LEARNING MADE EASY

2nd Edition

Arduino®

for dummies®
A Wiley Brand

Learn to build circuits with Arduino

Write simple code to control circuits

Create amazing interactive projects

John Nussey

Arduino®

2nd Edition

by John Nussey

for dummies®
A Wiley Brand

Arduino® For Dummies®, 2nd Edition

Published by: **John Wiley & Sons, Inc.,** 111 River Street, Hoboken, NJ 07030-5774, www.wiley.com

Library of Congress Control Number: 2018951004

ISBN 978-1-119-48954-2 (pbk); ISBN 978-1-119-48955-9 (ebk); ISBN 978-1-119-48957-3 (ebk)

Manufactured in the United States of America

10 9 8 7 6 5 4 3 2 1

Contents at a Glance

Table of Contents

Foreword

The moment a *For Dummies* book comes out, it's definitely a milestone in the history of a product.

Programming embedded computers used to be a very difficult task, reserved only to experienced engineers willing to master the obscure assembly language. In recent years, however, many platforms have tried to make this task simpler and more accessible to everyday people. Arduino is one of the latest attempts at making technology less scary and more creative.

With John, this book's author, we watched this creative tool being adopted by designers and artists in London, making its way into many memorable projects. Now Arduino has escaped the lab of Arts & Design and spread like a virus, becoming the tool of choice for all kinds of people who have great ideas they want to realize.

I'm really glad that John decided to write this book, because he's an early user of the Arduino platform from back in the days when it was still quite experimental. Having taught Arduino classes for many years, he has the ability to introduce the subject to all audiences.

Any newcomer to Arduino will, with the right tools and teaching — such as those found in this book — show true genius in no time.

— Massimo Banzi

Introduction

Arduino is a tool, a community, and a way of thinking that is affecting how we use and understand technology. It has rekindled a love and understanding of electronics for many people, including myself, who felt that electronics was something that they had left behind at school.

Arduino is tiny circuit board that has huge potential. It can be used to blink a Morse-code signal using a single light-emitting diode (LED) or to control every light in a building, depending on how far you take it. Its capabilities are limited only by your imagination.

Arduino is also providing a new, practical approach to technical education, lowering the entry level for those wanting to use electronics to complete small projects and, I hope, encouraging you to read further to take on big ones.

A huge and ever-growing community of Arduinists has emerged — users and developers who learn from each other and contribute to the open-source philosophy by sharing the details of their projects. This open-source attitude is responsible for the huge popularity of Arduino.

Arduino is more than just a gadget; it's a tool. A piece of technology that makes understanding and using today's technology easier.

So if the prospect of understanding the limitless possibilities of technology doesn't sound interesting to you, please put this book down and back away.

Otherwise, read on!

About This Book

This is a technical book, but it's not for technical people only. Arduino is designed to be usable by anyone, whether they're technical, creative, crafty, or just curious. All you need is an open mind or a problem to fix and you'll soon find ways that using Arduino can benefit you.

This book starts on the most basic level to get you started with using and understanding Arduino. At times throughout the book, I may refer to a number of technical things that will, like anything, take time to understand. I guide you through all the basics and then on to more advanced activities.

Much of what is in this book is based on my learning and teaching experiences. I learned all about Arduino from scratch, but have always found that the best way to learn is in practice, by making your own projects. The key is to understand the basics that I cover in this book and then build on that knowledge by thinking about how you can apply it to solve problems, create things, or just entertain yourself.

Foolish Assumptions

I assume nothing about your technical knowledge. Arduino is an easy-to-use platform for learning about electronics and programming. It is for people from all walks of life, whether you're a designer, an artist, or a hobbyist.

It can also be a great platform for people who are already technical. Maybe you've done a bit of coding but want to bring your projects into the physical world in some way. Or maybe you've worked with electronics and want to see what Arduino can bring to the table.

Whoever you are, you'll find that Arduino has great potential. It's really up to you to decide what to make of it.

Icons Used in This Book

Arduino For Dummies uses icons to highlight important points for you. Keep an eye out for the following:

TIP

This icon highlights a bit of helpful information. That info may be a technique to help you complete a project more easily or the answer to a common problem.

WARNING

Arduinos aren't dangerous on their own; indeed, they're made to be extremely safe and easy to use. But if you use them in a circuit without proper planning as well as care and attention, they can damage your circuit, your computer, and yourself. When you see a Warning icon, please take special note.

REMEMBER

Often, you must consider certain points before proceeding with a task. I use Remember icons to remind you of such points.

TECHNICAL STUFF

Some information is more technical than others and is not for the faint-hearted. The joy of Arduino is that you don't need to fully understand the technical details immediately. You can skip anything that's marked with this icon if it's more complicated than you want to deal with at the moment; you can always return to it when you're ready.

Beyond the Book

In addition to what you're reading right now, this product comes with a free access-anywhere Cheat Sheet that provides information on using resistors, getting the tools you'll need, and some system shortcuts. To get this Cheat Sheet, simply go to `www.dummies.com` and type *Arduino For Dummies 2nd Edition Cheat Sheet* in the Search box. I also provide a bonus chapter that teaches you all about using your Arduino to hack other hardware, such as games, controllers, and toys.

Where to Go from Here

If you're uncertain about where to start, I suggest the beginning. By the end of Chapter 2, you'll have acquired a simple understanding of Arduino and will know where you can get a kit to continue learning.

If you've used Arduino before, you may want to jump straight to Chapter 4 to cover the basics again, or head straight to the area that interests you.

1

Getting to Know Arduino

IN THIS PART . . .

Find out all about the little blue circuit board.

Discover everything you need to get started with Arduino and where to get them.

Learn how to wield the awesome power of an LED, blinking in on command with a few simple lines of code.

IN THIS CHAPTER

- » Discovering what Arduino is
- » Learning where Arduino came from
- » Introducing the basic principles

Chapter 1 Discovering Arduino

Arduino is made up of both hardware and software.

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

A microcontroller is a small computer contained in a single, integrated circuit or computer chip. Microcontrollers are an excellent way to program and control electronics. *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 to-do. For example, by writing a line of code, you can tell an light-emitting diode (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.

Similar to a conventional computer, an Arduino can perform a multitude of functions, but it's not much use on its own. It requires inputs or outputs to make it useful. These inputs and outputs allow a computer — and an Arduino — 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 Ivrea (IDII), a graduate school for interaction design 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: If you do something, you feel a change, and from that you can know something about the world.

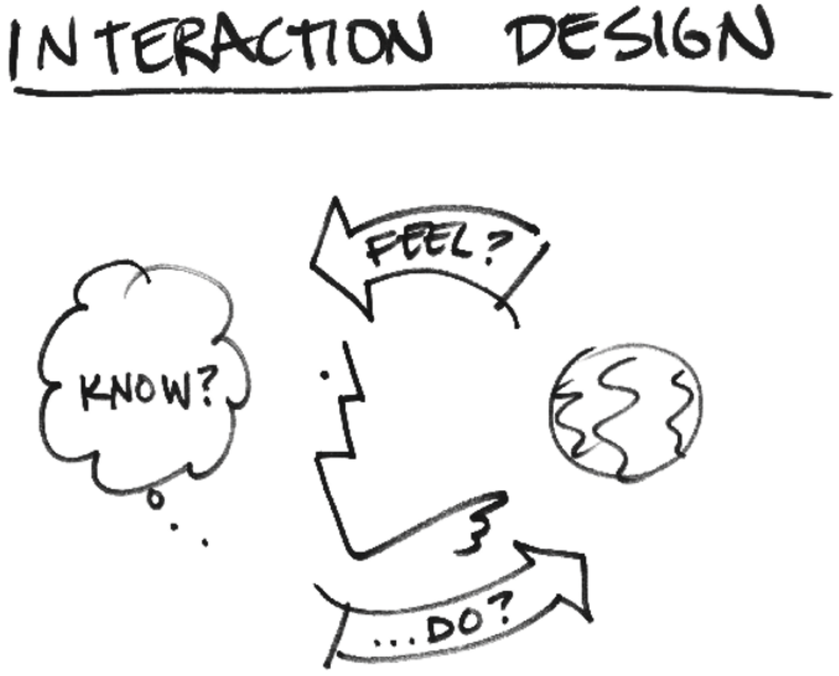

FIGURE 1-1: The principle of interaction design, illustrated by Bill Verplank.

Courtesy of Bill Verplank

Although interaction design is a general principle, it more commonly refers specifically 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.

Another avenue, referred to as *physical computing,* is about extending the range of these computer programs, software, or systems through electronics. By using electronics, computers can sense more about the world and have a physical effect on the world themselves.

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

In 2001, a project called Processing, started by Casey Reas and Benjamin Fry, aimed to get non-programmers 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 small investment of time. This project in turn inspired a similar project for experimenting in the physical world.

In 2003, building on the same principles as Processing, 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 non-technical people. However, Wiring was designed to get people into electronics as well as 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.

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 Ivrea, 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) 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 3.

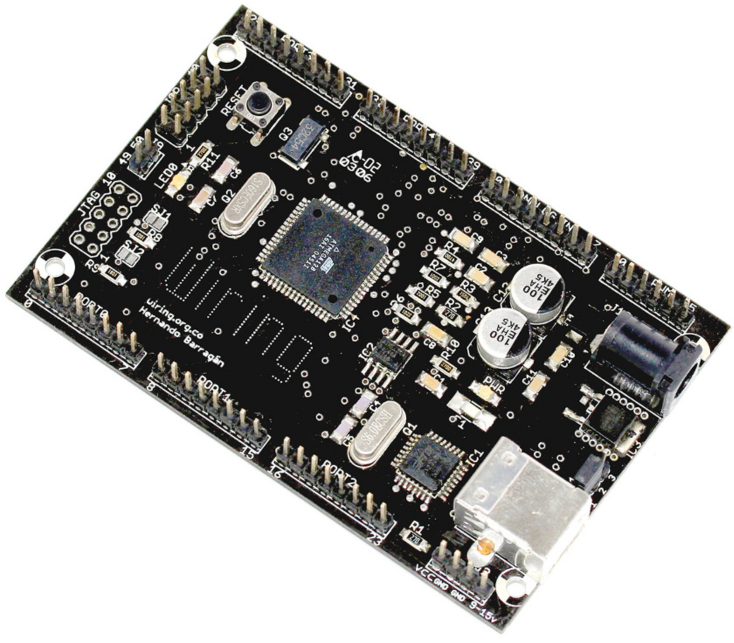

FIGURE 1-2: An early Wiring board.

Arduino also kept the naming convention from Processing, calling 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 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. You'll 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 and other earlier microcontrollers. It was not uncommon for students, 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. You can also change the microcontroller chip on an Arduino; if the chip becomes damaged, you can replace just it rather than the entire board.

Another important difference between Arduino and other microcontroller boards is the cost. Back in 2006, another popular microcontroller, the Basic Stamp, cost nearly four times as much ($119) as an Arduino ($32). Today, an Arduino Uno costs just $22.

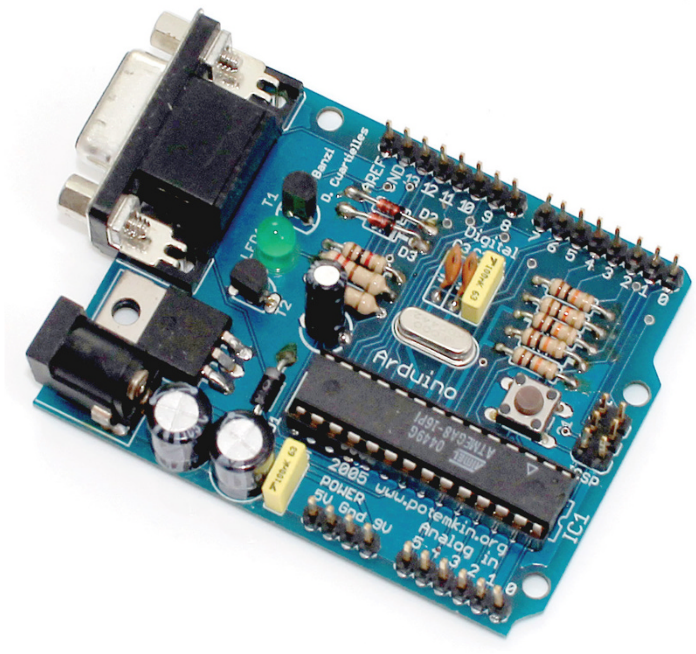

FIGURE 1-3: The original Arduino Serial board.

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 $42, 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 12, 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 (Global Positioning System) receiver, a mobile phone, or even a Geiger counter, to name just a few.

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 is a technique for experimenting with systems using wires. The earliest popular example of patching is in phone switchboards. For an operator to put you through to another line, he or she had to physically attach a cable.

This technique was also popular for synthesizing music, such as with the Moog synthesizer. When an electronic instrument generates a sound, it's 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, the musician proceeds largely through trial and error. But the simple interface means that this process is extremely quick and requires little preparation to get going.

Hacking

Hacking is a term that typically refers to the subversive use of technology. 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 programs contain keyboard shortcuts, and most PDF viewers move to the next page in a slideshow when the user presses the spacebar. If you know this, you ideally want a keyboard with only a spacebar.

Today's keyboards have a small circuit board, a bit smaller than a credit card (see Figure 1-4), containing lots of contacts that are connected when you press different keys. If you can find the correct combination, you can connect two contacts by using a pushbutton. Now every time you press 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 "Hacking Other Hardware" 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.

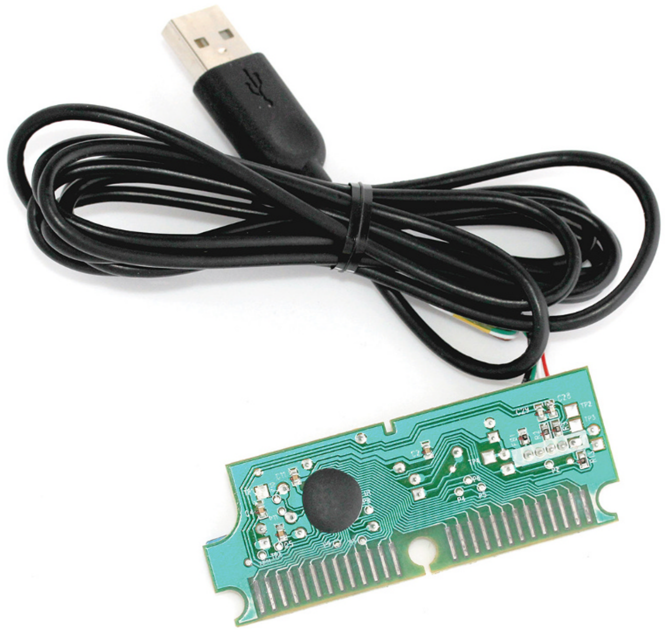

FIGURE 1-4: The insides of a keyboard, ready to be hacked.

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 a combination that produces a pleasing result, you can 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!

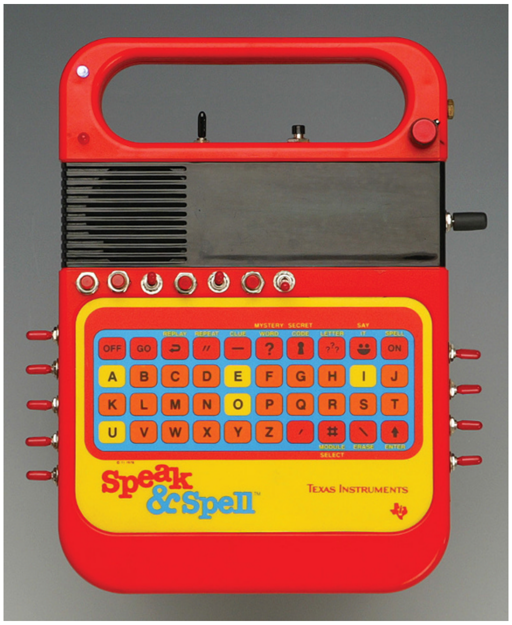
FIGURE 1-5: A Modified Toy Orchestra Speak & Spell after circuit bending.

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 kits are good at their specific job, but they can't do much else.

This is where microcontrollers come in. When used with analog circuitry, microcontrollers can give that circuitry a more advanced behavior. They can also be reprogrammed to perform different functions as needed. Your Arduino is designed around one of these microcontrollers, and 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 other electronic inputs and outputs to either sense or affect things in its environment.