Chapter 1

Exploring the World of Arduino

In This Chapter

- Discovering Arduino
- Understanding who uses Arduino
- Understanding microcontrollers
- Understanding Arduino capabilities

You probably wouldn't have picked up this book if you hadn't already heard about the "World of Arduino." You're probably already a part of it. I think of it as being made up of a community of creative people who are interested in making inanimate stuff do interesting and clever things with computers, programming, and *computational thinking* — which is just a fancy way of saying "writing recipes."

Computational thinking means considering problems and their potential solutions and trying to determine the best way to get to those solutions. Usually, it means deciding the best steps to take — and in what order — as well as keeping track of important decisions along the way, or getting the right information you need to make a decision. This could be doing something simple like baking cookies, in which case you probably don't need a computer. But you can use a little bit of computing power to carry out a simple sequence of steps and decisions to come up with something really creative.

Maybe you want to know when your cat is coming and going from your house. Perhaps you want to know when your houseplants need a little more water and then give it to them automatically. Or suppose that you want to be able to open your front door with a code or card, instead of a physical key. Each of these involves just a little bit of sensing what's going on in the real world, combined with decision making, and then performing some kind of action.

In the case of watering your plants, it's something a human might be prone to forgetting or something you just don't want to pay attention to all the time. Sounds like the perfect job for a computer. That's where Arduino comes to the rescue.

About Arduino

The Arduino Uno (see Figure 1-1) is a general purpose microcontroller programming and prototyping platform that you can easily program to react to things going on in the real world. You can also link between the real world and the virtual world by connecting up your Arduino to the Internet, either sending data to the Internet or responding to data on the Internet, or both.

You can use it to sense almost anything you can find an electronic sensor for, including light, temperature, pressure, sound, even smell — if you consider environmental pollution to be a smell. You can even build your own sensors. How your Arduino reacts depends on how you program it. You can use its output capabilities to sound alarms, open doors and windows, activate lights or motors — the possibilities are almost endless.

Arduino is used for *prototyping* ideas — getting them half built and then trying out what works. Prototyping means testing alternatives to come up with creative solutions to problems (see Figure 1-1). You try out part of a project to see how your sensors respond and then change how your Arduino program functions, depending on what works best for you. Although the projects in this book are like little recipes, they are just a starting point. You could — and should — use any of them to build much more elaborate ideas and projects.

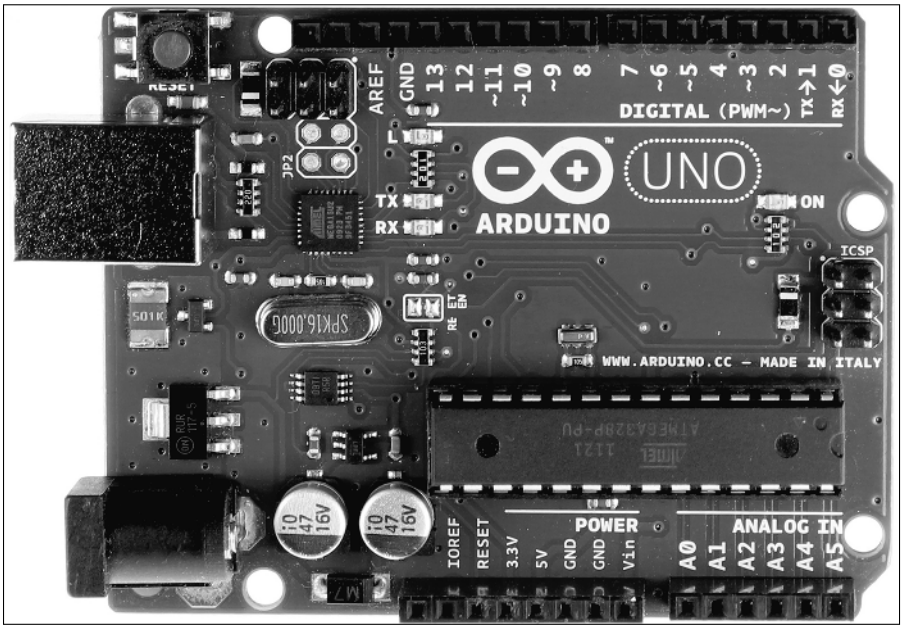

Figure 1-1: The general purpose Arduino Uno prototyping board.

Discovering Who Uses Arduino

The Arduino family is used by makers, hackers, designers, artists, architects, and even professional engineers to quickly and easily try out interactive design ideas. The Arduino Uno is inexpensive and easy to use, with a big community of supporters, tinkerers, and developers who are constantly coming up with new ways to use it and improve it. In the next sections, I go over a few of the kinds of people and communities that are using Arduinos every day.

Arduino in education

Arduino provides a really simple way to learn how to program microcontrollers to sense and react to events in the real world and even online. Because it was conceived as a way to support designers and artists — people who are not typically computer programmers — it is very easy to get started and easy to use. I have taught hundreds of people — from little kids to retirees — to get started programming with Arduino. They have gotten simple programs up and running in as little as a half-hour and built their skills to develop their own sophisticated projects in a weekend. As you see from the projects in this book, it doesn't take long to get your Arduino doing some pretty interesting stuff. And the more time you put into using it, the more you can get out of it.

Art and design schools use Arduino to design new interactive product prototypes, interactive artwork, performances, and even clothing. High schools and secondary schools teach core concepts in computer programming. University students in engineering and computer science departments use Arduino to create interactive models and prototypes as well as learn sophisticated computer-controlled engineering techniques.

Arduino in the corporate world

A growing community of industry professionals in the corporate world use Arduinos to make interactive stuff in their work. Design firms use them to develop interactive product prototypes. Software engineering companies use them to test software systems that interact with the physical world. Ad agencies use them to come up with new and creative interactive campaigns. Arduinos are used to control interactive exhibits and conferences and trade shows in both the industry and in digital media sectors. They are used as management-consulting tools to help teams coordinate problem solving and improve collaboration.

Making and hacking communities

In little pockets all over the world, a new community of tinkerers, makers, and hackers has emerged. Arduino has been a fuel for this creative fire and continues to be one of the key hardware prototyping platforms that people create projects with, talk about, and share with one another.

What are they about?

There have been small electronics and hardware clubs since the early days of the twentieth century, when teenage boys were encouraged to build their own "cat's whisker" radios to listen to the new local radio stations that were popping up all across the United States. Over the decades, a large community of radio buffs grew, especially among fans of the shortwave radio frequencies. These "ham" radio aficionados set up their own transmitters and spent long hours listening to the radio waves for new and far-flung transmissions from friends and strangers. By the 1970s, the stage was set for a whole new generation of electronics fans who started clubs around not just radios but also the newly available home computers. Lots of midnight oil was burned as tinkerers and hobbyists stayed up hacking code and trading ideas on electronic bulletin board systems. This was the breeding ground for some of today's giants, including Apple. Then the Internet exploded onto the scene and changed everything.

At about the same time Arduino was created in 2005, a small subculture emerged that was sort of an extension of the computer clubs and do-it-yourself groups and clubs. Fueled by the Internet, there was sort of a renaissance of computer clubs and do-it-yourself groups, as it became easier to use computers and electronics to make interesting interactive stuff. Some people even call it a "maker movement." The Arduino fits right in with DIY groups, makers, tinkerers, and hackers. There are now hundreds of makerspaces (also called hackspaces) around the world. If you live in a big or medium-size city, there is probably one near you. *Makerspaces* are community-operated physical space where people with common interests (like Arduino!) can meet, get ideas, collaborate, and share accomplishments. Check for a makerspace in your area. These are the best places to learn how to build even more cool stuff with your Arduino.

The open source world

The term *open source* is thrown around a lot these days. If you haven't come across it, you will, because the Arduino is one aspect of the open source world. Open source refers to both a philosophy and a software development approach that advocates for complete transparency in all the points of authorship of software. That lets anyone see how a program is built and potentially contribute to its development. The open source movement is a reaction to the tight control that software companies have had over their products. Their code is intellectual property, and they want to keep control of it both to prevent others from stealing their ideas and to maintain the quality of their products. However, the downside is that consumers are

disempowered from making changes and can sometimes be locked in to buying upgrades they may not want. In principle, anyone with a little know-how can pitch in and contribute to the software development of open source projects, because the code is all online and freely downloadable. The Linux operating system, Google's Android operating system for mobile phones, and Mozilla's Firefox Web Browser are popular examples of open source software.

Thinking about computer hardware as being open source is a relatively new idea, and Arduino is at the forefront. It was conceived as a tool that anyone can build and use to do his own prototyping, using the ATmega328 microcontroller. All the plans to produce your own Arduino are freely available online, and you can put one together without paying anyone else to do so. In practice, it's usually cheaper to buy one, but the principle still holds that the plans are freely available and redistributable.

Contributing to the Arduino project

In the spirit of collaborative development, people are also invited to contribute to the development of the Arduino platform and a thriving community of enthusiasts has contributed to both the hardware development and to the many software libraries that extend Arduino's capabilities. If you want to jump in on the action, all you have to do is join the conversation in the Arduino developer discussion boards and consider writing some libraries of your own. If you are really eager, you may even be able to contribute to the development of the next Arduino board.

Understanding Microcontrollers

The heart of an Arduino is a *microcontroller,* a little computer that performs menial decision-making tasks that might be tedious, too fast, too slow, or otherwise irritating for a human to do. You can make it sense events in the real world and then react to them by doing something. This little guy is perfectly happy to wait for days until the houseplant dries out and then give it a little drink. You simply need to tell him what to wait for and what actions to take. And he's really very little.

Because it's a *micro*controller, it's very small, so it doesn't need much power and can be put into tiny spaces like a project box. How small are microcontrollers? Physically, the one on the Arduino is about as large as they come, about half the size of a pack of gum, as you can see in Figure 1-2. The microcontroller is the rectangular *integrated circuit (IC)* on the blue *printed circuit board (PCB).* It's that size because it's easy to handle with your fingers, so you can replace the microcontroller on your Arduino if it croaks for some reason. But microcontrollers start about this large and go down from there, all the way to the microscopic level. The main factors that determine their size are their capabilities and cost. In fact, the actual processor *core* on your Arduino chip is much, much smaller than the exterior IC chip itself.

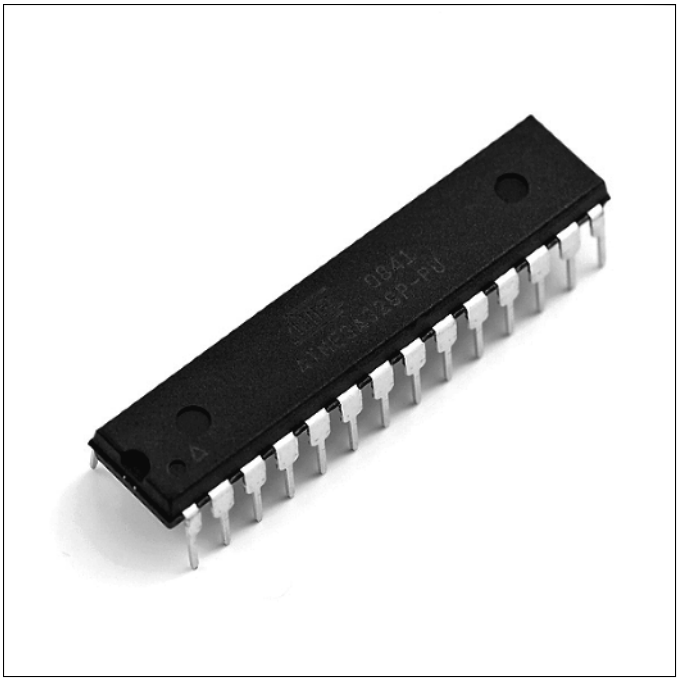

Figure 1-2: The Arduino's brain, an ATmega328 microcontroller.

Along with the processor core, which processes the instructions you give it, the silicon chip has a small memory area for storing your commands, called *program memory* and *random-access memory (RAM),* which is used to keep track of things while the program is running. It also has input and output peripherals to handle sending and receiving data, either in the real world or to other computers, and with the correct code, to the Internet.

Microcontrollers were invented in the early 1970s to do all sorts of everyday automation tasks for industry. Your Arduino uses the single-chip ATmega328 microcontroller, which is part of the AVR family of products from the chipmaker Atmel and was originally developed in the mid-1990s.

The best part about micro*controllers* is that they are inexpensive, unlike their big brothers, the micro*processors* in your computer, laptop, tablet, or phone. Microcontrollers are inexpensive because they have limited capabilities (see Figure 1-2). They are mainly designed to control things or otherwise respond to sensory input, and are called *embedded systems.* Bigger computers have more general capabilities and need more power and therefore, cost more, and use *general purpose* microprocessors.

Because they are inexpensive, you can use them for all kinds of small computing tasks that don't need a full-size computer, like opening your front door with

a code. The microcontroller on your Arduino costs less than a couple of bucks. The rest of the cost of an Arduino comes from all the convenient things that are onboard that help you to send programs to it and interact with the world.

Using tiny computers to do useful stuff

Microcontrollers are the unseen helping hands that are all around us, working tirelessly all the time to make modern life convenient and pleasant. They open doors for us (literally), keep us entertained, and can make a pretty decent cup of coffee. They also ensure that we get from Point A to Point B safely, being embedded in planes, trains, and yes, automobiles. Here are a few examples of what we use them for and similar projects in this book. It's not an exhaustive list, but it should give you an idea of what microcontrollers are used for and how ubiquitous they are!

Toys and games

If you walk into a toy store these days, you come across hundreds of devices that walk, talk, blink, flash, and even respond to how you position their parts or speak to them. Even very inexpensive interactive toys have embedded microcontrollers that perform the same functions as an Arduino. They are usually very tiny and specially designed for mass production and are often hidden under a dab of epoxy on the printed circuit board (PCB) inside the toy, as shown in Figure 1-3. In fact, some products may even use a microcontroller from the same Atmel family. They are programmed at the factory to respond to input and actuate lights, sounds, and movements.

Although it's not interactive, the light pet in Chapter 5 is a simple, preprogrammed toy like many you might see in a store. It's not interactive, but by the time you finish a few projects in this book, you'll be able to make it respond interactively to light, touch, temperature, or other kinds of input.

Home appliances

Your kitchen is almost literally a digital mission control center. A major proportion of the electronic appliances you use to whip up a meal have a microcontroller in them. The microwave has a timer to control power changes and timing. The oven has similar capabilities. A coffee machine also has a timing function and different programs for brewing different cups of java. Advanced food processors sense the consistency of the food mixture and have safety shutoffs. All of these capabilities are done with embedded microcontrollers that sense and respond to the world.

The Arduino Clock in Chapter 7 gives you a taste of what's possible and describes how to build a programmable alarm. With a little further research, you could even hook up its alarm to kick off your own cup of brew!

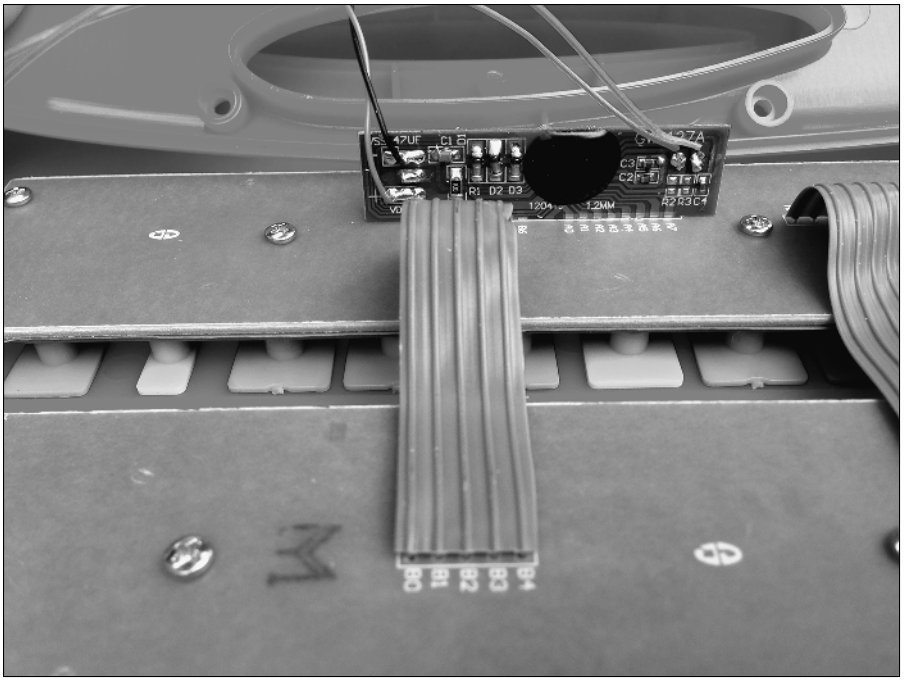

Figure 1-3: A close-up view of a toy's microcontroller hidden under epoxy.

Automated manufacture

If you are building lots of components into a single product, automation is essential and microcontrollers assist with the process. Whether it's a child's toy car or a real car, microcontrollers embedded into the assembly line ensure the precise placement of parts, test for errors in manufacture, adjust the feed of subcomponents, track inventory, and perform other useful functions. Their core capability of sensing the environment and responding quickly, and according to a fixed program, ensures that manufactured products are consistently built and product inventories carefully managed.

The radio frequency ID (RFID) reader in Chapter 9 uses the same RFID technology that many inventory tracking systems use to manage raw materials, parts, and inventory warehouses.

Field sensing and response

Microcontrollers can be placed into conditions where it is simply impractical or downright dangerous to place a human. Imagine you want to ensure that a leak in a gas pipeline doesn't progress into a full-scale explosion. A microcontroller embedded in the line can ensure that the supply is switched off

if a pressure leak is detected. Similarly, you wouldn't want to pay someone to monitor moisture levels in a greenhouse. A microcontroller can activate a spray of water at a fixed interval or according to measured environmental conditions.

The automated plant irrigator in Chapter 10 is a household version of this very useful capability.

Building automation

You are familiar with building security systems to keep out intruders. Along with this, many buildings are now using sensors to detect the internal climate and energy efficiency conditions. Architects now design many modern structures with a "nervous system" of embedded sensors that can adjust heating and cooling automatically, in specific zones or individual rooms, and with the use of energy-efficient heating, cooling, and air handling.

The home sensing project in Chapter 12 is a mini-sized version of a sensor network that you can build in your own home.

Process control

Microcontrollers are used in industry for things such as assembly line control and sensing. For example, microcontrollers can test to find out if all bottles in a line have been filled to the correct level. Microcontrollers attached to sensors can quickly and easily detect problems and either report the fill problem to a central computer or actuate a system to remove the bottle from the line. This can be done much faster than any human could do it. Many product manufacturing processes use microcontrollers because they are cheap and reliable. Similarly, mixing up the raw materials for batches of bread, candy, petroleum products, or concrete can be precisely monitored and controlled with microcontrollers like the one on an Arduino.

Although none of the projects in this book does quite this kind of thing, after you've built a few of them you can figure out how to modify, prototype, and pick and choose from the features you want to build into a project to control many different kinds of processes or activities.

Getting Started

If you haven't already jumped into the middle of the book to check out what you can do, stop now and take a peek. I wrote this book to get you going with some cool Arduino projects so that you can make something amazing

that nobody has dreamed up yet. I hope these projects inspire you. Poking around online may provide additional fuel for your creative fire.

Before you get going, though, it's a good idea to assemble a few tools that will make your Arduino adventures a bit easier. All the projects in this book require some basic tools — and an Arduino. If you are going to dive right in, more power to you. But do take a minute to peruse Chapter 2 to get together a few of the tools you'll need. If you have never used an Arduino before, check out Chapter 3, which covers some of the basics you need to know before you dive into a project.

So what are you waiting for? Take the plunge and get going!