Chapter 1

What Is Arduino and Where Did It Come From?

In This Chapter

- Discovering Arduino
- Learning where Arduino came from and why it's so important
- Introducing the basic principles

Arduino is made up of both hardware and software.

The Arduino board is a printed circuit board (PCB) that is specifically designed to use a microcontroller chip as well as other input and outputs. It also has many other electronic components that are needed for the microcontroller to function or to extend its capabilities.

Microcontrollers are small computers contained within a single, integrated circuit or computer chip, and they are an excellent way to program and control electronics. Many devices, referred to as microcontroller boards, have a microcontroller chip and other useful connectors and components that allow a user to attach inputs and outputs. Some examples of devices with microcontroller boards are the Wiring board, the PIC, and the Basic Stamp.

You write code in the Arduino software to tell the microcontroller what to do. For example, by writing a line of code, you can tell an LED to blink on and off. If you connect a pushbutton and add another line of code, you can tell the LED to turn on only when the button is pressed. Next, you may want to tell the LED to blink only when the pushbutton is held down. In this way, you can quickly build a behavior for a system that would be difficult to achieve without a microcontroller.

Similarly to a conventional computer, an Arduino can perform a multitude of functions, but it's not much use on its own. It requires other inputs or outputs to make it useful. These inputs and outputs allow a computer to sense objects in the world and to affect the world.

Before you move forward, it might help you to understand a bit of the history of Arduino.

Where Did Arduino Come From?

Arduino started its life in Italy, at Interaction Design Institute Ivera (IDII), a graduate school for interaction design. This is a specific school of design education that focuses on how people interact with digital products, systems, and environments and how they in turn influence us.

The term *interaction design* was coined by Bill Verplank and Bill Moggridge in the mid-1980s. The sketch in Figure 1-1 by Verplank illustrates the basic premise of interaction design. This diagram is an excellent illustration of how the process of interaction works: If you do something, you feel a change, and from that you can know something about the world.

Although it is a general principle, interaction design more commonly refers to how we interact with conventional computers by using peripherals, such as mice, keyboards, and touchscreens, to navigate a digital environment that is graphically displayed on a screen.

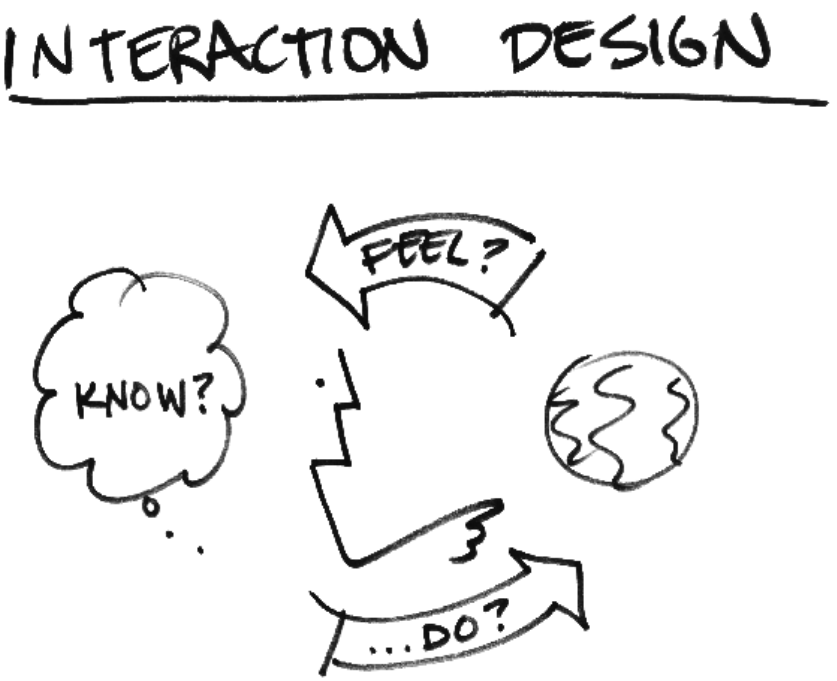

Figure 1-1: The principle of interaction design, illustrated by Bill Verplank.

Courtesy of Bill Verplank

There is another avenue, referred to as physical computing, which is about extending the range of these computer programs, software, or systems. Through electronics, computers can sense more about the world and have a physical impact on the world themselves.

Both of these areas — interaction design and physical computing — require prototypes to fully understand and explore the interactions, which presented a hurdle for nontechnical design students.

In 2001, a project called Processing that was started by Casey Reas and Benjamin Fry aimed to get nonprogrammers into programming by making it quick and easy to produce onscreen visualizations and graphics. The project gave the user a digital sketchbook on which to try ideas and experiment with a very small investment of time. This project in turn inspired a similar project for experimenting in the physical world.

Building on the same principles as Processing, in 2003 Hernando Barragán started developing a microcontroller board called Wiring. This board was the predecessor to Arduino.

In common with the Processing project, the Wiring project also aimed to involve artists, designers, and other nontechnical people, but Wiring was designed to get people into electronics rather than programming. The Wiring board (shown in Figure 1-2) was less expensive than some other microcontrollers, such as the PIC and the Basic Stamp, but it was still a sizable investment for students to make.

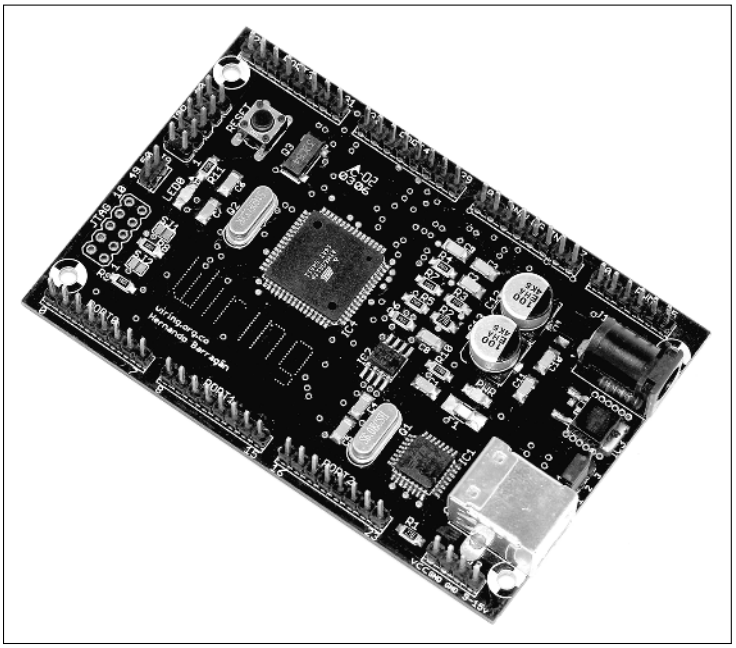

Figure 1-2: An early Wiring board.

In 2005, the Arduino project began in response to the need for affordable and easy-to-use devices for Interaction Design students to use in their projects. It is said that Massimo Banzi and David Cuartielles named the project after Arduin of Ivera, an Italian king, but I've heard from reliable sources that it also happens to be the name of the local pub near the university, which may have been of more significance to the project.

The Arduino project drew from many of the experiences of both Wiring and Processing. For example, an obvious influence from Processing is the *graphic user interface* (GUI) that is used in the Arduino software. This GUI was initially "borrowed" from Processing, and even though it still looks similar, it has since been refined to be more specific to Arduino. I cover the Arduino interface in more depth in Chapter 4.

Arduino also kept the naming convention from Processing, naming its programs *sketches*. In the same way that Processing gives people a digital sketchbook to create and test programs quickly, Arduino gives people a way to sketch out their hardware ideas as well. Throughout this book, I show many sketches that allow your Arduino to perform a huge variety of tasks. By using and editing the example sketches in this book, you can quickly build up your understanding of how they work and will be writing your own in no time. Each sketch is followed with a line-by-line explanation of how it works to ensure that no stone is left unturned.

The Arduino board, shown in Figure 1-3, was made to be more robust and forgiving than Wiring or other earlier microcontrollers. It was not uncommon for students and professions, especially those from a design or arts background, to break their microcontroller within minutes of using it, simply by getting the wires the wrong way around. This fragility was a huge problem, not only financially but also for the success of the boards outside technical circles.

It is also possible to change the microcontroller chip on an Arduino, so if it is damaged, you can just replace the chip rather than the whole board.

Another important difference between Arduino and other microcontroller boards is the cost. In 2006, another popular microcontroller, the Basic Stamp, cost nearly four times as much (`http://blog.makezine.com/2006/09/25/arduino-the-basic-stamp-k/`) as an Arduino, and even today, a Wiring board still costs nearly double the price of an Arduino.

In one of my first Arduino workshops, I was told that the price was intended to be affordable for students. The price of a nice meal and a glass of wine at that time was about 30 euros, so if you had a project deadline, you could choose to skip a nice meal that week and make your project instead.

The range of Arduino boards on the market is a lot bigger than it was back in 2006. In Chapter 2, you learn about just a few of the most useful Arduino and Arduino-compatible boards and how they differ to provide you with a variety

of solutions for your own projects. Also, in Chapter 13 you learn all about a special type of circuit board called a shield, which can add useful, and in some cases phenomenal, features to your Arduino, turning it into a GPS receiver, a Geiger counter, or even a mobile phone, to name just a few.

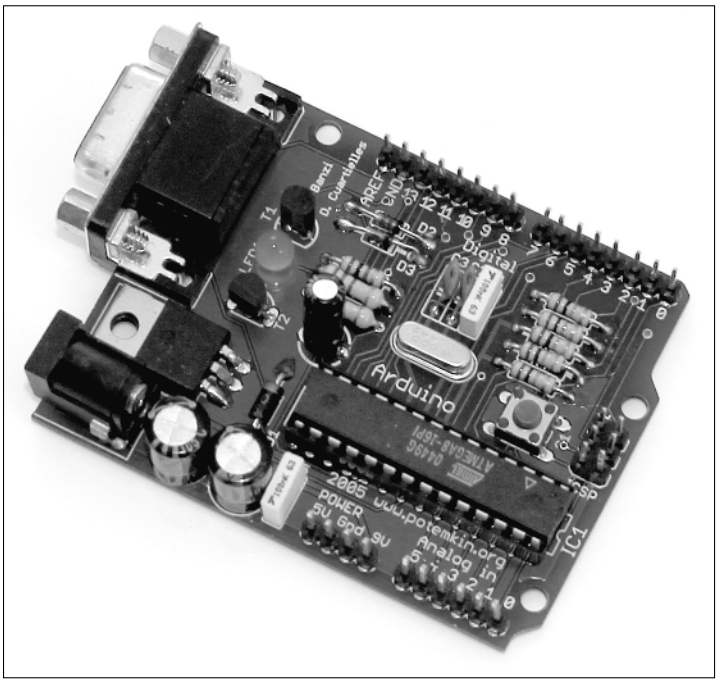

Figure 1-3: The original Arduino Serial board.

Learning by Doing

People have used technology in many ways to achieve their own goals without needing to delve into the details of electronics. Following are just a few related schools of thought that have allowed people to play with electronics.

Patching

Patching isn't just a town in West Sussex; it is also a technique for experimenting with systems. The earliest popular example of patching is in phone switchboards. For an operator to put you through to another line they had to physically attach a cable. This was also a popular technique for synthesizing music, such as with the Moog synthesizer.

When an electronic instrument generates a sound, it is really generating a voltage. Different collections of components in the instrument manipulate that voltage before it is outputted as an audible sound. The Moog synthesizer works by changing the path that that voltage takes, sending it through a number of different components to apply different effects.

Because so many combinations are possible, for the musician the experience is largely based on trial and error. But the simple interface means that this process is extremely quick and requires very little preparation to get going.

Hacking

Hacking is popular term and is commonly used to refer to subversive people on the Internet. More generally, though, it refers to exploring systems and making full use of them or repurposing them to suit your needs.

Hacking in this sense is possible in hardware as well as software. A great example of hardware hacking is a keyboard hack. Say that you want to use a big, red button to move through a slideshow. Most software has keyboard shortcuts, and most PDF viewers move to the next page when the user presses the spacebar. If you know this, then you ideally want a keyboard with only a spacebar.

Keyboards have been refined so much that inside a standard keyboard is a small circuit board, a bit smaller than a credit card (see Figure 1-4). On it are lots of contacts that are connected when you press different keys. If you can find the correct combination, you can connect a couple of wires to the contacts and the other ends to a pushbutton. Now every time you hit that button, you send a space to your computer.

This technique is great for sidestepping the intricacies of hardware and getting the results you want. In the bonus chapter (`www.dummies.com/go/arduinofd`), you learn more about the joy of hacking and how you can weave hacked pieces of hardware into your Arduino project to control remote devices, cameras, and even computers with ease.

Figure 1-4: The insides of a keyboard, ready to be hacked.

Circuit bending

Circuit bending flies in the face of traditional education and is all about spontaneous experimentation. Children's toys are the staple diet of circuit benders, but really any electronic device has the potential to be experimented with.

By opening a toy or device and revealing the circuitry, you can alter the path of the current to affect its behavior. Although this technique is similar to patching, it's a lot more unpredictable. However, after you find the combinations, you can also add or replace components, such as resistors or switches, to give the user more control over the instrument.

Most commonly, circuit bending is about sound, and the finished instrument becomes a rudimentary synthesizer or drum machine. Two of the most popular devices are the *Speak & Spell* (see Figure 1-5) and the *Nintendo GameBoy*. Musicians such as the Modified Toy Orchestra (`modifiedtoyorchestra.com`), in their own words, "explore the hidden potential and surplus value latent inside redundant technology." So think twice before putting your old toys on eBay!

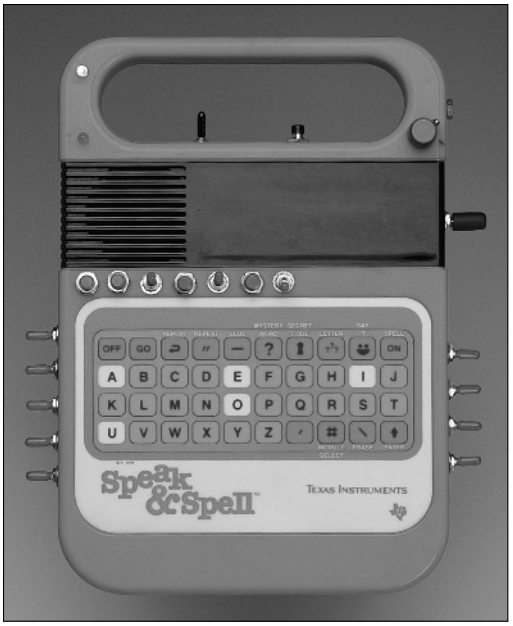

Figure 1-5: A Modified Toy Orchestra Speak & Spell after circuit bending.

Courtesy of Modified Toy Orchestra

Electronics

Although there are many ways to work around technology, eventually you'll want more of everything: more precision, more complexity, and more control.

If you learned about electronics at school, you were most likely taught how to build circuits using specific components. These circuits are based solely on the chemical properties of the components and need to be calculated in detail to make sure that the correct amount of current is going to the correct components.

These are the kind of circuits you find as kits at Radio Shack (or Maplin, in the United Kingdom) that do a specific job, such as an egg timer or a security buzzer that goes off when you open a cookie jar. These are very good at their specific job, but they can't do much else.

This is where microcontrollers come in. Microcontrollers are tiny computers, and if used in conjunction with analog circuitry, can give that circuitry a more advanced behavior. They can also be reprogrammed to perform different functions as needed. Your Arduino is actually designed around one of these microcontrollers and helps you get the most out of it. In Chapter 2, you look closely at an Arduino Uno to see exactly how it is designed and what it is capable of.

The microcontroller is the brains of a system, but it needs data to either sense things about or affect things in its environment. It uses inputs and outputs to do so.

Inputs

Inputs are senses for your Arduino. They tell it what is going on in the world. At its most basic, an input could be a switch, such as a light switch in your home. At the other end of the spectrum, it could be a gyroscope, telling the Arduino the exact direction it's facing in three dimensions. You learn all about basic inputs in Chapter 7, and more about the variety of sensors and when to use them in Chapter 12.

Outputs

Outputs allow your Arduino to affect the real world in some way. An output could be very subtle and discreet, such as in the same way that a mobile phone vibrates, or it could be a huge visual display on the side of a building that can be seen for miles around. The first sketch in the book walks you through "blinking" an LED (see Chapter 4). From there you can go on to motor control (Chapter 8) and even controlling huge numbers of outputs (see Chapters 14 and 15) to discover a variety of outputs for your Arduino project.

Open Source

Open source software, in particular Processing, has had a huge influence on the development of Arduino. In the world of computer software, open source is a philosophy involving sharing the details of a program and encouraging others to use, remix, and redistribute them, as they like.

Just as the Processing software is open source, so are Arduino software and hardware. This means that the Arduino software and hardware are both released freely to be adapted as needed. Possibly because of this openness on the part of the Arduino team, you find the same open source community spirit in the Arduino forums.

On the official Arduino forums (`www.arduino.cc/forum/`) and many other ones around the world, people have shared their code, projects, and questions for an informal peer review. This sharing allows all sorts of people, including experienced engineers, talented developers, practiced designers,

and innovative artists, to lend their expertise to complete novices in some or all of these areas. It also provides a means to gauge people's areas of interest, which then occasionally filters into the official release of Arduino software or board design with new refinements or additions. The Arduino website has an area known as the Playground (`www.playground.arduino.cc`) where people are free to upload their code for the community to use, share, and edit.

This kind of philosophy has encouraged the relatively small community to pool knowledge on forums, blogs, and websites, thereby creating a vast resource for new Arduin-ists to tap into.

There is also a strange paradox that despite the open source nature of Arduino, a huge loyalty to Arduino as a brand exists — so much so that there is an Arduino naming convention of adding -duino or -ino to the name of boards and accessories (much to the disgust of Italian members of the Arduino team)!