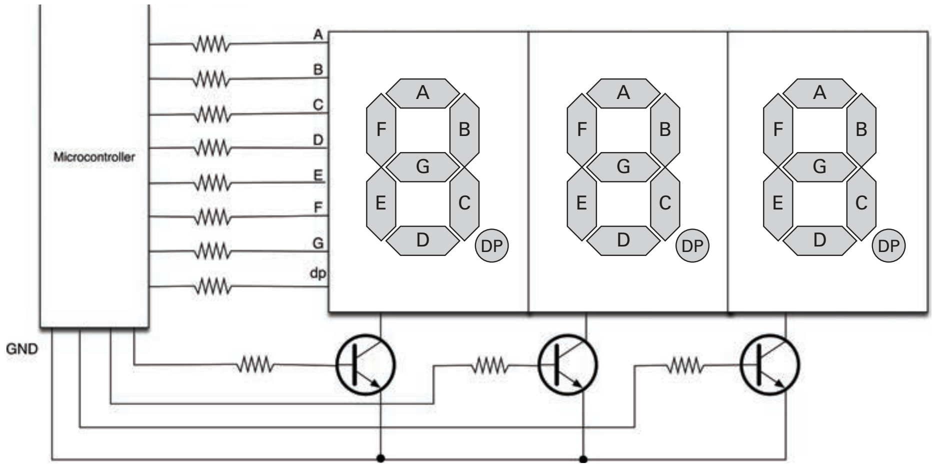


FIGURE 8-28 A common cathode LED display

In a common cathode display like this, the common cathode would be connected to ground and each segment anode driven by a microcontroller pin through a separate resistor. Do not be tempted to use one resistor on the common pin, and don't use any resistors on the non-common connections, since the current will be limited no matter how many LEDs are lit. Because of this, the display will get dimmer the more LEDs are illuminated.

It is quite common for multiple displays to be contained in the same case—for example, the three-digit, seven-segment common cathode LED display shown in Figure 8-29.

FIGURE 8-29 A three-digit, seven-segment LED display



In this kind of display, each digit of the display is like the single-digit display of Figure 8-29, and has its own common cathode. In addition, all the A segment anodes are connected together, as are each segment.

The Arduino using the display will then activate each common cathode in turn and then turn on the appropriate segments for that digit, and then move onto the next digit, and so on. This refresh happens very quickly so that the display appears to display different numbers on each digit. This is called multiplexing.

Note the use of transistors to control the common cathodes. This is simply to handle the current of potentially eight LEDs at once, which would be too much for most microcontrollers.

Fortunately for us, there is a much simpler way to use multi-digit, seven-segment LED displays. Modules ride to the rescue once again!

Figure 8-30 shows a four-digit, seven-segment LED display that has just four pins on its connector, and two of them are for power.

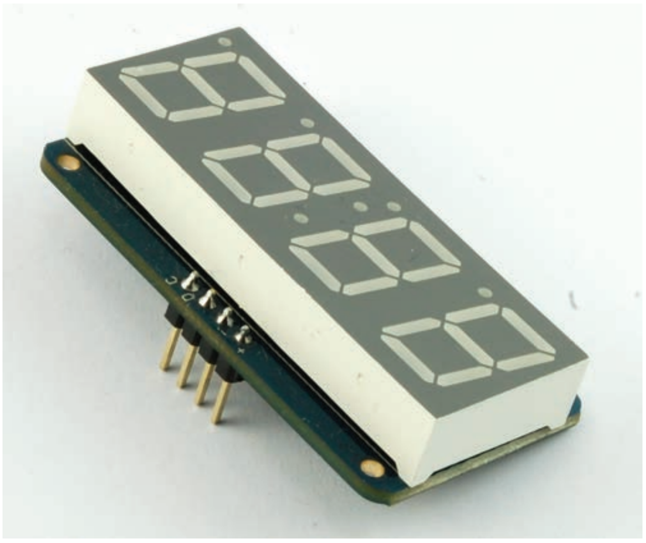


Figure 8-30 A four-digit, seven-segment I2C display

You Will Need

To build this, you will need the following items.

Quantity	Item	Appendix Code
1	Solderless breadboard	K1, T5
4	Male-to-female jumper wires	K1, T12
1	Arduino Uno R3	M2
1	Adafruit seven-segment display w/I2C backpack	M19

Construction

The module comes as a kit, so start by following the instructions that accompany the module to assemble it.

The LED module uses a type of serial interface on the Arduino called I2C (pronounced "I squared C"). This requires just two pins, but they have to be the two pins above "AREF" on the Arduino Uno. These pins are named SDA and SCL.

The easiest way to connect the display to an Arduino is using male-to-female jumper wires as shown in Figure 8-31.

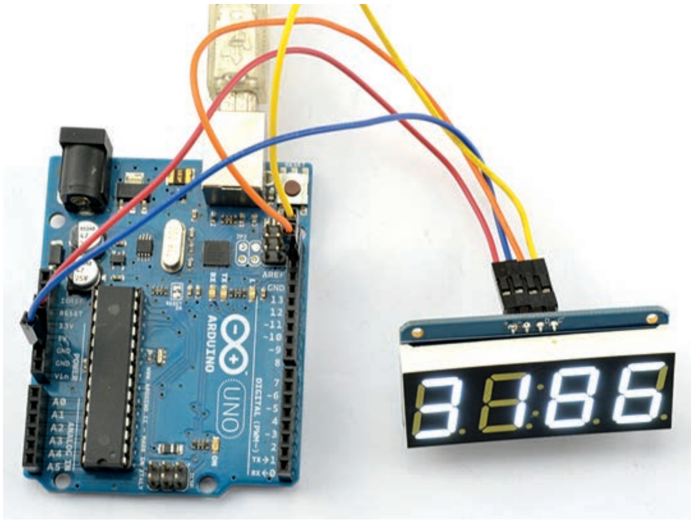


FIGURE 8-31 The seven-segment display in action

Software

Adafruit provides a library to simplify the use of the module. To install the Adafruit library, from the sketch menu of Arduino IDE, select Include Library and then Manage Libraries to open the Libraries Manager. Type “backpack” into the search field and then select “Adafruit LED Backpack Library” from the search results and then click on Install.

The test sketch for this is called “ch08_deven_seg_display”.

The three libraries that the module requires are loaded using the #includes statements.

```
#include <Wire.h>
#include "Adafruit_LEDBackpack.h"
#include "Adafruit_GFX.h"
```

The following line assigns a variable to the display object so we can tell it what to display.

```
Adafruit_7segment disp = Adafruit_7segment();
```

The “setup” function begins serial communication on the I2C pins and then initializes the display. The value 0x70 is the I2C address of the display module. This is the default value for its address, but there are solder connections on the module you can short together to change the address. You might want to do this if you need to use more than one display, since each display must have a different address.

```
void setup()
{
  Wire.begin();
  disp.begin(0x70);
}
```

The “loop” function simply displays the current number of milliseconds since the board was reset, divided by 10. The display will therefore count up in 1/100ths of a second.

```
void loop()
{
  disp.print(millis() / 10);
  disp.writeDisplay();
  delay(10);
}
```

Using a Seven-Segment Display with Raspberry Pi

In the previous section, you saw how a seven-segment display can be interfaced with an Arduino. These devices work just as well with a Raspberry Pi and because a Raspberry Pi can connect to the Internet to find the time, you can have the Raspberry Pi display the time as shown in Figure 8-32.

Since updating the display does not take much effort on the part of the Raspberry Pi, there is no reason why it shouldn't also be doing other things, such as browsing the Internet or playing music while it is displaying the time.

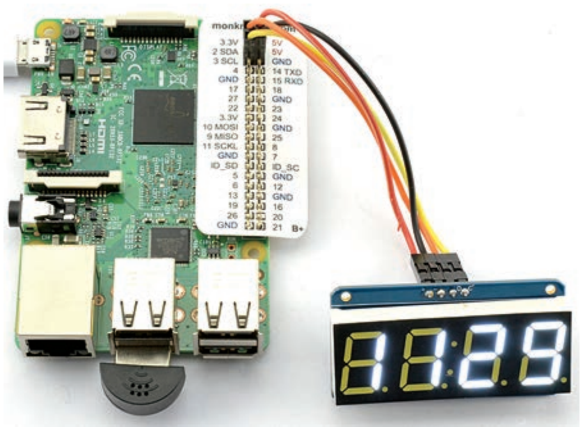


FIGURE 8-32 A Raspberry Pi displaying the time

You Will Need

To make this project, you will need the following items.

Quantity	Item	Appendix Code
1	Raspberry Pi 1, 2 or 3	M11
1	Adafruit seven-segment display w/I2C backpack	M19
4	Female-to-female jumper wires	K1, T14

Construction

This is a matter of connecting the jumper wires between the Raspberry Pi and display as follows:

- VCC on the display to 5V on the Raspberry Pi
- GND to GND
- SCL on the display to SCL on the Raspberry Pi
- SDA on the display to SDA on the Raspberry Pi

Software

Adafruit has a library for using the displays with Raspberry Pi. To install it and the other software that it needs, run the following commands:

```
$ sudo apt-get update
$ sudo apt-get install build-essential python-dev
$ sudo  apt-get install python-smbus python-imaging
$ git clone https://github.com/adafruit/Adafruit_
                    Python_LED_Backpack.git
$ cd Adafruit_Python_LED_Backpack
$ sudo python setup.py install
```

The library comes with a clock example, so there is little point in writing your own. You can find this inside the "examples" folder in Adafruit_Python_LED_Backpack and run it using the command:

```
$ cd examples
$ python ex_7segment_clock.py
```

If your clock is out by a few hours, then it probably means that your Raspberry Pi is set to the wrong time zone. To change the time zone of your Raspberry Pi, use the Raspberry Pi Configuration tool from the Preference section of the Start menu. Click on the Localisation tab and then Set Timezone.

Using RFID Modules

If you've ever wanted to read a smartcard or RFID tag, then you can do so surprisingly easily with a $5 RFID reader/writer and Raspberry Pi, as shown in Figure 8-33.

This hardware is also capable of writing onto RFID cards and tags that have that capability, but don't imagine that you will be able to top up your transport card for free. RFID cards generally employ encryption to stop that sort of thing.

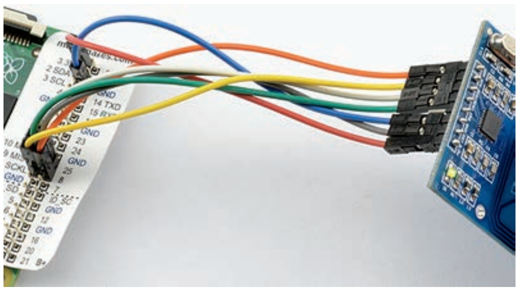


FIGURE 8-33 RFID reader/writer connected to a Raspberry Pi

You Will Need

To make this project, you will need the following items.

Quantity	Item	Appendix Code
1	Raspberry Pi 1, 2 or 3	M11
1	RC522 RFID Proximity reader/writer	M20
7	Female-to-female jumper wires	K1, T14

MonkMakes also provide a kit that includes these components (http://monkmakes.com/cck).

Construction

The RFID reader pins can be connected directly to the Raspberry Pi GPIO pins as listed in Table 8-3. Note that the RFID reader/writer is a 3.3V device, so VCC on the module must be connected to 3.3V and NOT 5V. I have also suggested color-coding for the leads, to help keep track of them.

Lead Color	Smartcard Reader	Raspberry Pi Pin
Orange	SDA	8
Yellow	SCK	11
White	MOSI	10
Green	MISO	9
	IRQ	Not connected
Blue	GND	GND
Gray	RST	25
Red	3.3V	3.3V

TABLE 8-3 Pin Connections from an RFID Module to Raspberry Pi

Software

If you remember back in Chapter 7, you enabled SSH on your Raspberry Pi, using the Raspberry Pi Configuration tool. I suggested that while you were there, you also enable the SPI and I2C interfaces. Well, the RFID module uses the SPI interface, so if you didn't enable it, go back and do so now.

You also need to install a Python library for the SPI interface and for the RFID module in particular using the commands:

```
$ sudo apt-get install python-dev
$ git clone https://github.com/simonmonk/
                    SPI-Py.git
$ cd SPI-Py
$ sudo python setup.py install
```

The example program "rfid_read.py" found in the directory "ch08_rfid_read" will wait until an RFID card comes close to the reader and then print out the ID of the card. The program is contained in a folder, because there are a couple of other helper Python files that handle the communication through the SPI interface.

Run the program using the command:

```
$ python rfid_read.py
```

When you run the program and hold a tag near the RFID module, you should see a number printed out a number of times, as the reader repeatedly reads the tag until you move it out of range.

```
$ python rfid_read.py
Hold a tag near the reader
905537575667
905537575667
905537575667
905537575667
```

Depending on the tag, you may also see authentication errors. This indicates that the data on the tag is protected.

Here is the listing for rfid_read.py.

```
import RPi.GPIO as GPIO
import SimpleMFRC522

reader = SimpleMFRC522.SimpleMFRC522()

print("Hold a tag near the reader")

try:
    while True:
        tag = reader.read()
        print(tag['id'])
        print(tag['text'])

finally:
    print("cleaning up")
    GPIO.cleanup()
```

The library function “read” waits until a tag is presented to a reader and then it reads its ID. It also attempts to read data from the card, for which it attempts to authenticate using the default key. This only succeeds if the tag’s security key has not been set. Both of these values are sorted in a Python table. You can then access the ID and data using tag[‘id’] and tag[‘text’], respectively.

RFID Tags

RFID tags are all allocated a unique ID at the time of their manufacture that cannot be changed. Often, in RFID applications, it is just this very large number that is used to identify the item being scanned, rather like a barcode. However, many tags also allow a small amount of data (often 1kB) of data to be stored on the tag. To gain access to these data, a key is required.

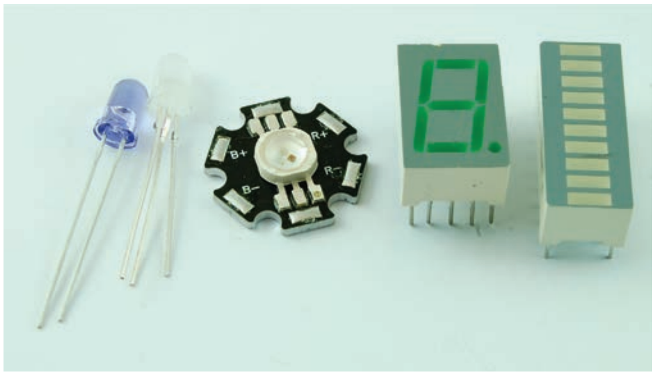
RFID tags are interesting because they are made from an IC and antenna, but no battery. What happens is that the card reader acts as a low-power transmitter and some of the energy from the transmission is received through the tag’s antenna and used to power it for the short while it needs to interact with the RFID reader/writer.

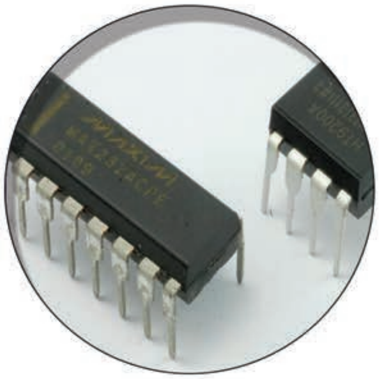
There are several different tag frequencies in use. The most common and the one used by this reader operates at a frequency of 13.56MHz.

Summary

In addition to the modules here, you will find lots of other useful modules on the web sites of companies like Adafruit and SparkFun. The web sites also include some information on how to use the modules and their specifications. If you find a module you would like to make use of, the first step is to research how you could use it. As well as the datasheets and tutorial information on the supplier’s web site, you will often find instructions on building the projects if you search for the module on the Internet.

ARDUINO
UNO
DIGITAL (PWM~)


B+
R+
R-
B-



9

Hacking with Sensors

Sensors are often used with an Arduino or Raspberry Pi to make physical measurements of temperature, light, or acceleration and can all be digitized into a form for use by the Arduino or Raspberry Pi, which can then do things like display the data or perform certain actions depending on the sensor readings.

For simple on/off control, say, for a thermostat, then you may not need an Arduino or Raspberry Pi. You might be able to just use an IC called a comparator to compare the sensor reading with a set value and perform the switching action.

Measuring Temperature with an Arduino

A number of different sensor ICs are designed for measuring temperature. Perhaps the simplest to use is the TMP36 (Figure 9-1).

You can experiment with the sensor, just printing the temperature to the Serial Monitor, or you can combine the sensor with a relay module to switch things on and off.

You Will Need

To experiment with temperature measurement and an Arduino, you will need the following items.

Quantity	Item	Appendix Code
1	Arduino Uno R3	M2
	TMP36	K1, S8

Construction

The TMP36 has just three pins, two for the power supply and one analog output. The power supply needs to be between 2.7V and 5.5V, making it ideal for use with the 5V of an Arduino. In

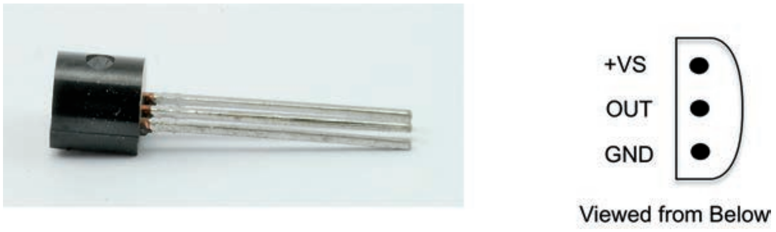


Figure 9-1 The TMP36 temperature sensor IC

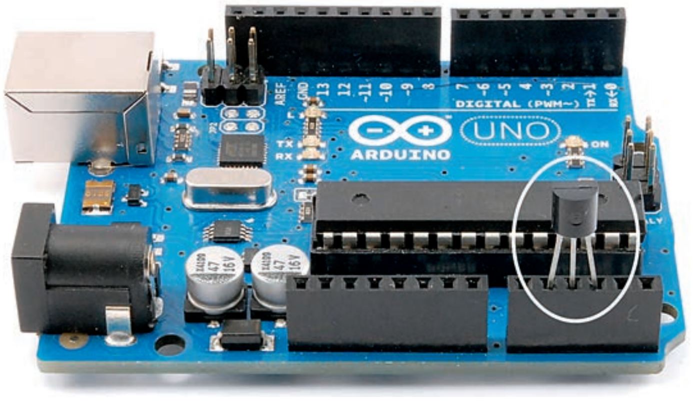


Figure 9-2 The TMP36 attached to an Arduino

fact, we can supply the power to it through digital outputs and just plug the whole chip into three pins on the analog connector of the Arduino (Figure 9-2).

Software

The sketch ("temperature_sensor") follows what should now be a fairly familiar pattern. The pins are defined, and then in the *setup* function the output pins that provide power to the sensor are set to LOW for GND and HIGH for the positive supply.

```
const int gndPin = A1;
const int sensePin = 2;
const int plusPin = A3;

void setup() {
  pinMode(gndPin, OUTPUT);
  digitalWrite(gndPin, LOW);
  pinMode(plusPin, OUTPUT);
  digitalWrite(plusPin, HIGH);
  Serial.begin(9600);
}
```

The loop function reads the value from the analog input and then does a bit of arithmetic to calculate the actual temperature.

First, the voltage at the analog input is calculated. This will be the raw value (between 0 and 1023) divided by 205. It is divided by 205 because a span of 1024 values occupies 5V, or:

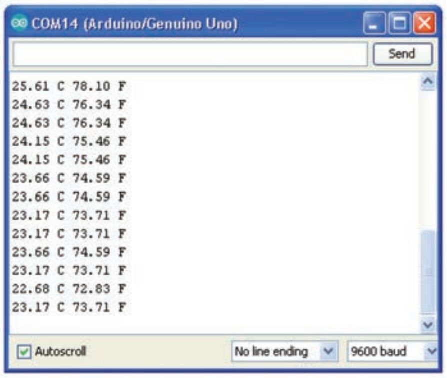
1024 / 5 = 205 per volt

The TMP36 outputs a voltage from which the temperature in degrees C can be calculated from the equation:

tempC = 100.0 * volts – 50

For good measure, the sketch also converts this into degrees F and prints both out to the Serial Monitor.

```
void loop() {
  int raw = analogRead(sensePin);
  float volts = raw / 205.0;
  float tempC = 100.0 * volts - 50;
  float tempF = tempC * 9.0 / 5.0 + 32.0;
  Serial.print(tempC);
  Serial.print(" C ");
  Serial.print(tempF);
  Serial.println(" F");
  delay(1000);
}
```



```
25.61 C 78.10 F
24.63 C 76.34 F
24.63 C 76.34 F
24.15 C 75.46 F
24.15 C 75.46 F
23.66 C 74.59 F
23.66 C 74.59 F
23.17 C 73.71 F
23.17 C 73.71 F
23.66 C 74.59 F
23.17 C 73.71 F
22.68 C 72.83 F
23.17 C 73.71 F
```

FIGURE 9-3 Displaying temperatures in the Serial Monitor

Open up the Serial monitor and you should see a series of temperatures being printed like those of Figure 9-3. Try pinching the TMP36 between your fingers to raise the temperature and notice how the readings change.

You could change this example so that it only displayed the temperature if it exceeded a certain value (let's say 25C). To do this, you just need to add an if statement to loop so that it now looks like this:

```
void loop() {
  int raw = analogRead(sensePin);
  float volts = raw / 205.0;
  float tempC = 100.0 * volts - 50;
  float tempF = tempC * 9.0 / 5.0 + 32.0;
  if (tempC > 25.0) {
    Serial.print(tempC);
    Serial.print(" C ");
    Serial.print(tempF);
    Serial.println(" F");
    delay(1000);
  }
}
```

Try modifying the sketch and uploading it again.

The Raspberry Pi does not have analog inputs, so you cannot attach a sensor like the TMP36 to it in the same way as in the previous section with an Arduino. But, that does not mean that you cannot use it with sensors.

In the following sections you will discover a few different ways of using sensors with a Raspberry Pi.

Threshold Sensing with Raspberry Pi

If you have a sensor like a photoresistor whose resistance varies with the amount of light falling on it, and you are only worried about being able to detect when the resistance is above or below some threshold level, then you can use the sensor in a voltage divider arrangement with a variable resistor and a digital input.

You Will Need

To experiment with light sensing and a Raspberry Pi, you will need the following items.

Quantity	Item	Appendix Code
1	Raspberry Pi (any model)	M11
1	Photoresistor	K1, R2
1	10kΩ trimpot variable resistor	K1, R1
1	Solderless breadboard	K1, T5
1	Male-to-male jumper wire or solid core wire	K1, T6
3	Male-to-female jumper wires	K1, T12

Construction

The digital input of a Raspberry Pi 3 counts as LOW if it is below about 1.2V and HIGH if it's above 1.2V. So you can use a photoresistor with a variable resistor as shown in Figure 9-4 to detect light and dark with a Raspberry Pi. Figure 9-5 shows the experiment on breadboard.

When connecting the positive supply to the breadboard, remember that it must go to the 3.3V pin on the Raspberry Pi and NOT the 5V pin; otherwise, you might damage the Raspberry Pi.

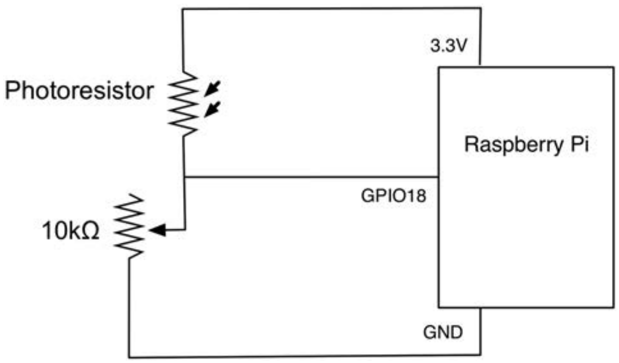


FIGURE 9-4 The schematic for detecting light and dark with a photoresistor and Raspberry Pi

Software

The Python program to detect light and dark is "ch09_light_detect.py". Run it and then, keeping your hand away from the photoresistor (so as not to shade it), adjust the variable resistor so that the last message is "It got light" and then let go of the variable resistor. You should find that when you shade the

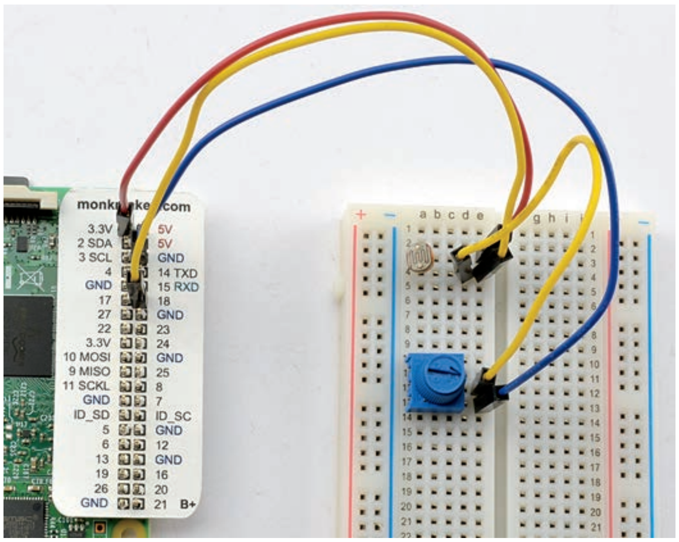


FIGURE 9-5 Connecting a photoresistor to a Raspberry Pi 3 on breadboard

photoresistor with your hand you see a message saying, “It went dark,” then moving your hand out of the way, it will say “It got light” again.

```
$ python ch09_light_detect.py
It got light
It went dark
It got light
It went dark
It got light
It went dark
```

The program “ch09_light_detect.py” is listed below:

```
import RPi.GPIO as GPIO

GPIO.setmode(GPIO.BCM)
sensor_pin = 18
GPIO.setup(sensor_pin, GPIO.IN)

was_light = False

try:
    while True:
        is_light = GPIO.input(sensor_pin)
        if (is_light == True) and
           (was_light == False):
            print("It got light")
            was_light = True
        elif (is_light == False) and
             (was_light == True):
            print("It went dark")
            was_light = False

finally:
    print("Cleaning up")
    GPIO.cleanup()
```

The program sets GPIO 18 to be an input. It also makes use of a variable called was_light so that messages only appear when the light level changes from light to dark or vice-versa. You can see how this variable is used in the if statement. Here the message “It got light” is only printed if it currently is light (sensor_pin is HIGH) and it was previously dark (was_light == False). Having printed the message, the variable was_light is then updated. A similar thing happens when changing from light to dark.

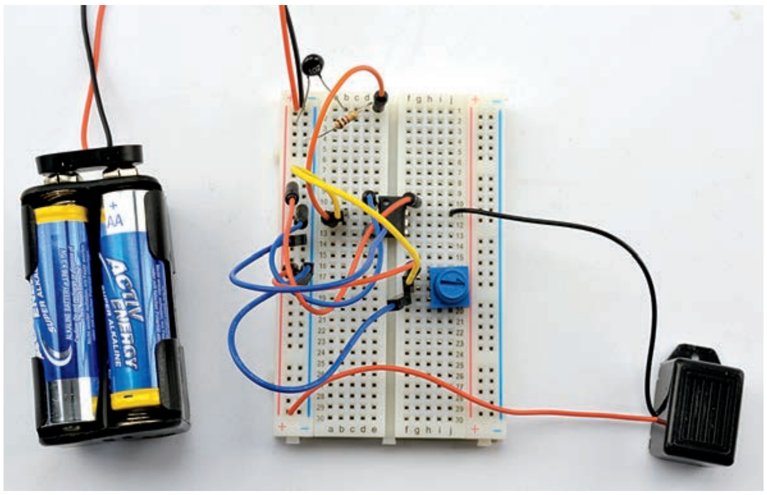
If you wanted to do more than simply display a message when it goes from light to dark or vice-versa, then you can add your own code after the print commands. For example, you could attach a relay module and turn on a light when it goes dark.

Switching with a Sensor and Comparator Chip

If you just want to detect when the output of a sensor crosses some threshold value, then you do not necessarily need an Arduino or Raspberry Pi. Sometimes you can just use a special chip called a comparator.

In this section, you will use a comparator IC (LM311) with a thermistor (temperature sensitive resistor). The output will drive a buzzer as an alarm when the temperature rises above the level set using a variable resistor. Figure 9-6 shows the project built on breadboard.

FIGURE 9-6 A temperature monitor



You Will Need

To experiment with temperature measurement and a comparator, you will need the following items.

Quantity	Item	Appendix Code
1	LM311 comparator IC	K1, S7
1	Thermistor (1kΩ)	K1, R4
1	1kΩ resistor	K1
1	10kΩ trimpot variable resistor	K1, R1
1	Solderless breadboard	K1, T5
7	Male-to-male jumper wires or solid core wire	K1, T6
1	6V DC buzzer	M22
1	4 x AA battery holder	K1, H1
4	AA batteries	
1	Battery clip	K1, H2

Construction

The schematic for the project is shown in Figure 9-7 and the breadboard layout in Figure 9-8.

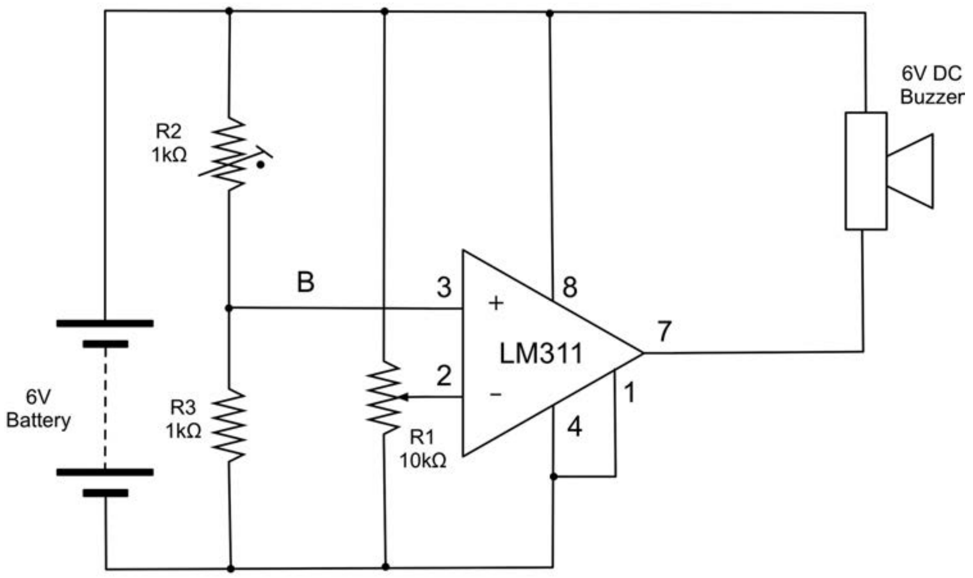


FIGURE 9-7 Schematic diagram for the temperature monitor

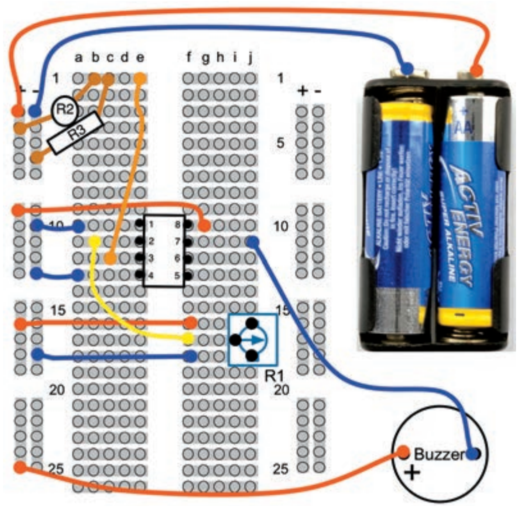


FIGURE 9-8 Breadboard layout for the temperature monitor

The LM311 comparator chip has two inputs (+ and –). If the + input is at a higher voltage than the – input, then the output transistor will be turned on. The + input is connected to a voltage divider formed of one fixed value 1kΩ resistor and a type of resistor called a thermistor, whose resistance is 1kΩ at 25°C but whose resistance decreases as the temperature increases. This means that the voltage at the + input to the comparator will increase as the temperature increases. The other input to the comparator (the – input) is connected to the slider of the variable resistor that allows the voltage to be set between 0 and 6V by turning the variable resistor's knob.

All this means is that by turning the knob you can set a "warning" temperature above which the buzzer will sound.

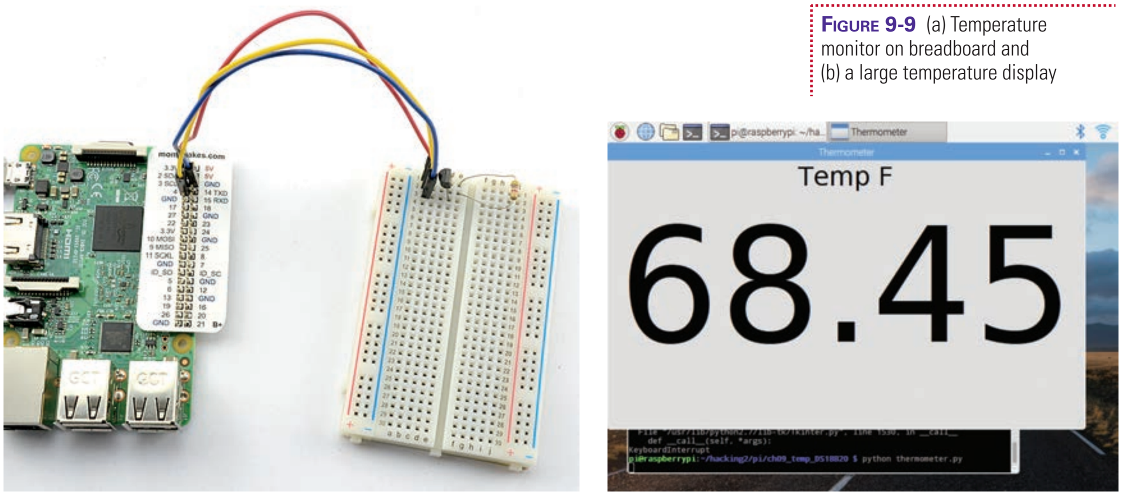
Build the circuit using the breadboard layout of Figure 9-8 and then adjust the variable resistor until the buzzer is just off. Then put your finger on the thermistor R2 to warm it up. Soon the buzzer will sound. Remove your finger and when the temperature has fallen back below the value set by the variable resistor, the buzzer will stop sounding.

The reason that we used the thermistor and fixed resistor instead of the TMP36 temperature sensor IC is that the TMP36 has a maximum supply voltage of 5.5V so the 6V supply of the battery could damage it. If you had a 3xAA battery box, then you could replace R2 and R3 with a TMP36.

Using a Digital Temperature Sensor

As you may have discovered, when you used the TMP36 with an Arduino, the TMP36 is not terribly accurate. In fact it's only guaranteed to be accurate to ±2° C. Also, having an analog voltage output means it's not possible to directly connect it to a Raspberry Pi.

An alternative temperature sensing IC (the DS18B20) is much more accurate (±0.5°C) and can be connected directly to a Raspberry Pi. To test out this chip, let's make a really large thermometer display using the DS18B20 and a Raspberry Pi 3 (Figure 9-9).



```
  File "/usr/lib/python2.7/lib-tk/Tkinter.py", line 1530, in __call__
    def __call__(self, *args):
KeyboardInterrupt
pi@raspberrypi:~/hacking2/pi/ch09_temp_DS18B20 $ python thermometer.py
```

(a) (b)

Figure 9-9 (a) Temperature monitor on breadboard and (b) a large temperature display

You Will Need

To experiment with temperature measurement and Raspberry Pi, you will need the following items.

Quantity	Item	Appendix Code
1	Raspberry Pi (any model)	M11
1	DS18B20 temperature sensor IC	S12
1	4.7kΩ resistor	K1
1	Solderless breadboard	K1, T5
3	Male-to-female jumper wires	K1, T12

Construction

Figure 9-10 shows the schematic diagram for the project and Figure 9-11 a close-up of the breadboard. The chip requires a 3.3V power supply with the middle output pin requiring a pull-up resistor connecting it to 3.3V.

Use male-to-female jumper wires to make the following connections:

- GND on the Raspberry Pi GPIO connector to row 3 of the breadboard.

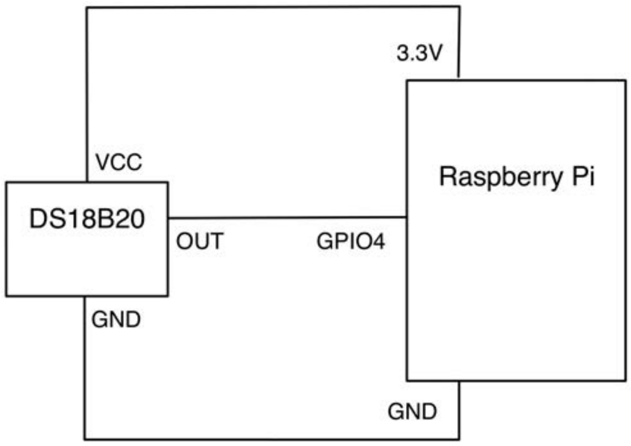


FIGURE 9-10 The schematic diagram for using a DS18B20 temperature sensor with Raspberry Pi

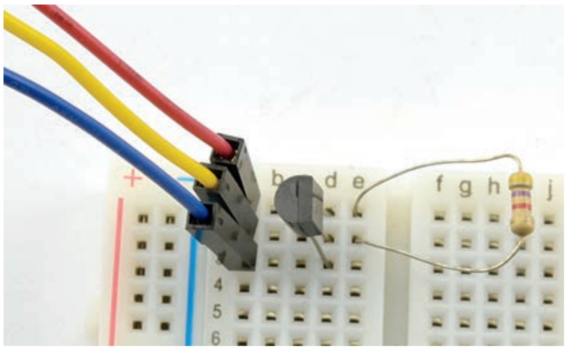


FIGURE 9-11 The breadboard layout for using a DS18B20 temperature sensor with Raspberry Pi

- 3.3V on the Raspberry Pi GPIO connector to row 1 of the breadboard.
- GPIO4 on the Raspberry Pi GPIO connector to row 2 of the breadboard.

Software

The program for this project is called “thermometer.py” and can be found in a folder of its own called “ch09_temp_DS18B20”.

```
from Tkinter import *
from DS18B20 import *
import time

class App:

  # this function gets called when the app is created
  def __init__(self, master):
    self.master = master
    # A frame holds the various GUI controls
    frame = Frame(master)
    frame.pack()
    label = Label(frame, text='Temp F',
            font=("Helvetica", 32))
    label.grid(row=0)
    self.reading_label = Label(frame, text='12.34',
                               font=("Helvetica",
                                      200))
    self.reading_label.grid(row=1)
    self.update_reading()

  # Update the temperature reading
  def update_reading(self):
    temp = read_temp_f()
    reading_str = "{:.2f}".format(temp)
    self.reading_label.configure(text=reading_str)
    self.master.after(500, self.update_reading)

# Set the GUI running, give the window a title, size
and position
root = Tk()
root.wm_title('Thermometer')
app = App(root)
root.geometry("800x450+0+0")
root.mainloop()
```

The code starts by importing the libraries Tkinter, DS18B20, and time. Tkinter is the user interface framework that allows you to create the great big text display of the temperature. The DS18B20 library is contained in the same directory as this program and handles the low-level interface to the temperature sensor IC. Feel free to use this library in your own projects.

Most of the rest of the program is concerned with creating the user interface on which to display the temperature. Much of this takes the form of a Python class, inside which the function __init__ creates a user interface frame as well as a label to display the text "Temp F" and a second reading_label that will display the temperature in a super-big font.

The function update_reading uses the DS18B20 library's read_temp_f function to get the temperature in degrees F and update the field. It also schedules itself to be called again after half a second.

The final section of code gives the window its title of "Thermometer" and opens it.

The Tkinter library is quite complex and powerful, so if you plan to design your own user interfaces, a good starting point is this tutorial: http://zetcode.com/gui/tkinter/. You will also find a chapter on Tkinter in my book *Programming the Raspberry Pi* (McGraw-Hill Education/TAB, 2015).

Arduino Egg and Spoon

Tiny accelerometer modules (Figure 9-12) are available at low cost. The two models shown are very similar, both being 5V compatible and providing analog outputs for each axis. The one on the left is from Freetronics (www.freetronics.com/am3x) and the one on the right is from Adafruit (www.adafruit.com/products/163).

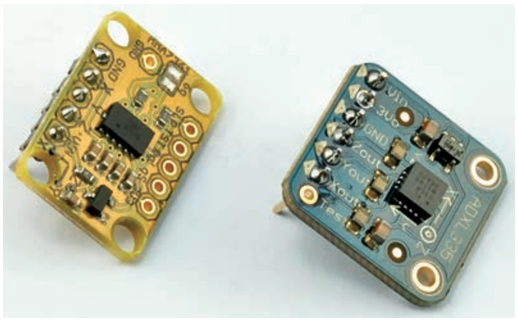


FIGURE 9-12 Low-cost accelerometer modules

These modules are three axis accelerometers that measure the force applied to a tiny weight inside the chip. Two of the dimensions, X and Y, are parallel to the modules PCB. The third dimension (Z) is at 90 degrees to the module's surface. There will normally be a constant force acting on this dimension due to gravity. So if you tip the module, the effect of gravity starts to increase on the dimension in which you tip it (see Figure 9-13).

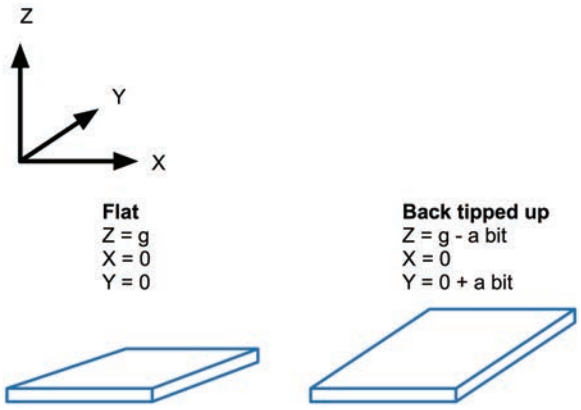


FIGURE 9-13 The effect of gravity on an accelerometer

As an example to test one of these accelerometers, we are going to build an electronic version of the children's game of egg and spoon. The idea behind this is to use the accelerometer to detect the level of tilt of the "spoon" and flash an LED when it starts to be in danger of losing its virtual egg. A buzzer sounds when the level of tilt is extreme enough for the egg to have fallen off (Figure 9-14).

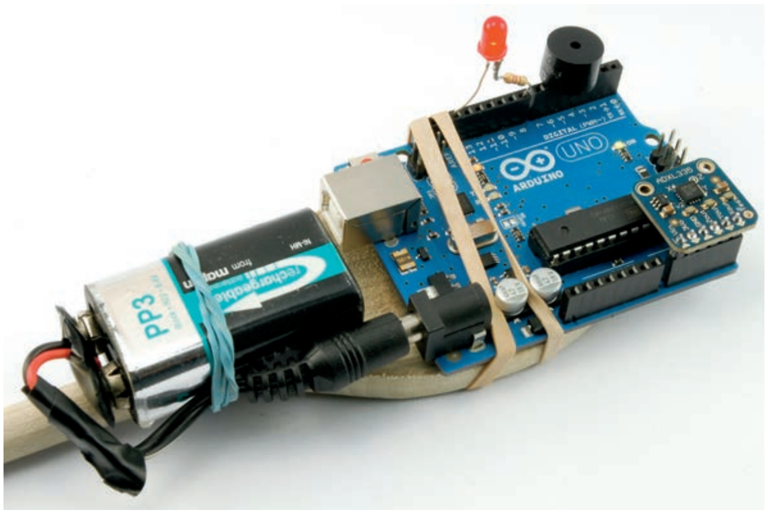


Figure 9-14 An Arduino and spoon race

You Will Need

To participate in an Arduino and spoon race, you will need the following items.

Quantity	Item	Appendix Code
1	Arduino Uno R3	M2
1	Accelerometer	M15
1	Piezo buzzer	K1, M3
1	LED	K1
1	270Ω resistor	K1
1	Battery clip to 2.1mm jack adapter	H9
1	Wooden spoon	The kitchen
1	PP3 9V battery	

Construction

With a bit of thought, both of the accelerometer modules are capable of being plugged directly into the Arduino, as are the buzzer and LED/resistor. You should program the Arduino with the right sketch for the accelerometer module you are using (two versions are provided) before you attach the module, just in case some of the pins on the A0 to A5 connector are set to be outputs from a previous sketch.

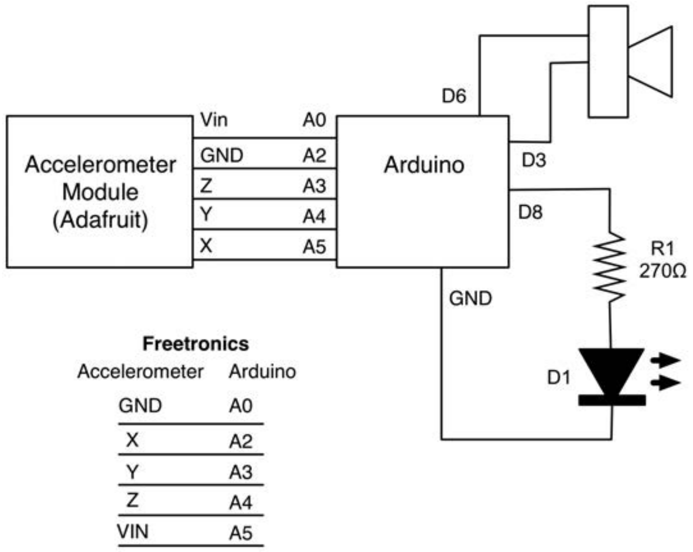


FIGURE 9-15 The schematic diagram for the egg and spoon

Figure 9-15 shows the schematic diagram for the Arduino Egg and Spoon. As you can see from Figure 9-14, all the components fit into the sockets on the Arduino. The LED/resistor combo is made up of the resistor soldered directly to the longer positive lead of the LED. The resistor end goes to digital pin 8 on the Arduino and the negative end of the LED to GND. The buzzer fits between pins D3 and D6—D6 being connected to the positive end of the buzzer. If the pins on your buzzer are at a different spacing, then you can pick other pins, but remember to change the variables "gndPin2" and "buzzerPin" to whatever pins you end up using.

Both of the accelerometer modules will fit in the Arduino sockets A0 to A5, as shown in Figure 9-14. However, their pin allocations are quite different.

The project is powered from a 9V battery using an adapter, and the Arduino and battery are attached to the spoon with elastic bands.

Software

There are two versions of the sketch provided: "ch09_egg_and_spoon_adafruit" and "ch09_egg_and_spoon_freetronics". Make sure you get the right one, and then program the Arduino with it BEFORE you attach the accelerometer. The only difference between the two sketches is the pin allocations.

This is the sketch for the Adafruit version. We start by defining the pins used. The two variables "levelX" and "levelY" are used to measure the resting values of acceleration for X and Y if the spoon is level.

```
int levelX = 0;
int levelY = 0;
```

The "ledThreshold" and "buzzerThreshold" can be adjusted to set the degree of wobble before the LED lights and the buzzer sounds to indicate a "dropped egg."

```
int ledThreshold = 10;
int buzzerThreshold = 40;
```

The "setup" function initializes the pins and then calls the function "calibrate" that sets the values of "levelX" and "levelY".

```
void setup() {
  pinMode(gndPin1, OUTPUT);
  digitalWrite(gndPin1, LOW);
  pinMode(gndPin2, OUTPUT);
  digitalWrite(gndPin2, LOW);
  pinMode(plusPin, OUTPUT);
  pinMode(ledPin, OUTPUT);
  pinMode(buzzerPin, OUTPUT);
  digitalWrite(plusPin, HIGH);
  pinMode(A1, INPUT); // 3V output
  calibrate();
}
```

In the "loop" function, we read the X and Y accelerations and see how much they have strayed from the values of "levelX" and "levelY". The "abs" function returns the absolute value of a number, so if the difference is negative, it is turned into a positive value, and it is this that is compared with the thresholds that have been set.

```
void loop() {
  int x = analogRead(xPin);
  int y = analogRead(yPin);
  boolean shakey = (abs(x - levelX) > ledThreshold ||
                    abs(y - levelY) > ledThreshold);
  digitalWrite(ledPin, shakey);
  boolean lost = (x > levelX + buzzerThreshold ||
                  y > levelY + buzzerThreshold);
  if (lost) {
    tone(buzzerPin, 400);
  }
}
```

The only complication in the "calibrate" function is that we must wait for 200 milliseconds before we can take the readings. This gives the accelerometer time to turn on properly.

```
void calibrate() {
  delay(200); // give accelerometer time to turn on
  levelX = analogRead(xPin);
  levelY = analogRead(yPin);
}
```

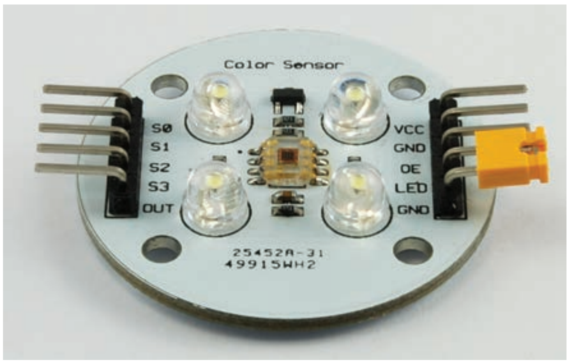
Measuring Something's Color

The TCS3200 is a handy little IC for measuring the color of something. There are several different variations on this chip, but they all work the same way. The chip has a transparent case, and dotting its surface are photodiodes with different color filters over them (red, green, and blue). You can read the relative amounts of each primary color to work out of the color of the object in front of the module.

The easiest way to use the chip is to buy a module like the one shown in Figure 9-16.

This module, which cost less than USD 10, also has four white LEDs that illuminate the object whose color you want to measure, as well as convenient header pins that make it easy to connect to an Arduino.

FIGURE 9-16 A light-sensing module



The IC does not produce an analog output, but instead varies the frequency of a train of pulses. To do this, it uses four selection pins, S0 to S3.

S0 and S1 select the frequency range of the pulses to be generated. These should both be set high. You choose which color the pulse frequency corresponds to by changing the values on the digital inputs S2 and S3 as follows:

- S2 low, S3 low—red
- S2 low, S3 high—blue
- S2 high, S3 low—white (no filter)
- S2 high, S3 high—green

You Will Need

To build this project, you will need the following items the following items.

Quantity	Item	Appendix Code
1	Arduino Uno R3	M2
1	TCS3200 color sensing module	M12
3	Male-to-female jumper wires	K1, T12

Construction

Construction is perhaps too strong a word for it. The module will fit directly into the Arduino (Figure 9-17), facing outward. It will make the following connections:

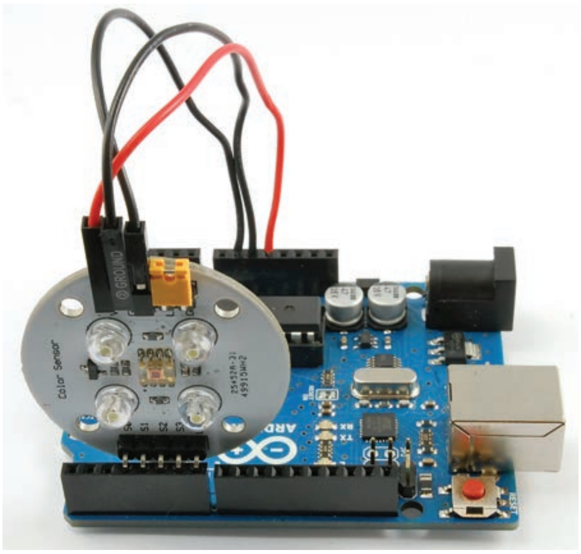
- S0 on the module to D3 Arduino
- S1 on the module to D4 Arduino
- S2 on the module to D5 Arduino
- S3 on the module to D6 Arduino
- OUT on the module to D7 Arduino

You will also need three male-to-female jumper leads to connect:

- VCC on the module to 5V Arduino
- GND on the module to GND Arduino
- OE (output enable) on the module to GND Arduino

Figure 9-18 shows the module sensing colors on a Rubik's cube.

Figure 9-17 A light sensor attached to an Arduino



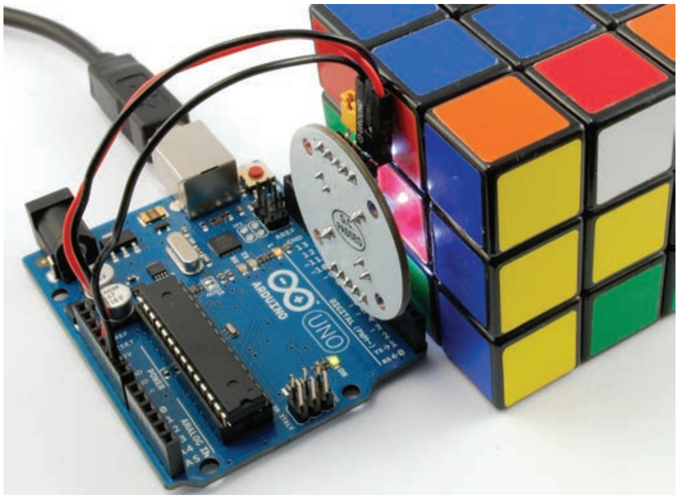


FIGURE 9-18 Sensing colors on a Rubik's cube

Software

The sketch "ch09_color_sensing" demonstrates the use of this module. When you upload the sketch and open the Serial Monitor, you will see something like this:

```
Hold white paper infront of sensor.
then press Reset
255      255    255
255      255    255
255      255    255
255      255    198
229      255    255
15       11     23
47       70     127
47       72     137
46       68     127
45       67     127
```

Hold a piece of white paper or card in front of the sensor and then press the Reset button on the Arduino. This starts a calibration function in the sketch that adjusts for the different sensitivities of the red, green, and blue sensors. After calibrating, the Serial monitor will update every half second or so with three numbers between 0 and 255 for the red, green and blue components of whatever is in front of the sensor. So, with the white paper in front of the sensor it will report roughly 255, 255, 255. But, if you replace the white paper with something of a different color, you should see the readings change.

In the example above, after the sixth reading a piece of blue card was placed infront of the sensor. You can see that the reading from the blue channel has increased to 127 and the first two channels (red and green) have decreased.

The sketch is quite long, so just the key parts will be highlighted.

Most of the work takes place in the readColor function:

```
int readColor(int bit0, int bit1) {
  digitalWrite(colorSelect0pin, bit0);
  digitalWrite(colorSelect1pin, bit1);
  long start = millis();
  for (int i=0; i< 1000; i++) {
    pulseIn(pulsePin, HIGH);
  }
  long period = millis() - start;
  int colorIntensity = int((maxPeriod
                      / float(period)) * 255.0);
  return colorIntensity;
}
```

This function takes two parameters that determine which color is to be sensed. The function then times how long it takes to register 1000 pulses. Since the longer this takes, the smaller that color reading, this needs to be inverted so that the number increases as the intensity of the color increases, so the constant maxPeriod is divided by the period taken for the 1000 pulses to give a value proportional to the frequency.

To get the value for a particular color there are three separate functions readRed, readGreen, and readBlue. They all follow the same pattern. The readRed function is listed below:

```
int readRed() {
  return min((readColor(LOW, LOW) * redFactor),
             255);
}
```

Each of these "read" functions uses readColor, with the appropriate color identification bits set, and then scales this by a factor (in this case redFactor). Finally the min function is used to make sure that the color value does not exceed 255.

The values for redFactor, greenFactor, and blueFactor are all initially set to 1.0. But, when the Arduino resets, the calibrate function is called to automatically adjust these values.

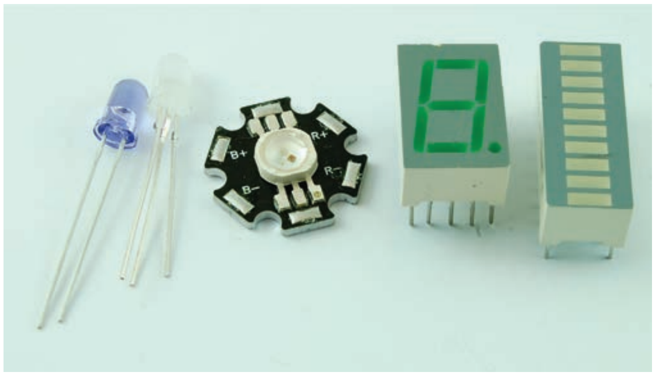
```
void calibrate() {
  Serial.println("Hold white paper in front of
                 sensor.");
  Serial.println("then press Reset");
  redFactor = 255.0 / readColor(LOW, LOW);
  greenFactor = 255.0 / readColor(HIGH, HIGH);
  blueFactor = 255.0 / readColor(LOW, HIGH);
}
```

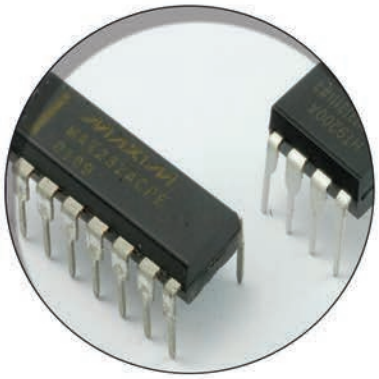
Summary

There are many other sensors out there, and many will interface to an Arduino or Raspberry Pi or even just a circuit using a comparator. Hopefully, these examples will help you to devise your own projects.

In the next chapter, we will change tack and look at sound and audio electronic hacks.

ARDUINO
UNO
DIGITAL (PWM~)


B+
R+
R-
B-



10

Audio Hacks

In this chapter, you will look at audio electronics and find out how to make and amplify sounds so you can drive a loudspeaker.

You will also discover how to hack an FM transmitter intended for use with MP3 players in the car, so that it works as a surveillance bug.

First though, we will look at the more mundane topic of audio leads, how to use them, mend them, and make your own.

Hacking Audio Leads

Ready-to-use audio leads are pretty cheap to buy unless you go for the high-end connectors. Sometimes though, if you need a lead in a hurry, or an unusual lead, it helps to know how to wire one up from parts in your junk box or from connectors you have bought.

Many items of consumer electronics are supplied with a range of leads, and you do not always need them for use with the item you bought. Keep them in your junk box since you never know when you might need to make some kind of lead.

Figure 10-1 shows a selection of audio plugs, some designed to have leads soldered to them, and others that have plastic moldings around the lead and cable, which cannot be soldered to. Plugs with plastic moldings around them are still useful, however. It just means you will have to cut and strip the wire that leads to the plug rather than solder it to the plug itself.

General Principals

Audio leads carry audio signals, often on their way to an amplifier, and the last thing you want is for them to pick up electrical noise that will affect the quality of the sound. For this reason, audio leads (before amplification) are normally screened (see Figure 10-2).

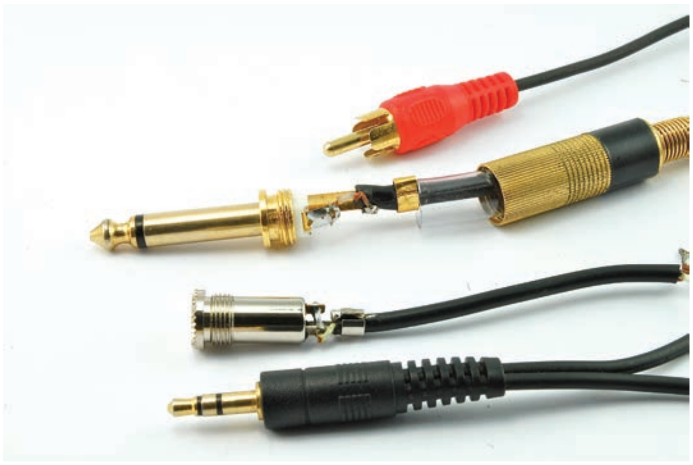
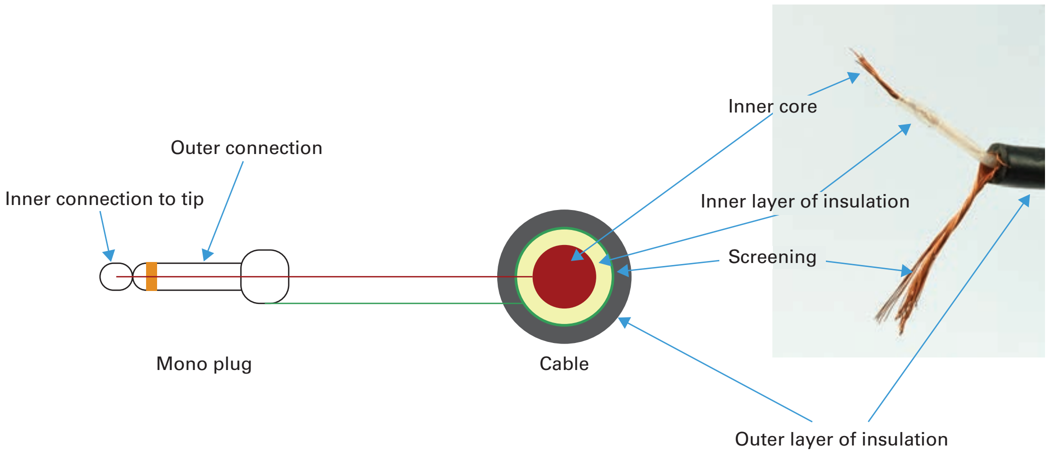


FIGURE 10-1 A variety of audio plugs

The audio signal itself (or two audio signals for stereo) is carried on insulated multi-core wires that are then enclosed in an outer conductive sheath of screening wire that carries the ground connection.

FIGURE 10-2 A screened audio lead



The exception to this is for leads to loudspeakers. These are not screened because the signal has been amplified to such a degree that any noise the speaker cables might pick up would be undetectable.

Soldering Audio Connectors

Stripping audio connectors is made more difficult by the fact that there is more than one layer of insulation. It is very easy to accidentally cut through the shielding. Nicking the outer insulation all around before stripping it will usually help with this problem.

Figure 10-3 shows the sequence involved in soldering a screened lead to a 6.3mm jack plug of the sort often used to connect an electric guitar to its amplifier.

The first step is to strip off the outer insulation about 20mm (a bit less than an inch) from the end of the lead and tease the shielding wires around to one side of the lead and twist them together. Strip about 5mm of insulation off the inner core insulation (Figure 10-3a). Then, tin both bare ends (Figure 10-3b).

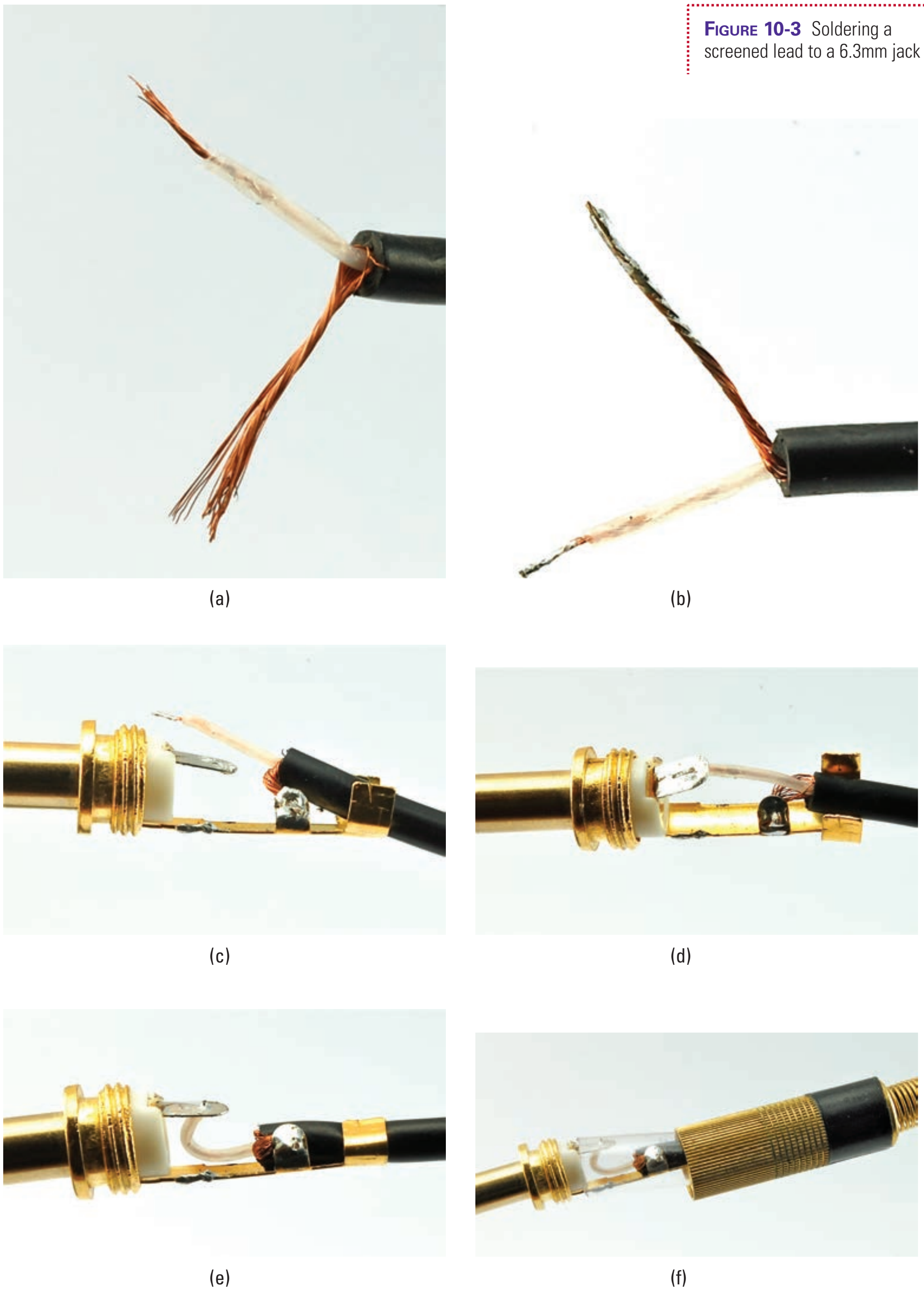
The jack plug has two solder tags: one for the outer part of the plug and one connected to the tip. Both will usually have holes in them. Figure 10-3c shows the screening trimmed to a shorter length and pushed through the hole ready to solder. Once the screening is soldered into place, solder the inner core to the solder tag for the tip (Figure 10-3d).

These wires are quite delicate, so make sure the inner core wire has some extra length (as shown in Figure 10-3e) so that if the plug flexes, it will not break the connection. Notice that the strain relief tabs at the end of the plug have been pinched around the outer insulation. Finally, the plug will often have a plastic sleeve that protects the connections. Slide this over the connections and then screw in the plug casing.

Tip

If there is a plug on the other end of the lead, remember to push the new plug enclosure and plastic sleeve onto the lead BEFORE you solder the second plug on; otherwise, you will end up having to unsolder everything to put it on. The author has made this mistake more times than he cares to admit.

Figure 10-3 Soldering a screened lead to a 6.3mm jack plug



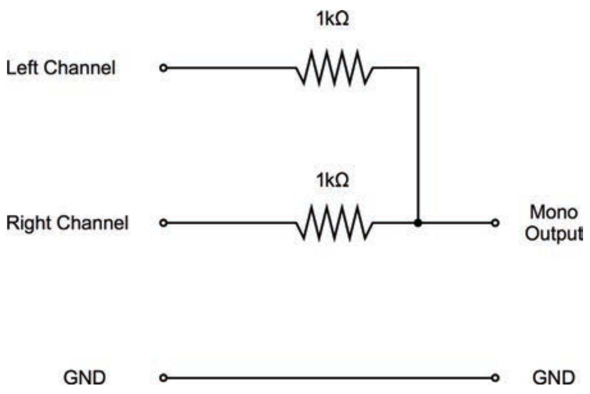


FIGURE 10-4 Mixing stereo to mono

Converting a Stereo Signal to Mono

Stereo audio is made up of two slightly different audio signals that give the stereo effect when played through two separate speakers. Sometimes, you have a stereo output that you want to input to a single channel (mono) amplifier.

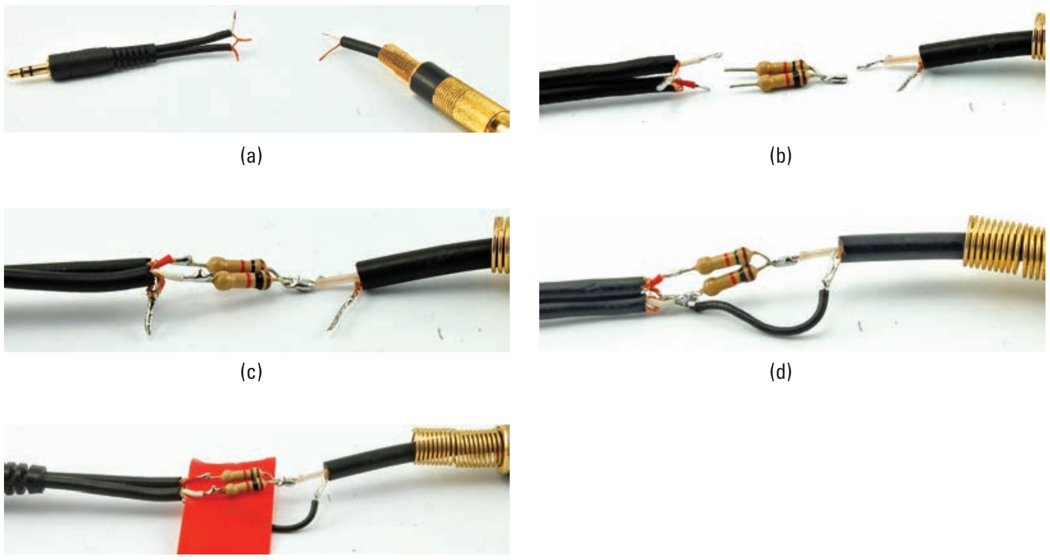
You could use just one of the channels of the stereo signal (say, the left channel), but then you will lose whatever is on the right channel. So a better way of converting stereo to mono is to use a pair of resistors to mix the two channels into one (Figure 10-4).

Looking at the schematic of Figure 10-4, you could be forgiven for thinking all you need to do is connect the left and right channels to each other directly. This is not a good idea, because if the signals are very different, there is the potential for a damaging current to flow from one to the other.

As an example, we could use the mono 6.3mm jack we just soldered leads to, and combine it with a pair of resistors and a stereo 3.5mm jack plug so we could, for example, plug an MP3 player into a guitar practice amplifier.

Figure 10-5 shows the steps involved in this. To make it easier to photograph, the author's lead is made very short. You

FIGURE 10-5 Making a lead



will probably want to make yours longer. This is not a problem, unless you plan to make it longer than a few yards or meters.

The 3.5mm plug is of the plastic molded variety, reclaimed from some unwanted lead. The first step is to strip both leads (Figure 10-5a). Note that the stereo plug has two screened connections in one twin cable. The screened ground connections of both channels of the stereo plug can be twisted together.

Tin the ends of all the leads, and then solder the resistors together, as shown in Figure 10-5b.

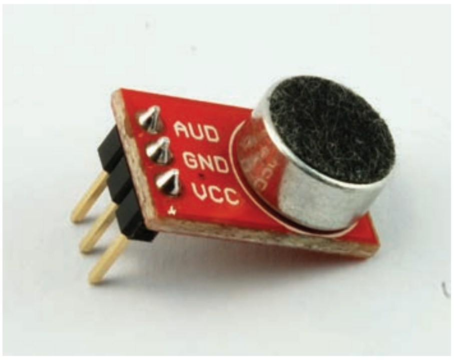
Next, solder the stereo and mono leads to the resistors, as shown in Figure 10-5c, and cut and tin a short length of wire to bridge the ground connections. Solder it into place (Figure 10-5d) and then wrap everything in insulating tape, taking care to put tape in between any places where wires could short together (Figure 10-5e).

Using a Microphone Module

Microphones (mics) respond to sound waves, but sound waves are just small changes in air pressure, so it is not surprising that the signal you get from a mic is usually very faint. It requires amplification to bring it up to a usable level.

While it is perfectly possible to make a little amplifier to boost the signal from your mic, you can also buy a mic module that has an amplifier built in. Figure 10-6 shows such a module.

FIGURE 10-6 A microphone module



The mic module just requires a supply voltage between 2.7V and 5.5V. This makes it ideal for interfacing with an Arduino.

In Chapter 11, you will find out a bit more about oscilloscopes. But for now, here is a sneak preview of what an oscilloscope will display (see Figure 10-7) when connected to the mic module while a constant tone is being generated.

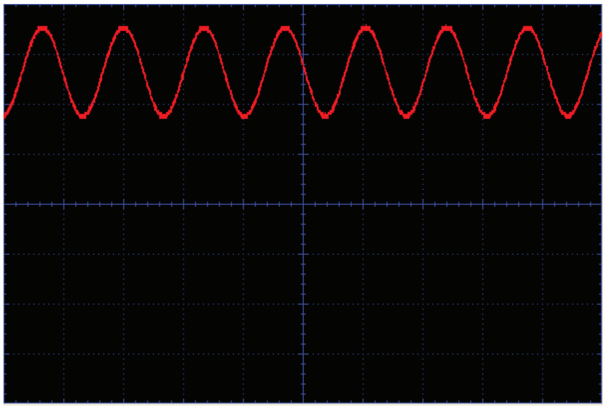
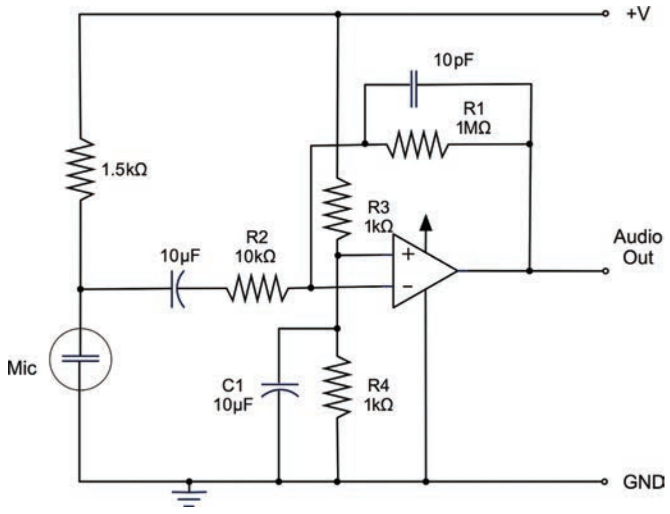


FIGURE 10-7 The output of the microphone module

The oscilloscope is displaying the sound. In this case, a constant and rather irritating tone of 7.4 kHz. The horizontal axis is time, and each blue square represents 100 microseconds. The vertical axis is the voltage and each square is 1V. The output of the mic module is a voltage that varies very quickly between about 1.8V and 3.5V. If you draw a horizontal line straight down the middle of the waveform, it would be at about 2.5V. This is halfway between 0V and 5V. So if there is no sound at all, there will just be a flat line at 2.5V, and as the sound gets louder, the waveform will swing further and further either way. It will not, however, go higher than 5V or lower than 0V. Instead, the signal will clip and become distorted.

FIGURE 10-8 The schematic diagram for a mic module



The mic module shown is sold by SparkFun (BOB-09964). The schematic for this, along with all its design files have been made public. Figure 10-8 shows a schematic for a typical microphone pre-amp.

The chip at the center of this design has a similar circuit symbol to the comparator you used in the “Threshold Sensing with Raspberry Pi” section at the beginning of Chapter 9. However, it is not a comparator; it is an amplifier IC of a type known as an “operational amplifier” (or “op amp” for short).

Whereas a comparator turns its output on when the “+” input is higher than the “–” input, an op amp amplifies the difference between the “+” and “–” inputs. Left to its own devices, it amplifies this by a factor of millions. This means that the tiniest signal or noise on the input would be turned into meaningless thrashing of the output from 0 to 5V. To tame the op amp and reduce its amplification factor (called “gain”), something called “feedback” is used.

The trick is to take a portion of the output and feed it into the negative input of the op amp. This reduces the gain to an amount determined by the ratio of R1 to R2, shown in Figure 10-8. In this case, R1 is 1MΩ and R2 is 10kΩ, so the gain is 1,000,000/10,000 or 100.

The signal from the microphone is being amplified by a factor of 100. This shows just how weak the signal is in the first place.

The "+" input to the op amp is held halfway between GND and 5V (2.5V) by using R3 and R4 as a voltage divider. C1 helps to keep this constant.

From the schematic, you can see how you could build the module yourself on, say, stripboard. Op amps like the one used (which is a surface-mounted device) are also available in the eight-pin DIP form. However, a module like this will save you a lot of effort and may even turn out cheaper than buying and building a module from scratch.

I realize this is a rather cursory introduction to op amps. These are very useful devices, but unfortunately require more space to explain fully than this book can accommodate. You will find good information on op amps at the Wikipedia site, as well as in books with a more theoretical bent like *Practical Electronics for Inventors*, by Paul Sherz and Simon Monk, which has a chapter devoted exclusively to op amps.

In the next section, we will combine this module with a hacked FM transmitter of the sort used to let you play your MP3 player through your car radio, thus creating an audio "bug."

Making an FM Bug

To make an FM transmitter that will broadcast sound picked up from a microphone to a nearby FM radio receiver would require a lot of effort. We are hackers, so we are going to cheat and take apart an FM transmitter and wire it up to a mic module. Figure 10-9 shows the end result of this hack.

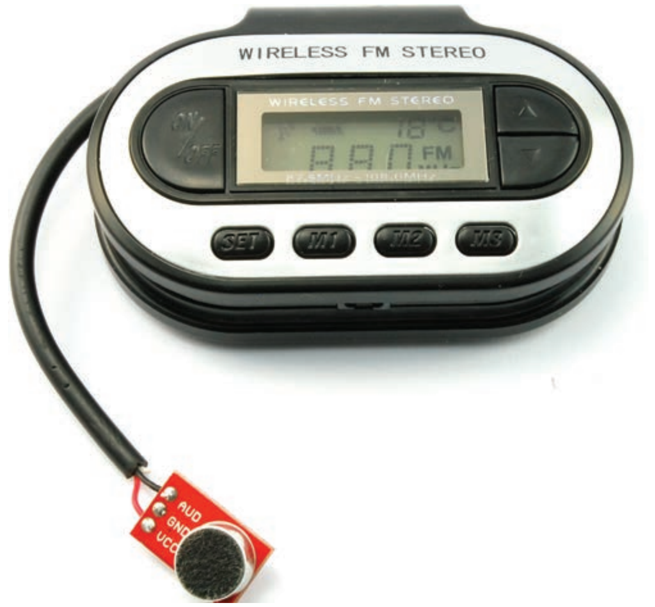


Figure 10-9 An FM radio bug

You Will Need

To build the bug, you will need the following items.

Quantity	Item	Appendix Code
1	Microphone module	M5
1	FM transmitter for MP3 players*	
1	FM radio receiver	

* For suitable FM transmitters, try searching on eBay using the search terms "fm transmitter mp3 car." Expect to pay about USD 5 and look for the most basic of models. You do not need remote control or an SD card interface. You just want something that has an audio input lead, and for simplicity purposes runs on two AA or AAA batteries (3V).

Construction

This is a very easy project to make. Figure 10-10 shows the schematic diagram for the bug.

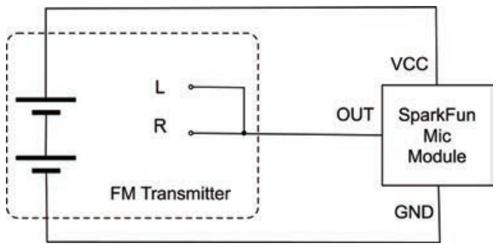


FIGURE 10-10 Schematic diagram for the radio bug

The 3V battery of the FM transmitter is used to provide power to the mic module, and the single output of the mic module is connected to both the left and right inputs of the stereo FM transmitter.

Figure 10-11 shows how the FM transmitter is modified to connect the mic module to it.

The first step is to unscrew any screws that hold the case together and pull it apart. Then, chop off the plug, leaving most of the lead in place since the lead often doubles as an antenna in these devices. Strip and tin the three wires inside the lead (Figure 10-11a).

Looking at the three wires, in Figure 10-11a, the red wire is the right signal, the white the left, and the black ground. This is a common convention, but if you are not sure it applies to your transmitter, you can check by stripping the wires on the plug end of the lead you cut off and using the continuity setting on your multimeter to see which lead is connected to what on the plug. The farthest tip and next ring should be the left and right signals, and the metal nearest the plastic should be the ground connection.

We are going to leave the ground and left connections as they are, but disconnect the wire and connect it to the 3V connection of the battery (Figure 10-11b). In this transmitter, the positive terminal of the battery box underneath the PCB is soldered to the top surface of the PCB.

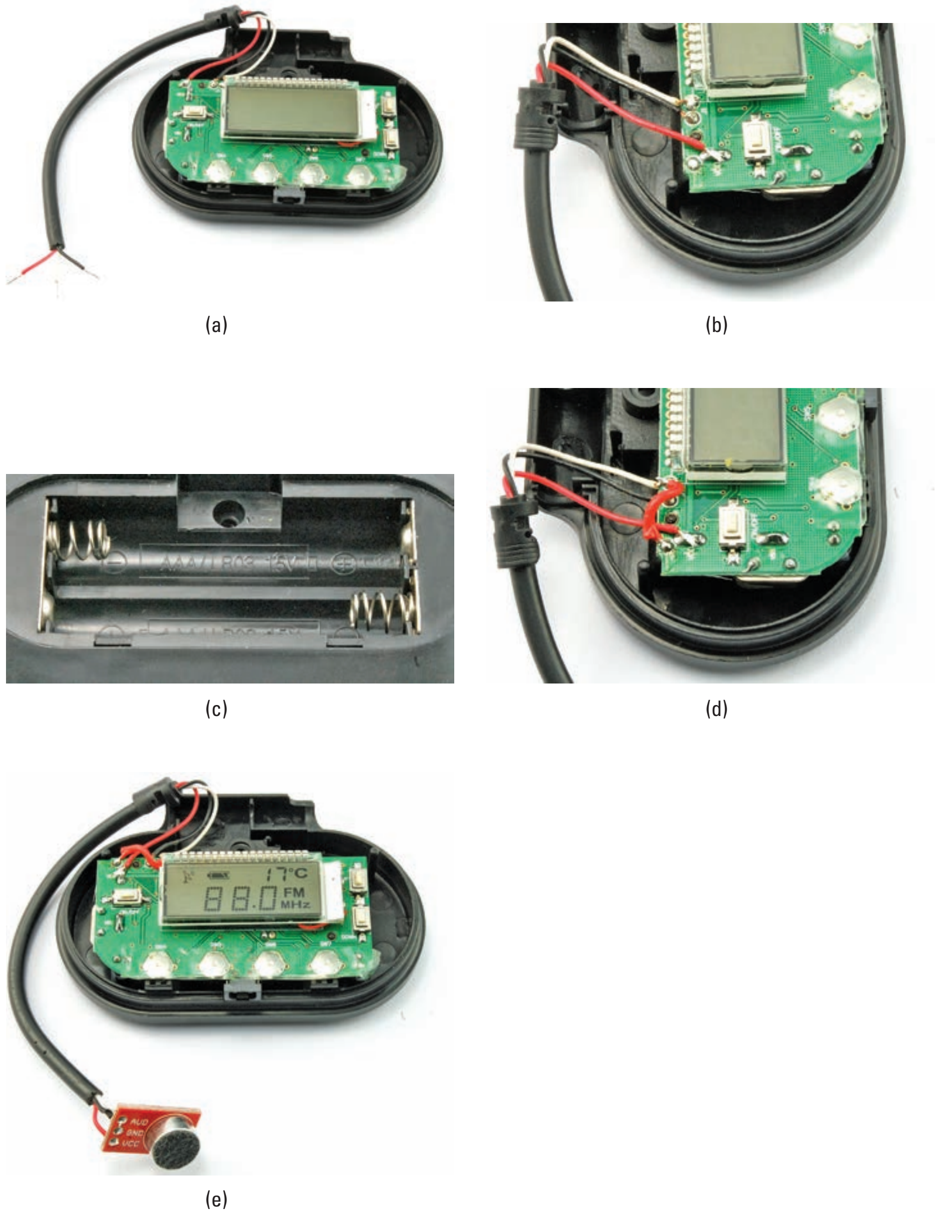


FIGURE 10-11 Modding the FM transmitter

To find the positive connection, look carefully at the battery box. In Figure 10-11c, you can see that the metal piece on the left of the figure links the negative of the top cell to the positive of the bottom cell. The 3V connection will therefore be the top-right connection of the battery box, so trace where this comes out on the top of the PCB. If it is attached by wires, then find an appropriate place for the red wire of the audio jack lead to be joined to it.

Referring back to the schematic diagram of Figure 10-10, we need to make a little wire just to link the left and right channels (Figure 10-11d). When all the changes are complete, it should look like Figure 10-11e.

Testing

Note that the on/off button of the transmitter will have no effect on the power going to the mic module. So to fully turn off the bug, remove the batteries.

To test the module out, set the frequency of the FM transmitter to one not occupied by a radio station and then set the radio receiver to the same frequency. You may well hear the howl of feedback through the radio. To prevent that, take the radio receiver to a different room. You should find that you can hear what is happening in the room with the bug in it pretty clearly.

Selecting Loudspeakers

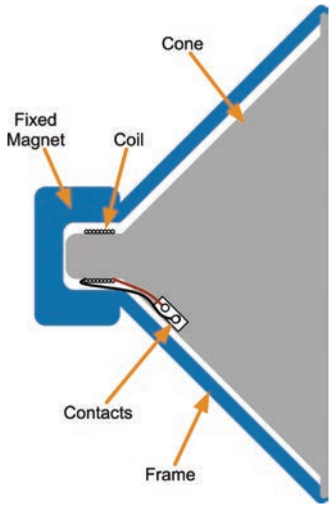


Figure 10-12 How a loudspeaker works

Loudspeakers have remained largely unchanged in design since the early days of radio. Figure 10-12 shows how a loudspeaker works.

The cone (often still made of paper) has a light coil around the end that sits within a fixed magnet attached to the frame of the loudspeaker. When the coil is driven by an amplified audio signal, it moves toward and away from the magnet in time with the audio. This creates pressure waves in the air, producing a sound.

Electronically speaking, a loudspeaker just looks like a coil. When you buy a speaker like this, it will have a number of ohms associated with it. Most speakers are 8Ω, but you can also commonly find 4Ω and 60Ω speakers. If you measure the resistance of the coil of an 8Ω speaker, you should find that it is indeed about 8Ω.

Another figure that is normally stated with the speaker is the power. This specifies how hard the loudspeaker can be driven before the coil will get too hot and burn out. For a small loudspeaker such as one you might put in a small radio receiver, values of 250 mW and up are not untypical. As you progress toward the kind of speakers you would use with a hi-fi set, you will see figures in the tens of watts, or even hundreds of watts.

It is very hard to build speakers that can cover the whole range of audio frequencies, which is generally standardized as 20 Hz up to 20 kHz. So you will often find hi-fi speakers that group a number of speakers into a single box. This might be a "woofer" (for low frequencies) and a "tweeter" (for high frequencies). Because woofers cannot keep up with the high frequencies, a module called a "crossover network" is used to separate the low and high frequencies and drive the two types of speakers separately. Sometimes this is taken a step further and three drive units are used: one for bass, one for mid-range tones, and a tweeter for high frequencies.

The human ear can pick out the direction of a high-frequency sound very easily. If you hear a bird tweeting in a tree, you will probably be able to look straight at it without having to think about where it is. The same is not true of low frequencies. For this reason, surround-sound systems often have a single low-frequency "woofer" and a number of other speakers that handle midrange and higher frequencies. This makes life easier, because bass speakers have to be much larger than higher-frequency units in order to push large amounts of air about relatively slowly to produce bass sounds.

Making a 1-Watt Audio Amplifier

Building a small amplifier to drive a loudspeaker is made easier by an IC like the TDA7052, which contains pretty much all the components you need, on a chip costing less than $1. In this section, you will make a little amplifier module on stripboard (Figure 10-13).

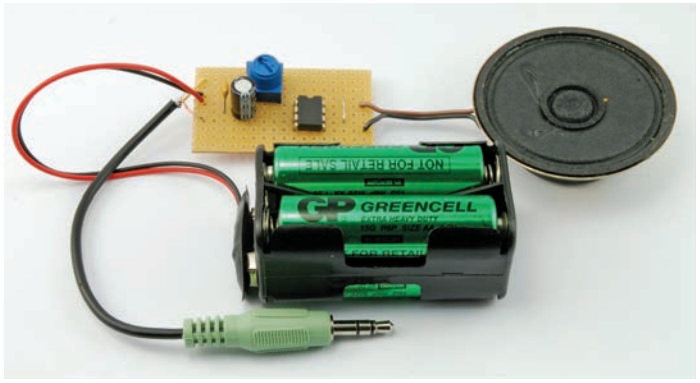


Figure 10-13 A 1-watt amplifier module

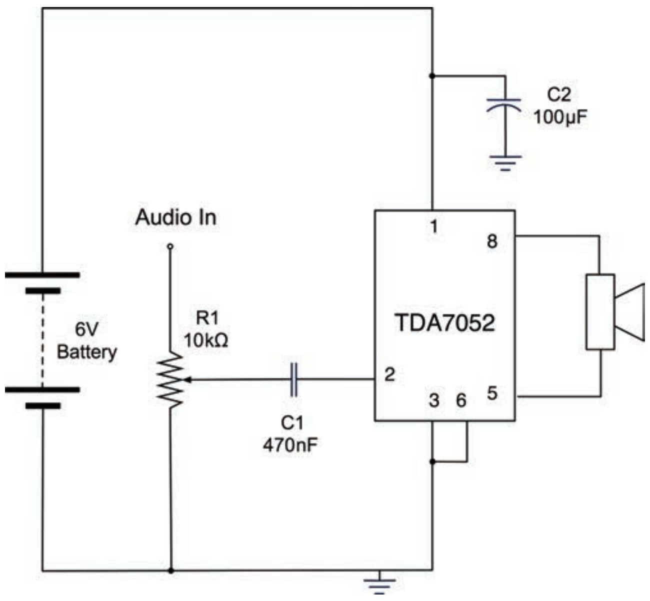


Figure 10-14 A typical TDA7052 amplifier schematic

An alternative to making your own amplifier is to buy a ready-made module. You will find these available for a wide range of different powers and in mono and stereo configurations. eBay is a good source for such modules, as are SparkFun (BOB-11044) and Adafruit (product ID 987). These modules often use an advanced type of design called "class-D," which is far more efficient in its use of energy than the module we are going to build.

Figure 10-14 shows the typical schematic for a TDA7052 amplifier.

R1 acts as a volume control, reducing the signal before amplification.

C1 is used to pass the audio signal on to the input to the amplifier IC without passing on any bias voltage that the signal may have from the audio device producing the signal. For this reason, when you use a capacitor like this, it is called a coupling capacitor.

C2 is used to provide a reservoir of charge that can be drawn on quickly by the amplifier when it needs it for very rapid changes in the power supplied to the speaker. This capacitor should be positioned close to the IC.

You Will Need

To build the amplifier module, you will need the following items.

Quantity	Name	Item	Appendix Code
1	IC1	TDA7052	S9
1	R1	10kΩ variable resistor	K1, R1
1	C1	470nF capacitor	K1, C3
1	C2	100µF capacitor	K1, C2
1		8Ω speaker	H14
1		Stripboard	H3

Construction

Figure 10-15 shows the stripboard layout for the amplifier module. If you have not used stripboard before, read through the section titled "Using Stripboard (LED Flasher)" in Chapter 4.

To build the module, follow the steps shown in Figure 10-16.

First, cut the stripboard to size and make the three cuts in the tracks using a drill bit (Figure 10-16a).

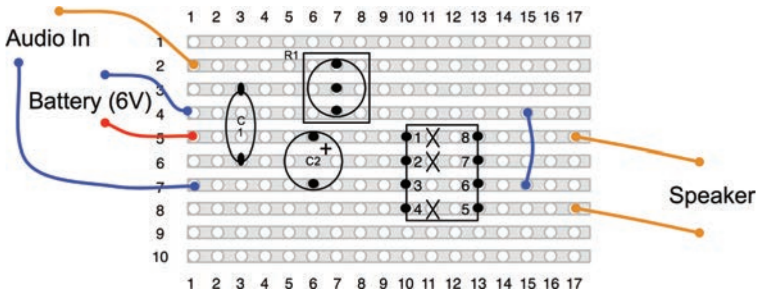


FIGURE 10-15 The stripboard layout for an amplifier module

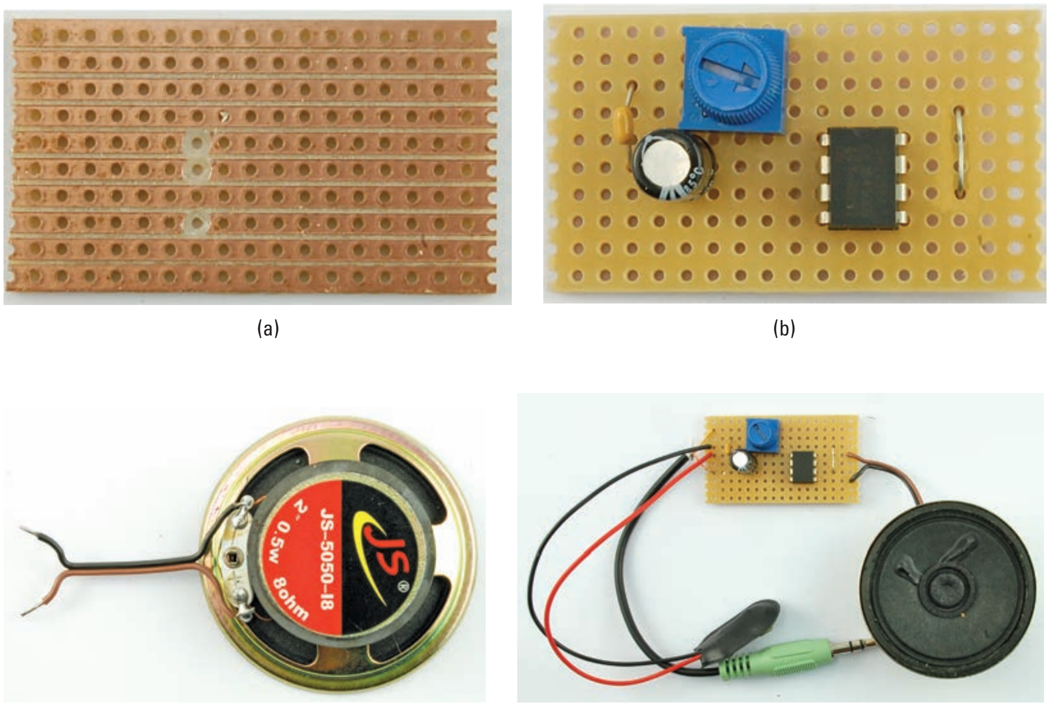


FIGURE 10-16 Building the audio amplifier module

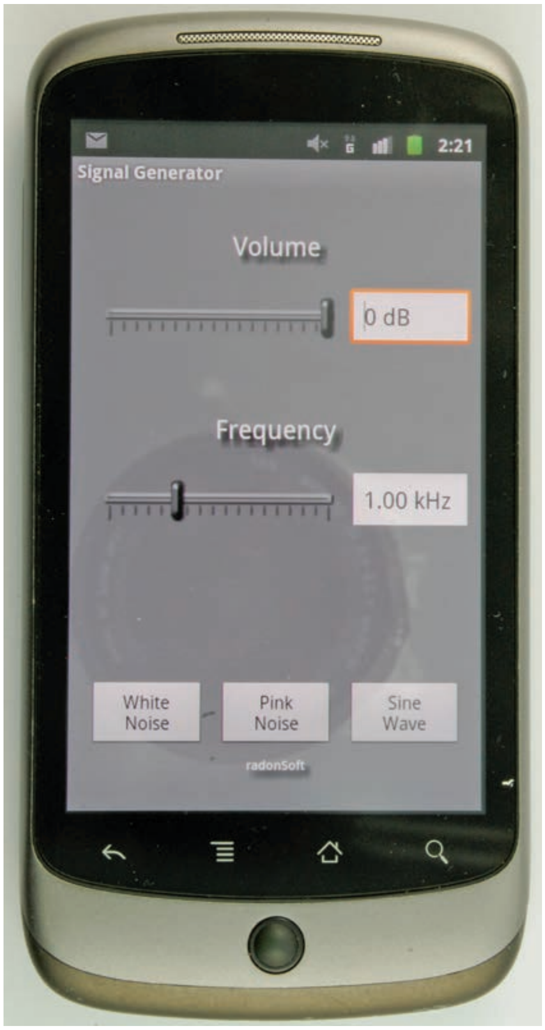


FIGURE 10-17 A signal generator app

The next step is to solder the link into place, and then the IC, C1, C2, and R1 in that order (Figure 10-16b). It is easiest to solder the components that are lowest to the board first.

Attach leads to the speaker (Figure 10-16c) and finally attach the battery clip and a lead ending in a 3.5mm stereo jack plug (Figure 10-16d). Note that only one channel of the audio lead is used. If you want to use both left and right channels, you should use a pair of resistors (see the section "Hacking Audio Leads" at the beginning of this chapter).

Testing

You can try the amplifier out by plugging it into an MP3 player, or, if you have an Android phone or an iPhone, download a signal generator app like the one shown in Figure 10-17. There are a number of such apps, many of them free, including this one for Android from RadonSoft.

With this, you can play a tone at a frequency you select. By noting when the volume of the speaker starts to drop off, you can work out the useful frequency range of your amplifier module.

Generating Tones with a 555 Timer

Back in Chapter 4, you used a 555 timer to blink a pair of LEDs. In this section, we will see how to use a 555 timer IC oscillating at much higher frequencies to generate audio tones.

The pitch will be controlled using a photoresistor so that as you wave your hand over the light sensor, the pitch will change in a theremin-like manner.

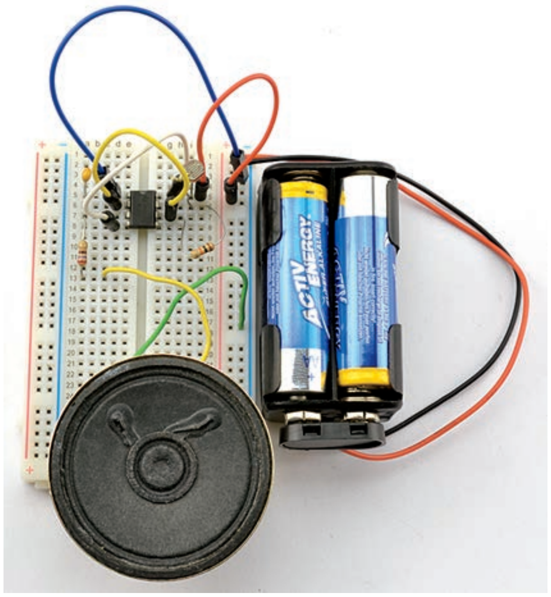


FIGURE 10-18 Generating tones with a 555 timer IC

Figure 10-18 shows the tone generator built onto breadboard.

Figure 10-19 shows the schematic diagram for the tone generator.

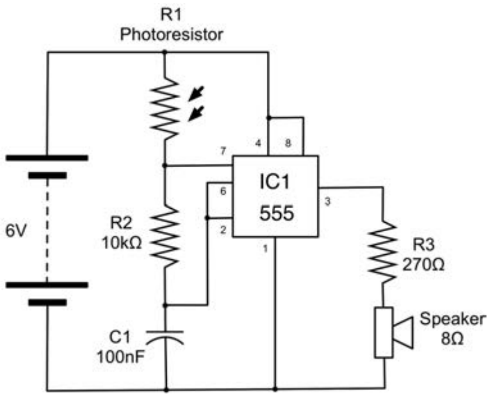


FIGURE 10-19 Schematic diagram for a 555 tone generator

This is similar to the design of the LED flasher in Chapter 4. In this case, instead of two fixed resistors and a capacitor setting the frequency, R1 is the LDR, whose resistance will vary between about 1kΩ and 4kΩ depending on the light falling on it. We need a much higher frequency than our LED flashing circuit—in fact, if we aim for a maximum frequency of around 1 kHz, we need a frequency of about 1000 times what we had before.

The 555 timer oscillates at a frequency determined by the formula:

frequency = 1.44 / ((R1 + 2 * R2) * C)

where the units of R1, R2, and C1 are in Ω and F.

So, if we use a 100nF capacitor for C1, and R2 is 10kΩ, and R1 (the LDR) has a minimum frequency of 1kΩ, then we can expect a frequency of:

1.44 / ((1000 + 20000) * 0.0000001) = 686 Hz

If the LDR's resistance increases to 4kΩ, then the frequency will drop to:

1.44 / ((4000 + 20000) * 0.0000001) = 320 Hz

To calculate the frequency and when deciding what values of R1, R2, and C1 to use, there are online calculators like this one at www.bowdenshobbycircuits.info/555.htm that will calculate the frequency for you.

You Will Need

To build the amplifier module, you will need the following items.

Quantity	Name	Item	Appendix Code
1	IC1	555 timer IC	K1, S10
1	R1	Photoresistor	K1, R2
1	R2	10kΩ resistor	K1
1	R3	270Ω resistor	K1, C4
1	C1	100nF capacitor	K1, C5
1		8Ω speaker	H14

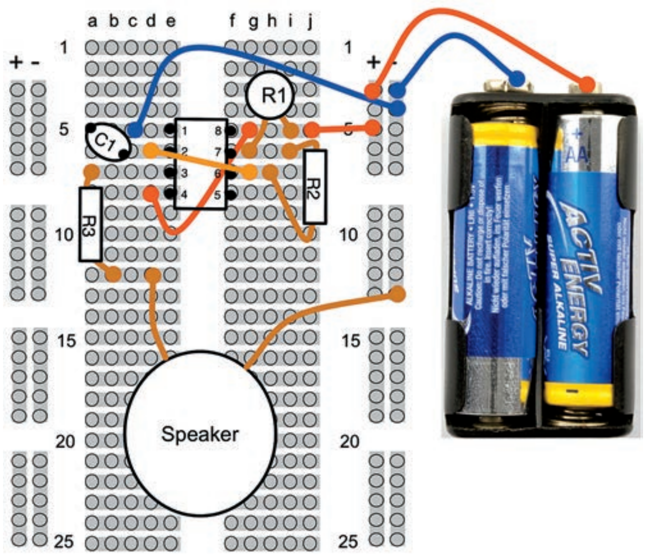


FIGURE 10-20 A signal generator app

Construction

Figure 10-20 shows the breadboard layout for the tone generator.

It would be quite straightforward to build this design onto stripboard. The stripboard layout in the section "Using Stripboard (LED Flasher)" in Chapter 4 would be a good starting point.

Making a USB Music Controller

Music software like Ableton Live™ is designed to allow USB controllers that emulate a keyboard to control virtual musical instruments and do all kinds of exciting things. You can use the USB keyboard emulation features of the Arduino Leonardo with an accelerometer so that tilting the board produces a key press of a number between 0 and 8, with 4 being pressed if the board is level, 0 if tilted almost vertically to the right, and 8 being pressed when it is tilted the other way.

The only hardware on the Arduino is the accelerometer (shown in the bottom right of Figure 10-21).

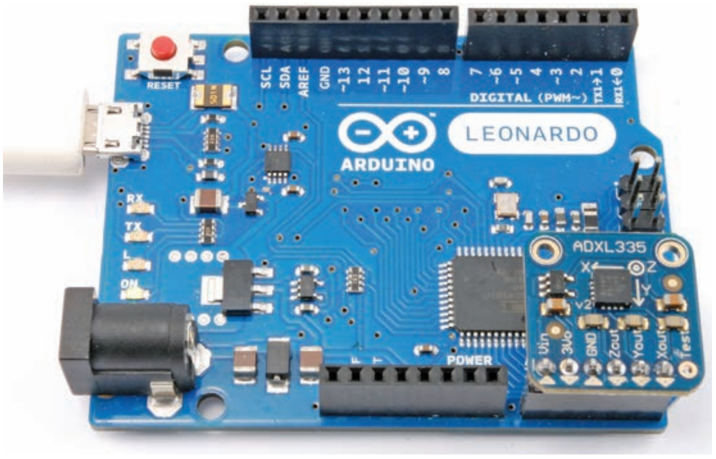


Figure 10-21 A USB music controller

You Will Need

To build this controller, you will need the following items.

Quantity	Item	Appendix Code
1	Arduino Leonardo	M21
1	Accelerometer	M15 (Adafruit version)

Construction

There is actually very little to construct in this project. The schematic is actually the same as in the section titled "Arduino Egg and Spoon" in Chapter 9. The Freetronics accelerator will also work, but you will need to change the pin assignments before attaching the accelerometer.

Software

The software for the music controller combines code for sensing the angle of tilt on the X-axis with emulating a keyboard press.

The first step is to assign the pins to be used. The accelerometer module is powered from output pins.

```
// music_controller

int gndPin = A2;
int xPin = 5;
int yPin = 4;
int zPin = 3;
int plusPin = A0;
```

The variable “levelX” is used during calibration and holds the analog value when the accelerometer is flat.

The “oldTilt” variable contains the old value of the tilt of the board, which is a value between 0 and 8, where 4 means level. The old value is remembered, so that a key press is only sent if the tilt angle changes.

```
int levelX = 0;
int oldTilt = 4;
```

The “setup” function sets the output pins to power the accelerometer, calls “calibrate”, and starts the Leonardo keyboard emulation mode.

```
void setup()
{
  pinMode(gndPin, OUTPUT);
  digitalWrite(gndPin, LOW);

  pinMode(plusPin, OUTPUT);
  digitalWrite(plusPin, HIGH);
  calibrate();
  Keyboard.begin();
}
```

In the main loop, the accelerometer reading is converted to a number between 0 and 8, and if it has changed since the last reading, a key press is generated.

```
void loop()
{
  int x = analogRead(xPin);
  // levelX-70 levelX levelX + 70
  int tilt = (x - levelX) / 14 + 4;
  if (tilt < 0) tilt = 0;
  if (tilt > 8) tilt = 8;
  // 0 left, 4 is level, 8 is right
  if (tilt != oldTilt)
  {
      Keyboard.print(tilt);
      oldTilt = tilt;
  }
}
```

The “calibrate” function takes an initial reading of the acceleration on the X-axis, after waiting 200 milliseconds for the accelerometer to turn on properly.

```
void calibrate()
{
  delay(200); // give accelerometer time to turn on
  levelX = analogRead(xPin);
}
```

Summary

In addition to the how-tos just covered, there are lots of audio modules you can make use of. Low-cost stereo power amplifiers are available from eBay and suppliers like SparkFun and Adafruit.

You can also buy ultra-low-cost amplified speakers intended for computers and reuse them in your projects.

11

Mending and Breaking Electronics

In this chapter, we will look at taking things apart and putting them back together again, or just taking them apart to salvage components.

In today's throw-away society, many consumer electronics items that stop working go directly into the garbage. Economically, they are simply not worth paying someone to repair. However, that does not mean it is not worth *trying* to repair them. Even if the attempt fails, some serviceable components may be scavenged for use in your projects.

Avoiding Electrocution

When working on something that is powered by household electricity, NEVER work on it when it is plugged into the outlet. I actually like to have the electrical plug for the appliance right in front of me, so that I know it is not plugged in. Household electricity kills many people every year. Take it seriously!

Some devices, such as switch mode power supplies, contain high-value capacitors that will hold their charge for hours after the device has been unplugged. These capacitors are simply biding their time, waiting for some unsuspecting fingers to complete the circuit.

Unless it is a very small capacitor, it should not be discharged by shorting the leads with a screwdriver. A large capacitor at high voltage can supply huge amounts of charge in a fraction of a second, melting the end of the screwdriver and flinging molten metal around. People have been blinded by capacitors exploding in this manner, so don't do it.

Figure 11-1 shows the safe way to discharge a capacitor.

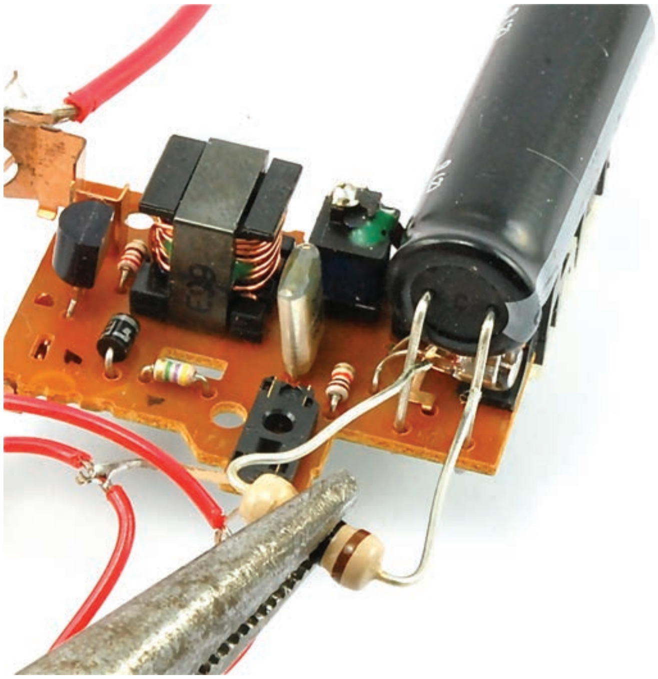


Figure 11-1 Safely discharging a capacitor

The legs of a 100Ω resistor are bent to the right fit for the capacitor contacts and held in the teeth of a pair of pliers for a few seconds. You can use the highest setting of your voltmeter to check that the capacitor has discharged to a safe level (say, 50V). If you have a high-wattage resistor, all the better. If it is not high enough power, it will break, but not in as spectacular a way as a capacitor being discharged dangerously.

Some devices that can pack a painful and sometimes lethal punch are:

- Old glass CRT TVs
- Switch-mode power supplies
- Camera flash guns and disposable cameras with a flash

Taking Something Apart AND Putting It Back Together Again

It is often said that "any fool can take something apart, but putting it back together is a totally different matter."

Just remember that taking things apart usually voids their warranty.

By following a few simple rules, you shouldn't have any problems.

- Have a clear working area with lots of room.
- As you take out the screws, place them in the same pattern on your worktop as they were in the case they came out of. Sometimes the screws can be different sizes. If they are likely to be knocked or roll about on the surface, then push them into a piece of expanded polystyrene or something similar.
- After undoing the screws, when you come to take the case apart, watch out for any little plastic bits like switch buttons that might fall out. Try and keep them in place until you are ready to remove them.
- If something looks tricky, draw a sketch or take a photograph. (I tend to take a lot of photographs when repairing things, like with a hair dryer or straighteners, that have a large mechanical design component.)
- Try not to force things apart. Look to see where the clips are.
- If all else fails, try cutting the case apart with a handsaw (something your author has resorted to in the past), and then later glue the case back together.

Checking a Fuse

The most convenient problem to fix in an appliance is the fuse. It's convenient because it is easy to test and easy to fix. Fuses are basically just wires designed to burn out when the current flowing through them gets too high. This prevents further damage to more expensive components, or can stop a fire from starting.

Sometimes fuses are clear, so you can see that the wire inside them has broken and that they have "blown." Fuses are rated in amps and will generally be labeled to show the maximum current in A or mA they can take. Fuses also come as "fast blow" and "slow blow." As you would expect, this determines how fast the fuses react to over-current.

Some household electrical plugs contain a fuse holder, and you can also find fuses on PCBs. Figures 11-2a-c show the

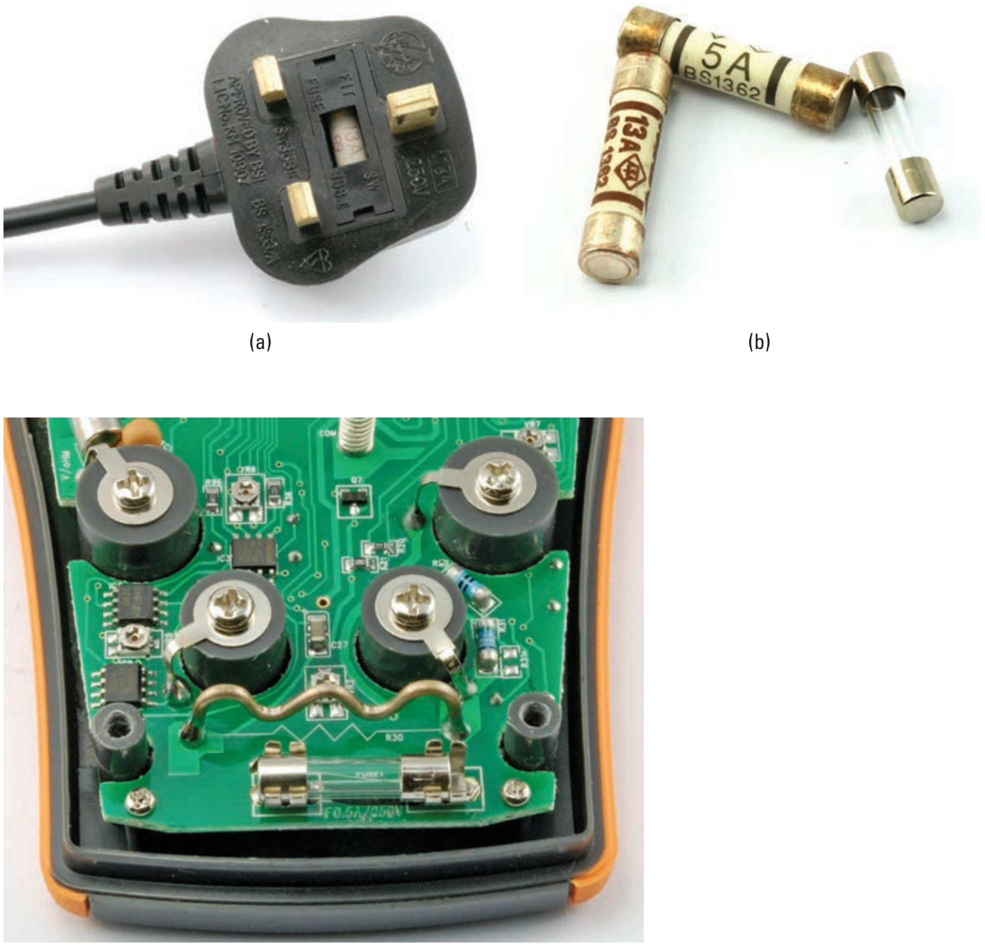


Figure 11-2 Fuses

inside of a UK fused plug and also a fuse holder on the PCB for the author's multimeter.

You have used your multimeter in Continuity mode enough times now that you can probably guess how to test a fuse (Figure 11-3).

If a fuse has blown, there may be a good reason for this. Occasionally, however, they blow for other reasons, such as a momentary spike in the electric power lines or when turning on a heating element on a particularly cold day. So, generally, if there is no obvious sign of a problem with the device

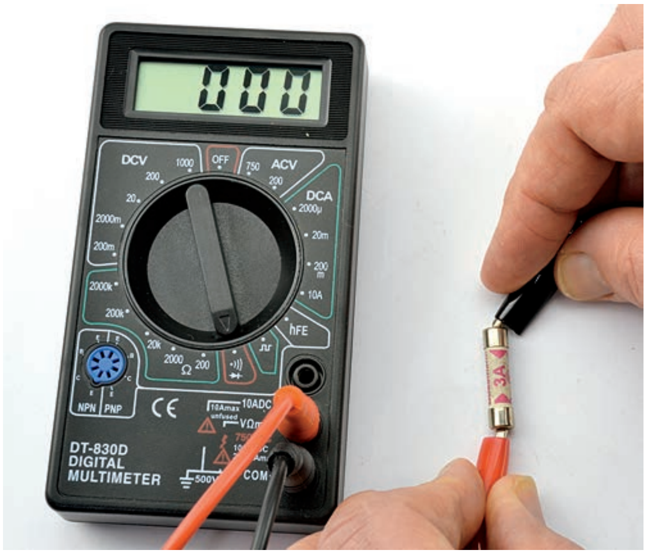


FIGURE 11-3 Testing a fuse with a multimeter

(look for wires that have come lose, or any sign of charring), then try replacing the fuse.

If the fuse immediately blows again, don't try another. You should instead find the source of the problem.

Testing a Battery

Spent batteries are, of course, another common reason for something not working. Simply measuring the voltage will tell you very quickly if the battery is empty.

During testing, if a 1.5V battery like an AA or AAA is showing less than 1.2V, or a 9V battery is showing less than 8V, it is probably time to throw it away. However, the voltage of a battery shown when it is not powering anything can be a little misleading. For a more accurate picture, use a 100Ω resistor as a "dummy" load. Figure 11-4 shows a resistor and multimeter being used to assess the state of the battery.

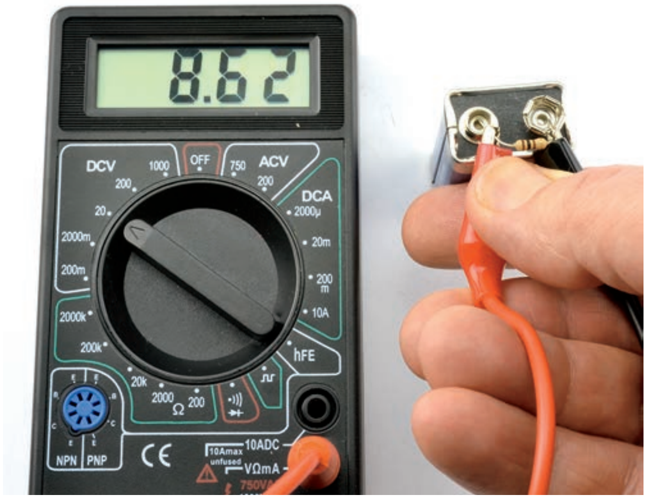


FIGURE 11-4 Testing a battery using a resistor and multimeter

Testing a Heating Element

If you have a suspect heating element from an oven, hair dryer, or so on, you can check it by measuring its resistance. As with anything using household electricity, only do this when the appliance is completely disconnected.

It's a good idea to roughly work out what you think the resistance should be before you measure it. So, for example, if you have a 2-kW 220V heating element, then rearrange:

$P = V^2 / R$

to

$R = V^2 / P = 220 \times 220 / 2000 = 24\Omega$

Calculating what you expect before you measure it is always a good idea, because if you measure it first, it is all too easy to convince yourself that it was what you were expecting. For instance, one time your humble author convinced himself that a suspect element was fine because it was showing a resistance of a few hundred ohms. Eventually, it transpired that there was a light bulb in parallel with the heating element and that the element itself was instead broken.

Finding and Replacing Failed Components

When something stops working on a PCB, it is often the result of something burning out. This sometimes leads to charring around the component. Resistors and transistors are common culprits.

Testing Components

Resistors are easy to test with a multimeter set to its resistance range. Although the results can be misleading, you can test them without removing them. Most of the time, you are looking for an open circuit, very high resistance, or sometimes a short (0Ω).

If your multimeter has a capacitance range, these too can easily be tested.

Other components are less easily identified. It is usually possible to make out some kind of device name on the case.

A magnifying glass is sometimes useful, as is taking a digital photograph and then zooming in to a high magnification. Having found some kind of identifying mark, type it into your favorite search engine.

Bipolar transistors can also be tested (see the section "Using a Multimeter to Test a Transistor" in Chapter 12). However, if you have a spare, it is often easier just to replace it.

Desoldering

There is definitely a knack to desoldering. You often have to add more solder to get the solder to flow. I find it quite effective to draw the solder off onto the tip of the soldering iron, which I keep cleaning using the sponge.

Desoldering braid (Appendix – code T13) is also quite effective. Figure 11-5 shows the steps involved in using desoldering braid to remove the solder from around a component lead so it can be removed.

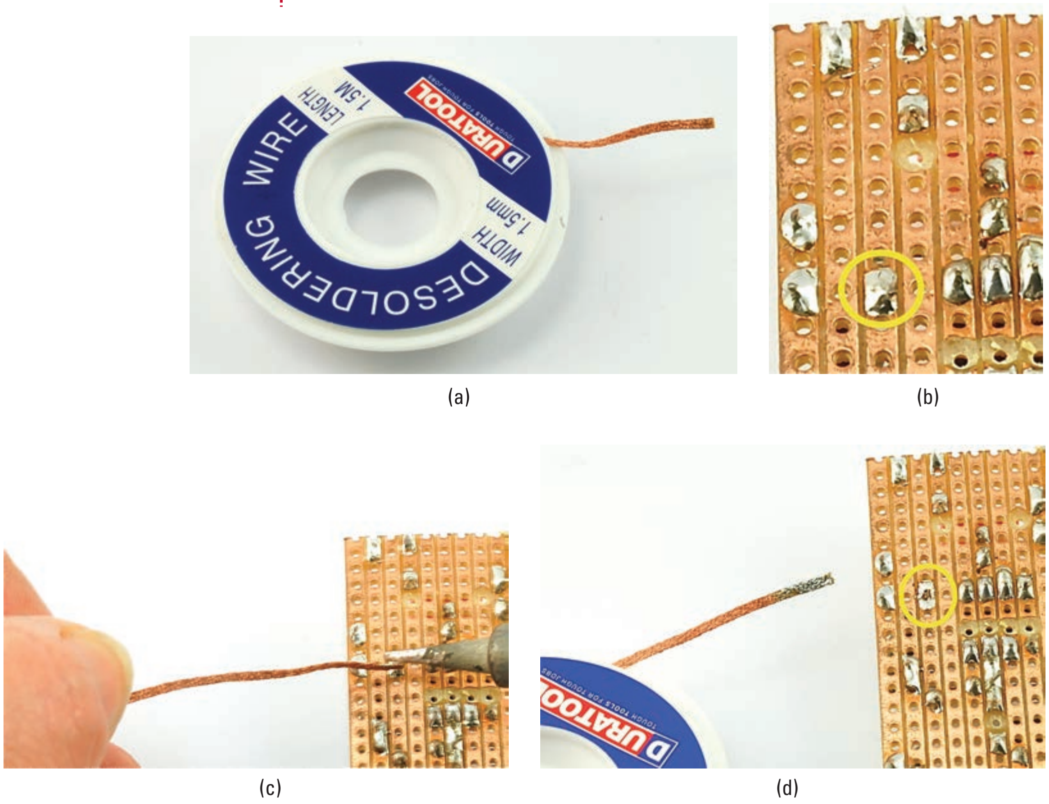


FIGURE 11-5 Using desoldering braid

Desoldering braid (Figure 11-5a) is supplied in small lengths on a small reel. You do not need much. It is braided wire impregnated with flux that encourages the solder to flow into it and off the PCB or stripboard copper.

Figure 11-5b shows the joint (circled in yellow) that we are going to remove the solder from. Press the braid onto the joint with the soldering iron (Figure 11-5c) and you should feel the blob of solder on the joint start to melt into the braid. Remove the braid while everything is hot and you should see a nice clean joint with the solder transferred to the braid (Figure 11-5d).

Cut off the section of the braid with solder on it and throw it away.

You may have to do this a couple of times to remove enough solder to release the component.

Replacement

Soldering in the replacement component is straightforward, you just have to make sure you get it the right way around. This is where photographing the board before making the replacement can be a good idea.

Scavenging Useful Components

Dead consumer electronics are a good source of components. But be selective, because some components are really not worth saving. Resistors are so cheap that it is really not worth the effort of removing them.

Here is what I look for when scavenging:

- Any kind of motors
- Connectors
- Hookup wire
- Seven-segment LED displays
- Loudspeakers
- Switches
- Large transistors and diodes
- Large or unusual capacitors
- Screws nuts and bolts

Figure 11-6 shows the insides of a dead video cassette recorder, with some of the more interesting parts for scavenging labeled.

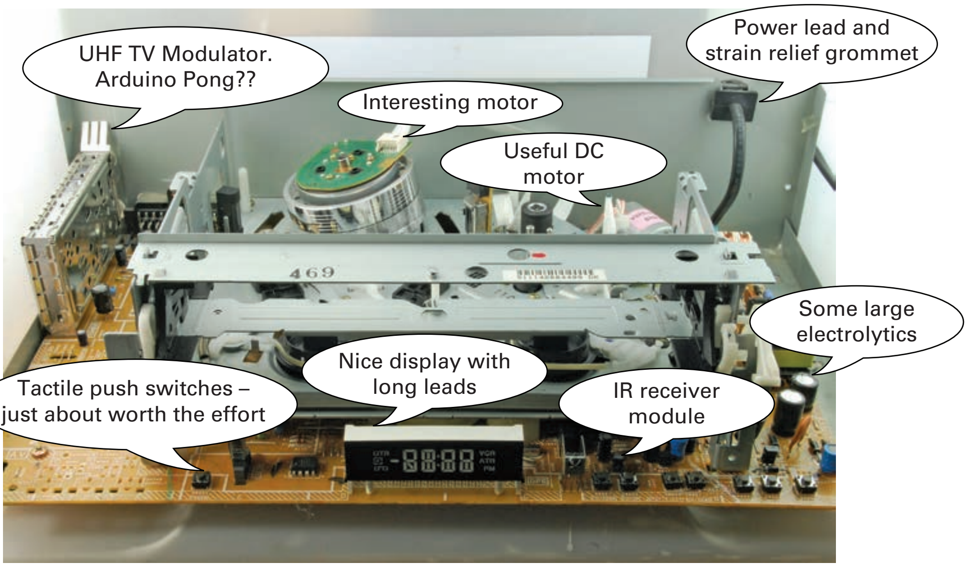

UHF TV Modulator. Arduino Pong??
Interesting motor
Power lead and strain relief grommet
Useful DC motor
Some large electrolytics
Nice display with long leads
Tactile push switches – just about worth the effort
IR receiver module

FIGURE 11-6 Scavenging from a VCR

The easiest way to remove a lot of components, and things like hookup wire, is simply to snip them with wire cutters. The same applies to large electrolytic capacitors and other items, as long as they still leave leads long enough to use. Alternatively, you can desolder the items.

Reusing a Cell Phone Power Adapter

Everything you make in electronics requires a power source of some sort. Sometimes this will be batteries, but often it is more convenient to power the device from your household electricity.

Given that most of us have drawers stuffed with obsolete mobile phones and their charges, it makes sense to be able to reuse an old mobile phone charger. If they are newish phones, they may well have some kind of standard connector on them like a mini-USB or micro-USB, but many older phone models had a proprietary plug, used only by that phone manufacturer.

There is nothing to stop us from taking such an adapter and putting a more standard plug on the end of it, or even connecting the bare wires to screw terminals.

Figure 11-7 shows the steps involved in putting a different type of connector, such as a 2.1mm barrel jack, on the end of an old cell phone charger.

The charger is of the "wall-wart" type that plugs directly into an electrical outlet. The connector is of a type long since discontinued (Figure 11-7a). The charger has a label saying that it can supply 5V at 700mA, so the first step is (making sure the charger is unplugged) to chop off the existing connector and strip the bare wires. There should be two wires, and if one is black and one is red, then the red one is usually positive and the black negative. In this case, the wires are red and yellow. Whatever colors the wires are, it is always a good idea to use a multimeter to check the polarity (Figure 11-7b).

Remember to put the lead through the plastic body of the barrel jack before you start soldering!

You can then solder on a barrel jack plug (Appendix–code H11). This is much the same procedure we used for an audio lead in the section "Hacking Audio Leads" in Chapter 10. Figure 11-7c shows the plug ready to solder, while Figure 11-7d displays the final lead ready to use.

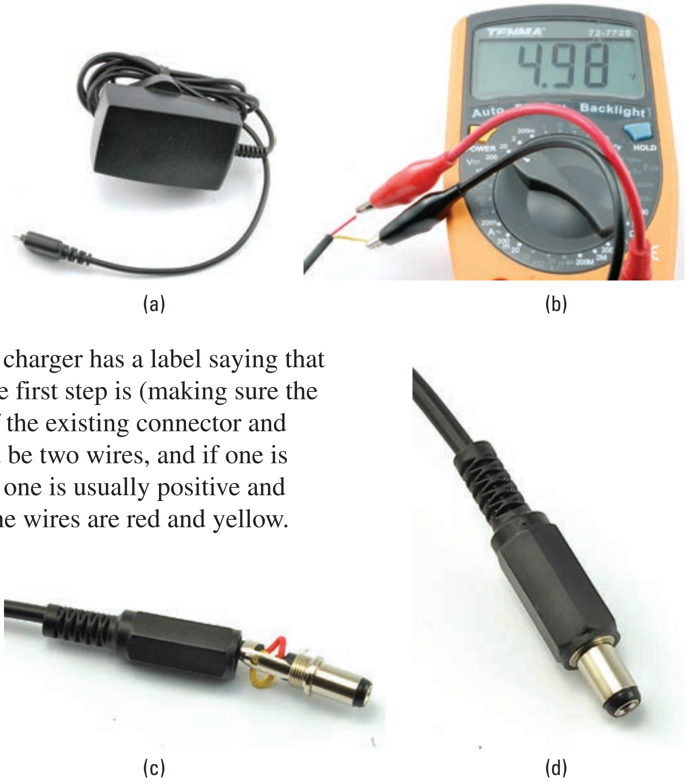


FIGURE 11-7 Attaching a barrel jack to a cell phone charger

Summary

In this chapter, we have discovered some of the treasures that can be rescued from dead electronic equipment and also briefly looked at testing and mending.

If you want to learn more about mending things, I recommend the book *How to Diagnose and Fix Everything Electronic* by Michael Geier (McGraw-Hill Education/TAB, 2011).

12

Tools

This chapter is mainly for reference. You have already met some of the techniques described here while working your way through the book.

Using a Multimeter (General)

Figure 12-1 shows a close-up of the range selector of my multimeter.

This is typical of a medium-range multimeter costing around USD 20. We have probably only used four or five of the settings during the course of this book, so it is worth pointing out some of the other features of a multimeter like this.

Continuity and Diode Test

Starting at the bottom of the range selector, we have the Continuity mode, represented by a little music symbol and also a diode symbol. We have used the Continuity mode many times. It just beeps when there is very low resistance between the leads.

The reason a diode symbol appears here is because this mode also doubles for testing diodes. With some multimeters, this feature will also work on LEDs, allowing you to measure the forward voltage.

Connect the anode of the diode (the end without a stripe in a normal diode, and with a longer lead on an LED) to the red test lead of the multimeter, and then the other end of the diode to the black lead. The meter will then tell you the forward voltage of the diode. So, expect to see about 0.5V for a normal diode and 1.7V to 2.5V for an LED. You will probably also find that the LED glows a little.

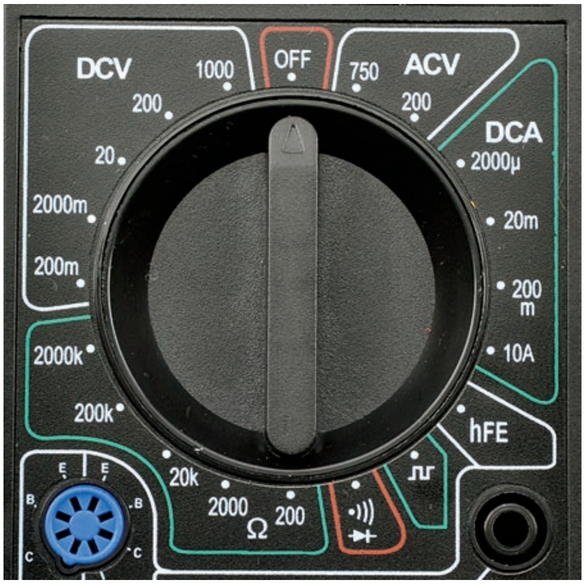


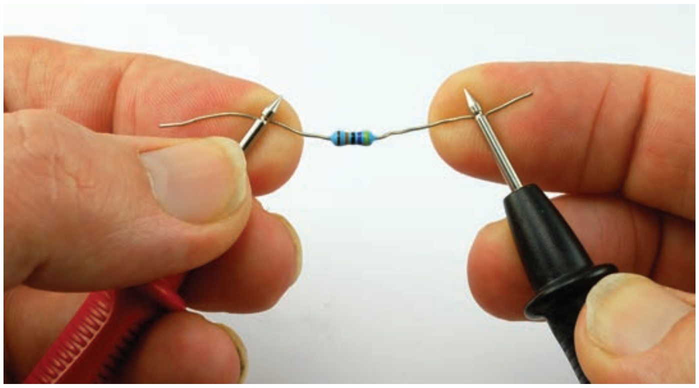
Figure 12-1 Multimeter range selection

Resistance

The multimeter in Figure 12-1 has five resistance ranges, from 2000kΩ down to 200Ω. If you pick a range that has a maximum resistance lower than the resistor you are measuring, then the meter will indicate this. Mine does so by displaying a "1" on its own without any further digits. This tells me I need to switch to a higher resistance range. Even better, start at the maximum range and work your way down until you get a precise reading. For the most precise reading, you need the meter to be on the range above the one that tells you it's out of range.

When measuring high-value resistors of 100kΩ and up, remember that you yourself are also a big resistor, so if you hold the test lead to the resistor at both ends (see Figure 12-2), you are measuring both the resistor in question and your own resistance.

Figure 12-2 How not to measure high-value resistors



Use test leads with crocodile clips, or pin the resistor to your work surface with the flat of the test leads.

Capacitance

Some multimeters include a capacitance range. While not particularly useful for finding the value of unknown capacitors (capacitors have their value written on them), being able to test a capacitor and make sure it still has a capacitance something close to its stated value is useful.

The capacitance range on most meters is quite inaccurate, but then the values of actual capacitors—especially electrolytics—often have quite a wide tolerance.

In other words, if your meter tells you that your 100μF capacitor is actually 120μF, then that is to be expected.

Temperature

If your multimeter has a temperature range, it probably also comes with a special set of leads for measuring it, such as those shown in Figure 12-3.

The leads are actually a thermocouple that can measure the temperature of the tiny metal bead on the end of the leads. This thermometer is a lot more useful than your average digital thermometer. Check the manual for your meter, but the range of temperature is likely to be something like –40°C to 1000°C (–40°F to 1832°F).

So, you can use it to check how hot your soldering iron is getting, or if you have a component in a project that seems to be getting a bit toasty, you can use this to check just how hot it is getting.

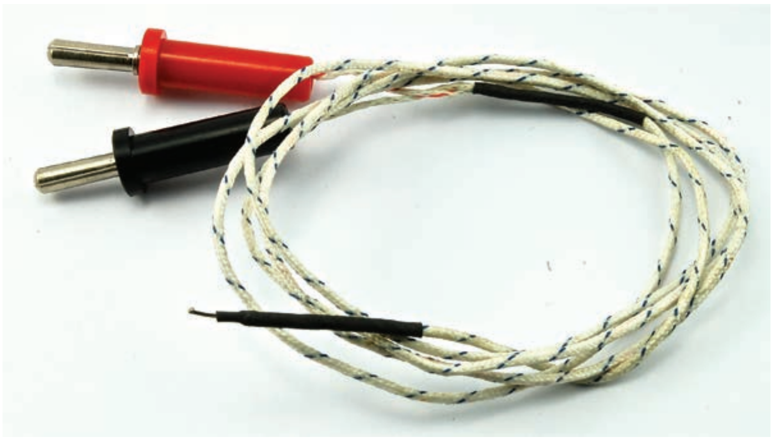


FIGURE 12-3 Thermocouple leads for temperature measurement

AC Voltage

We have not talked about AC very much in this book. AC stands for alternating current and refers to the type of electricity you get in a home wall socket's 110V or 220V supply. Figure 12-4 shows how 110V AC household electricity voltage varies over time.

From Figure 12-4, it is apparent that the voltage actually reaches a peak of 155V and swings all the way negative to –155V. So you might be wondering why it is referred to as 110V at all.

The answer is that since a lot of the time, the voltage is quite low, at those times, it delivers very little power. So the 110V is a kind of average. It's not the normal average voltage, because that would be (110 – 110) / 2 = 0V, and because half the time it is negative.

110V is the RMS voltage (root mean squared). This is the peak positive voltage divided by the square root of 2 (1.4). You can think of this as the DC equivalent voltage. So a light bulb running on 110V AC would appear to be the same brightness as if it were running on 110V DC.

You are unlikely to need to measure AC unless you are doing something exotic and dangerous, and you should not do that unless you are very sure about what you are doing and therefore probably already knew what I just told you.

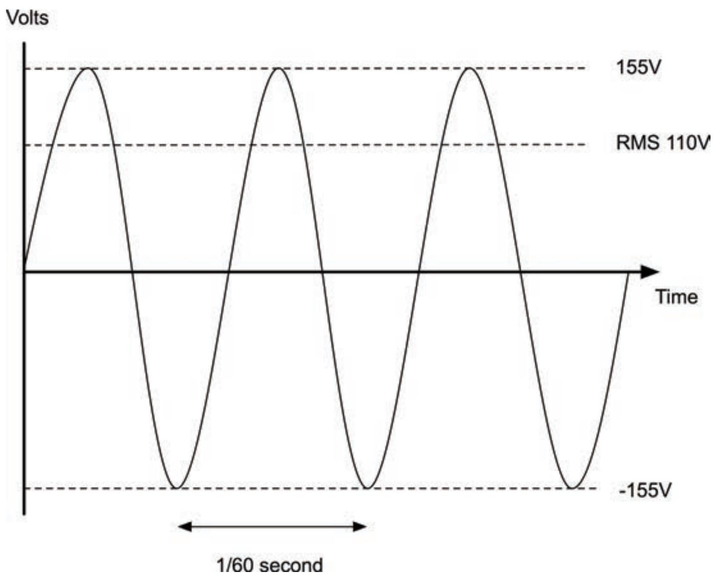


Figure 12-4 Alternating current

DC Voltage

We have already measured DC voltage quite a lot—mostly at the 0 to 20V range.

There is nothing much more to say about this, except to always start with the highest voltage range you believe you are about to measure and then work your way down.

DC Current

When measuring current, you will probably find that for all current ranges you will need to use different sockets on the multimeter for the positive probe lead. There is usually one connection for low currents and a separate one for the high-current ranges (10A on the author's multimeter; see Figure 12-5).

There are two important points to consider here. First, if you exceed the current range, your meter will not just give you a warning, it may well blow a fuse within the meter.

The second point is that when the probe leads are in the sockets for current measurement, there is a very low resistance between them. After all, they need to allow as much of the original current as possible to flow through them. So, if you forget that the leads are in these sockets and go to measure a voltage elsewhere in the circuit, you will effectively short-out your circuit and probably blow the fuse on your multimeter at the same time.

So, just to reiterate, if you have been using your multimeter to measure current, ALWAYS put the probe leads back to their

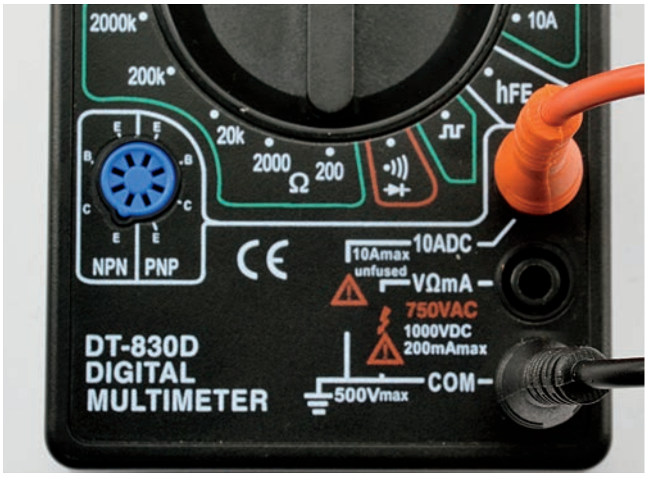


FIGURE 12-5 High-current measurement

voltage sockets, as these are more likely to be used next. If you try and measure current with the leads in the voltage sockets, then all that will happen is that you get a reading of zero.

AC Current

The same argument that we gave for measuring AC voltage applies here. Exercise extreme caution.

Frequency

If your multimeter has a frequency setting, this can be useful. For example, in the section "Generating Tones with a 555 Timer" in Chapter 10 where we create an audio tone using a 555 timer, you could use this feature to measure the frequency of the tone being produced. This can be handy if you do not have access to an oscilloscope.

Using a Multimeter to Test a Transistor

Some multimeters actually have a transistor test socket where you can plug in a transistor. The multimeter will not only tell you if the transistor is alive or dead, but also what its gain (Hfe) is.

If your multimeter does not have such a feature, you can use the diode test feature to at least tell you if the transistor is undamaged.

Figure 12-6 shows the steps involved in testing an NPN bipolar transistor like the 2N3906.

Put the multimeter into diode test mode and attach the negative lead of the meter to the center base connection of the transistor, and the positive lead to one of the other leads of the transistor. It does not matter if it is the emitter or collector (check the transistor's pinout to find the base). You should get a reading, somewhere between 500 and 900. This is the forward voltage in mA between the base and whichever other connection you chose (Figure 12-6a). Then, move the positive lead to the other lead of the transistor (Figure 12-6b) and you should see a similar figure. If either reading is zero, either your transistor is dead, or it is a PNP type of transistor, in which case you need to carry out the same procedure but with the positive and negative leads to the multimeter reversed.

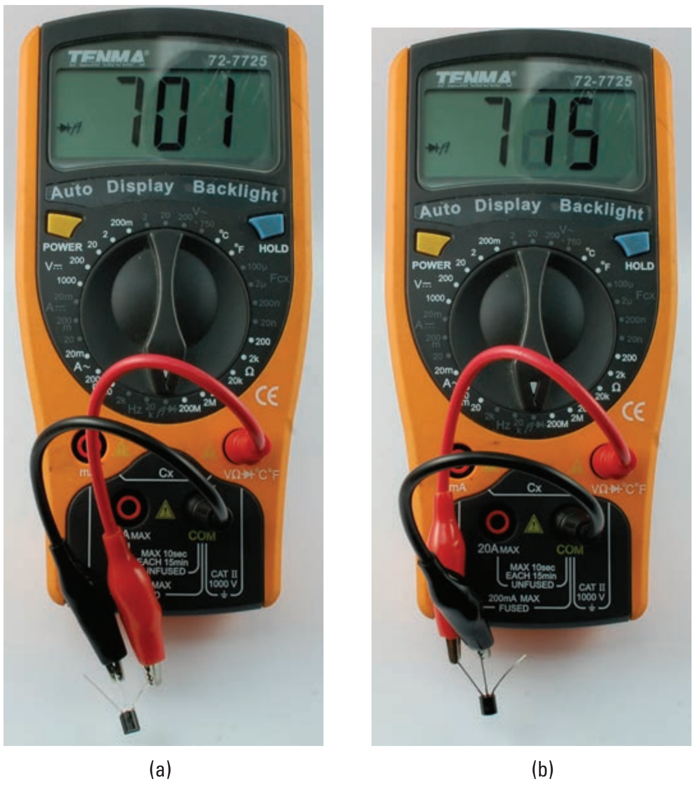


FIGURE 12-6 Testing a transistor

Using a Lab Power Supply

We came across a lab power supply back in Chapter 5. If you have your soldering equipment and a multimeter, then a lab power supply (Figure 12-7) is probably the next item to invest in. It will get a lot of use.

The power supply shown in Figure 12-7 is a simple-to-use basic design. In the figure, it is being used to charge a lead–acid battery. You will find that you use it to power your projects while developing them. You should be able to get something similar for under USD 100.

It plugs into your home electrical socket and can deliver up to 20V at 4A, which is more than enough for most purposes. The screen displays the voltage at the top, and the current being consumed at the bottom.

The reasons why it is more convenient than using batteries or a fixed power supply are

- It displays how much current is being consumed.
- You can limit the current consumption.
- You can use it in constant current mode when testing LEDs.
- You can adjust the voltage easily.

The control panel has an Output switch that turns the output voltage on and off, and two knobs that control the voltage and current.

If I am powering up some project for the first time, I will often follow this procedure:

1. Set the current to its minimum setting.
2. Set the desired voltage.
3. Turn on the output (the voltage will probably drop).
4. Increase the current and watch the voltage rise, making sure that the current isn't rising to an unexpected level.

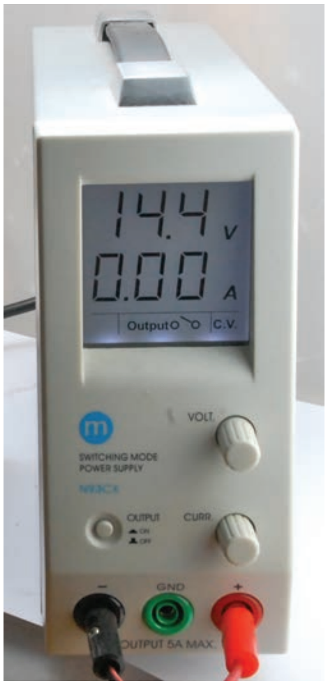
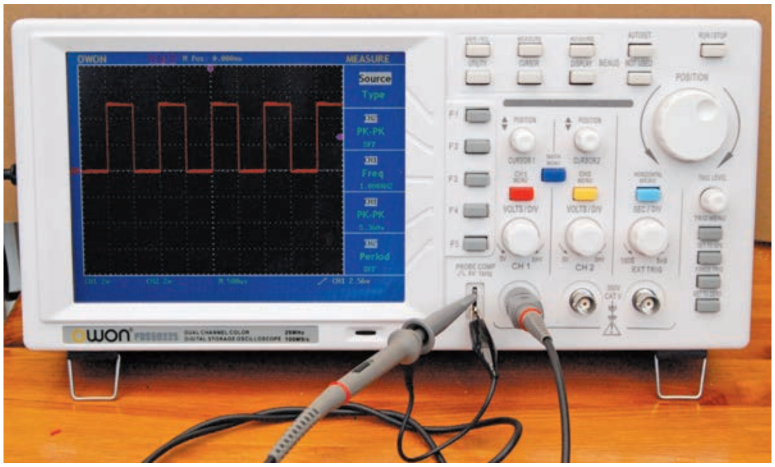


Figure 12-7 A lab power supply

Introducing the Oscilloscope

Oscilloscopes (Figure 12-8) are an indispensable tool for any kind of electronics design or test where you are looking at a

Figure 12-8 A low-cost digital oscilloscope



signal that changes over time. They are a relatively expensive bit of equipment (from USD 200 on up) and there are various kinds. One of the most cost-effective types does not have any display at all, but connects to your computer over USB. If you don't want to risk blobs of solder on your laptop, or wait for it to boot up, then a dedicated oscilloscope is probably best.

Entire books have been written about using an oscilloscope effectively, and every oscilloscope is different, so we will just cover the basics here.

As you can see from Figure 12-8, the waveform is displayed over the top of a grid. The vertical grid is in units of some fraction of volts, which on this screen is 2V per division. So the voltage of the square wave in total is 2.5×2 or 5V.

The horizontal axis is the time axis, and this is calibrated in seconds—in this case, 500 microseconds (μS) per division. So the length of one complete cycle of the wave is 1000 μS—or 1 millisecond—indicating a frequency of 1 KHz.

The other advantage of an oscilloscope is that the test leads are very high impedance, which means that they have very little effect on the thing you are trying to measure.

Software Tools

As well as hardware tools for hacking electronics, there are lots of useful software tools that can help us out.

Simulation

If you like the idea of trying out electronic designs in a virtual world, you should try one of the online simulators like PartSim (www.partsim.com). This online tool (Figure 12-9) allows you to draw your circuits online and simulate how they will behave.

You will have to pick up a bit more theory than this book covers, but a tool like this can save you a lot of effort.

Fritzing

Fritzing (www.fritzing.org) is a really interesting open-source software project that lets you design projects. It is intended

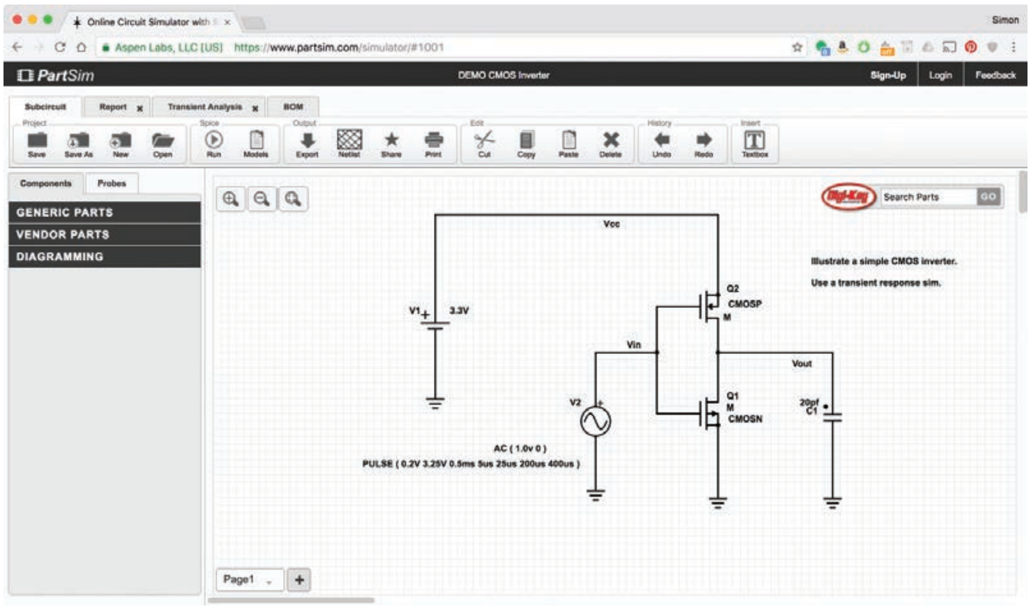


Figure 12-9 The CircuitLab simulator

primarily for breadboard design and includes libraries of components and modules, including Arduino and Raspberry Pi, that can all be wired up (Figure 12-10).

EAGLE PCB

If you want to start creating your own PCBs for your electronics designs, then look for the most popular tool for this, which is called EAGLE PCB (Figure 12-11). It allows you to draw a schematic diagram and then switch to a PCB view where you can route the connections between components before creating the CAM (computer-aided manufacturing) files, which you can then send off to a PCB fabrication shop.

Creating PCBs is a subject in its own right. For more information on this, take a look at the book *Make Your Own PCBs with EAGLE: From Schematic Designs to Finished Boards* by Simon Monk (McGraw-Hill Education/TAB, 2013).

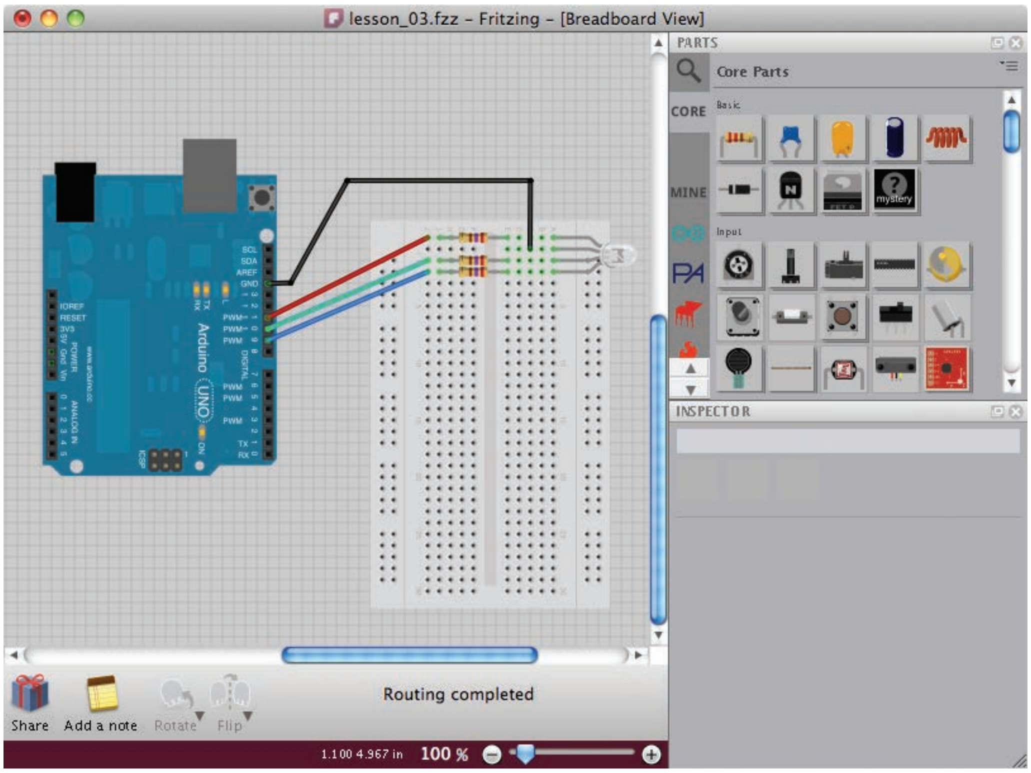


FIGURE 12-10 Fritzing

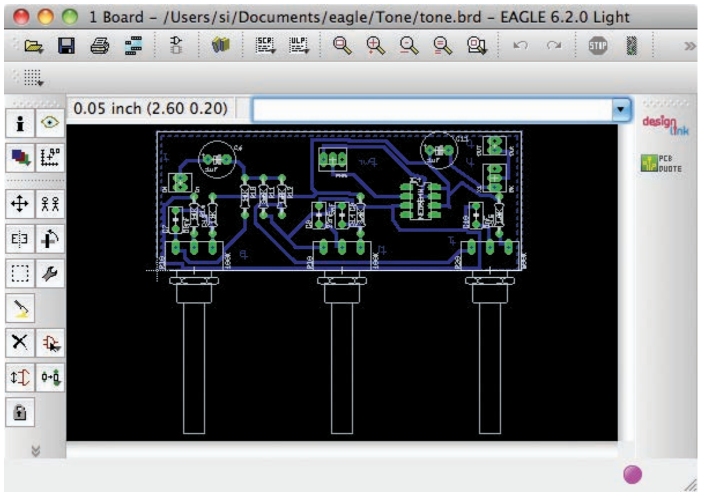

1 Board – /Users/si/Documents/eagle/Tone/tone.brd – EAGLE 6.2.0 Light
0.05 inch (2.60 0.20)
designlink
PCB QUOTE

FIGURE 12-11 EAGLE PCB

Online Calculators

Online calculators can make your electronics math a whole lot easier. Some of the more useful ones are

- **http://led.linear1.org/1led.wiz** A series resistor calculator for LEDs
- **http://led.linear1.org/led.wiz** Designed for driving large numbers of LEDs
- **www.bowdenshobbycircuits.info/555.htm** A 555 timer IC component calculator

Summary

This is the last chapter in this book and I hope it will help you get started "hacking electronics." There is much satisfaction in making something physical, or modifying a device so it does just what you want.

The line between producer and consumer is blurring more and more today as people start designing and building their own electronic devices.

The Internet offers many useful resources. The following web sites are worth a special mention:

- www.hacknmod.com
- www.instructables.com
- www.arduino.cc (for Arduino)
- www.sparkfun.com (modules and interesting components)
- www.adafruit.com (more cool stuff)
- www.dealextreme.com (bargains; search for LEDs, etc.)
- www.ebay.com (search for the same items as that in the other URLs in this list)

See also the components suppliers mentioned in the Appendix.

Appendix

Parts

Prices of components vary enormously, so please treat the following lists as a guide and shop around.

I know some people who buy almost everything on eBay. But beware. Though things are often very cheap there, occasionally they are much more expensive than at other suppliers.

I have listed part codes for the tools, modules, and so on from SparkFun and Adafruit, as these suppliers are very accessible to hobbyists and also provide good accompanying documentation. They also have distributors throughout the world, so you do not have to buy direct from either company if you live outside the U.S.

For other components, I have tried to list product codes for Mouser and DigiKey since these predominate as suppliers to hobbyists in the U.S., and also Farnell, who are UK-based but will ship to anywhere.

Kits to Accompany This Book

I have designed a kit to accompany this book, which includes a multimeter and many of the components used in this book. You can find out more about this kit at http://monkmakes.com/hacking2.

The "You Will Need" sections of the book provide an Appendix Code for the part. If that code is K1, then it means that the part is included in the MonkMakes Hacking Electronics Mega Kit.

Kits are also available from MonkMakes for the robot rover and smart card projects in Chapter 8 (http://monkmakes.com/pi-rover and http://monkmakes.com/cck).

If you need a wider collection of basic components then you will find the MonkMakes Basic Component Pack (https://www.monkmakes.com/basic_comp_pack/) useful.

Tools

Many of the components below as well as other useful things are to be found in Adafruit's "Ladyada's Electronics Toolkit" (https://www.adafruit.com/product/136).

Also, parts with (K1) after them are also included in the MonkMakes Hacking Electronics Mega Kit.

Book Code	Description	SparkFun	Adafruit
T1	Beginner toolkit (soldering kit, pliers, snips)	TOL-09465	
T2	Multimeter	TOL-09141	
T3	PVC insulating tape	PRT-10688	
T4	Helping hands	TOL-09317	ID: 291
T5	Solderless breadboard	PRT-00112	ID: 239
T6 (K1)	Male-to-male jumper wire set	PRT-00124	ID: 758
T7 (K1)	Red hookup wire (22 AWG)	PRT-08023	ID: 288
T8 (K1)	Black hookup wire (22 AWG)	PRT-08022	ID: 290
T9 (K1)	Yellow hookup wire (22 AWG)	PRT-08024	ID: 289
T10	Red multi-core wire (22 AWG)	PRT-08865	
T11	Black multi-core wire (22 AWG)	PRT-08867	
T12 (K1)	Male-to-female jumper set	PRT-09385	ID: 825
T13	Desoldering braid/wick	TOL-09327	ID: 149
T14 (K1)	Female-to-female jumper set	PRT-08430	ID: 266

Components

To get yourselves a basic stock of components, you are strongly recommended to buy a starter kit of components.

Component Starter Kits

As well as the MonkMakes Hacking Electronics Mega Kit, you will find other kits that provide a great way of getting a basic set of components together. Some of these are listed below.

Description	Source
SparkFun Beginner Parts Kit	https://www.sparkfun.com/products/13973
SparkFun Resistor Kit	https://www.sparkfun.com/products/10969
MonkMakes Basic Components Pack	https://www.monkmakes.com/basic_comp_pack/

Resistors

The resistors listed below that have (K1) after their book code are included in the MonkMakes Hacking Electronics 2 Kit.

Book Code	Description	SparkFun	Adafruit	Other
R1 (K1)	10kΩ trimpot, 0.1-inch pitch	COM-09806	ID: 356	DigiKey: 3362P-103LF-ND Mouser: 652-3362P-1-103LF Farnell: 9354301
R2 (K1)	LDR	SEN-09088	ID: 161	DigiKey: PDV-P8001-ND Farnell: 1652637
R3	4.7Ω 0.5W			DigiKey: 4.7H-ND Mouser: 594-SFR16S0004708JA5 Farnell: 2329802
R4	1k NTC Thermistor			Digikey: BC2394-ND Mouser: 995-2DC102K Farnell: 1672360

Capacitors

The capacitors listed below that have (K1) after their book code are included in the MonkMakes Hacking Electronics Mega Kit.

Book Code	Description	SparkFun	Other
C1	1000µF 16V electrolytic		DigiKey: P10373TB-ND Mouser: 667-ECA-1CM102 Farnell: 2113031
C2 (K1)	100µF 16V electrolytic	COM-00096	DigiKey: P5529-ND Mouser: 647-UST1C101MDD Farnell: 8126240
C3 (K1)	470nF capacitor		DigiKey: 445-8413-ND Mouser: 810-FK28X5R1E474K Farnell: 1179637
C4 (K1)	100nF capacitor	COM-08375	DigiKey: 445-5258-ND Mouser: 810-FK18X7R1E104K Farnell: 1216438 Adafruit: 753
C5 (K1)	10µF capacitor	COM-00523	DigiKey: P14482-ND Mouser: 667-EEA-GA1C100 Farnell: 8766894

Semiconductors

The semiconductors listed below that have (K1) after their book code are included in the MonkMakes Hacking Electronics Mega Kit.

Book Code	Description	SparkFun	Adafruit	Other
S1 (K1)	2N3904	COM-00521	756	DigiKey: 2N3904-APTB-ND Mouser: 610-2N3904 Farnell: 9846743
S2 (K1)	High-brightness white LED (5mm)	COM-00531	754	DigiKey: C513A-WSN-CV0Y0151-ND Mouser: 941-C503CWASCBADB152 Farnell: 1716696
S3	1W Lumiled LED on heatsink	BOB-09656	518	DigiKey: 160-1751-ND Mouser: 859-LOPL-E011WA Farnell: 1106587
S4 (K1)	7805 voltage regulator	COM-00107		DigiKey: 296-13996-5-ND Mouser: 512-KA7805ETU Farnell: 2142988
S5 (K1)	1N4001 diode	COM-08589	755	DigiKey: 1N4001-E3/54GITR-ND Mouser: 512-1N4001 Farnell: 1651089
S6 (K1)	FQP30N06L	COM-10213	355	DigiKey: FQP30N06L-ND Mouser: 512-FQP30N06L Farnell: 2453442
S7 (K1)	LM311 comparator			DigiKey: 497-1570-5-ND Mouser: 511-LM311N Farnell: 9755942
S8 (K1)	TMP36 Temperature IC	SEN-10988	165	DigiKey: TMP36GT9Z-ND Farnell: 1438760
S9	TDA7052			DigiKey: 568-1138-5-ND Mouser: 771-TDA7052AN Farnell: 526198
S10 (K1)	NE555 timer IC	COM-09273		DigiKey: 497-1963-5-ND Mouser: 595-NE555P Farnell: 1467742
S11 (K1)	Red LED 5mm	COM-09590	297	DigiKey: 751-1118-ND Mouser: 941-C503BRANCY0B0AA1 Farnell: 1249928
S12	DS18B20 temperature IC	SEN-00245	374	Digikey: DS18B20-ND Mouser: 700-DS18B20 Farnell: 2515605
S13	LM317	COM-00527		Digikey: LM317TFS-ND Mouser: 595-LM317KCSE3 Farnell: 9756027

Hardware and Miscellaneous

The items listed below that have (K1) after their book code are included in the MonkMakes Hacking Electronics Mega Kit.

Book Code	Description	SparkFun	Adafruit	Other
H1 (K1)	4 × AA battery holder	PRT-00550	830	DigiKey: 2476K-ND Mouser: 534-2476 Farnell: 4529923
H2 (K1)	Battery clip			DigiKey: BS61KIT-ND Mouser: 563-HH-3449 Farnell: 1183124
H3	Stripboard			eBay—search for "stripboard" Farnell: 1201473
H4 (K1)	Pin header strip	PRT-00116	392	
H5 (K1)	2A two-way screw terminal			eBay—search for "terminal block" Mouser: 538-39100-1002
H6 (K1)	6V motor or gearmotor			eBay—search for "6V DC motor" or "gearmotor"
H7	Magician chassis	ROB-10825		
H8	6 × AA battery holder		248	DigiKey: BH26AASF-ND Farnell: 3829571
H9 (K1)	Battery clip to 2.1mm jack adapter		80	
H10 (K1)	9g servo motor	ROB-09065	169	
H11	2.1mm barrel jack plug			DigiKey: CP3-1000-ND Farnell: 1737256
H14	8Ω speaker	COM-09151		
H15	Large pushbutton switch	COM-09336	559	
H16 (K1)	5V relay	COM-00100		Digikey: T7CV1D-05-ND
H17 (K1)	MonkMakes protoboard			MonkMakes: SKU00054

Modules

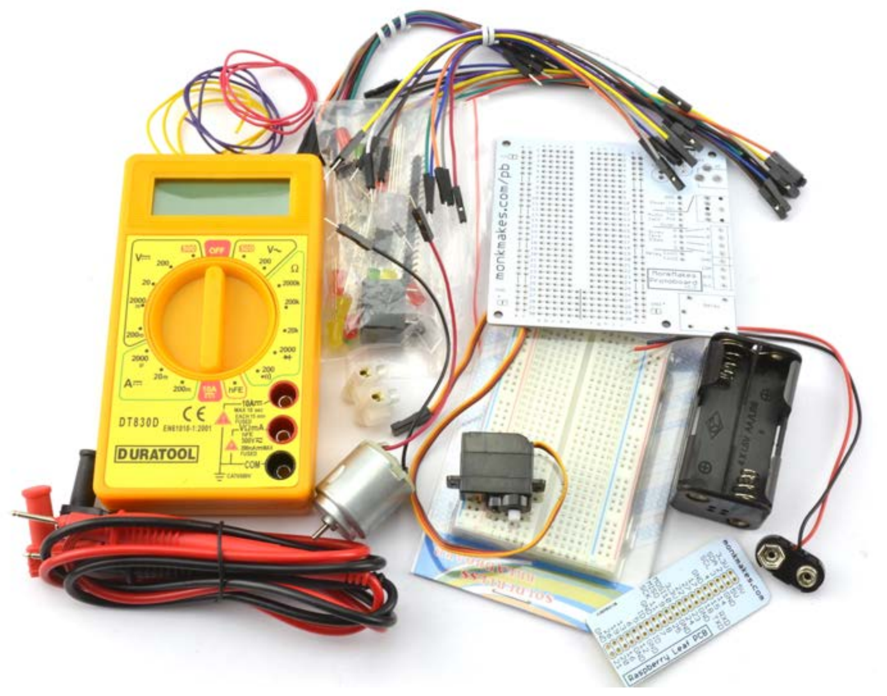
The modules listed below that have (K1) after their book code are included in the MonkMakes Hacking Electronics Mega Kit.

Book Code	Description	SparkFun	Adafruit	Other
M1	12V 500mA power supply	TOL-09442	798	Note: U.S. model listed here.
M2	Arduino Uno R3	DEV-11021	50	
M3 (K1)	Piezo sounder	COM-07950	160	
M4	Mode MCU board			eBay—search for “Node MCU”
M5	PIR module	SEN-08630	189	
M6	Relay module			eBay—search for “1 channel relay module”
M7	HC-SR04 rangefinder			eBay—search for “HC-SR04”
M8	AK-R06A RF Kit			eBay—search for “433MHZ 4 Channel RF Radio”
M9	SparkFun TB6612FNG breakout board	ROB-09457		
M10	Piezo sounder (built-in oscillator)			eBay—search for “Active Buzzer 5V”
M11	Raspberry Pi 3	DEV-13825	3055	
M12	Color sensing module			eBay—search for “TCS3200D Arduino”
M13	MonkMakes RasPiRobot Board V3		1940	Amazon.com
M14	SparkFun mic module	BOB-09964		
M15	Accelerometer module		163	Freetronics: AM3X
M16	USB LiPo charger	PRT-10161	259	
M17	Combined LiPo charger, Buck-booster	PRT-11231		
M18	Arduino LCD shield			eBay—search for “DFRobot LCD shield”
M19	4-digit, 7-segment display w/I2C backpack		880	
M20	RC522 RFID Reader kit			eBay—search for “RC522”
M21	Arduino Leonardo	DEV-11286	849	
M22	6V DC buzzer			eBay—search for “"LZQ-3022 DC buzzer”

Index

Hacking Electronics Mega Kit

This kit doesn't include an Arduino or Raspberry Pi, but it does include everything else you need to get started with your electronics hacking adventure.



- Digital Multimeter with continuity setting
- Half-sized breadboard
- MonkMakes Protoboard
- Jumper lead sets for connecting up breadboard Arduino and Raspberry Pi
- 6V DC motor and servo motor
- A wide selection of components: resistors, capacitors, LEDs, transistors and ICs
- Relay, buzzer and switches

http://monkmakes.com/hacking2