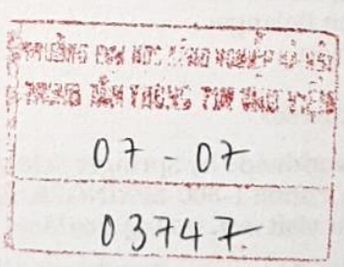
TECHNOLOGY IN ACTION™

Beginning Arduino Programming

Writing Code for the Most Popular Microcontroller Board in the World

Brian Evans

Beginning Arduino Programming

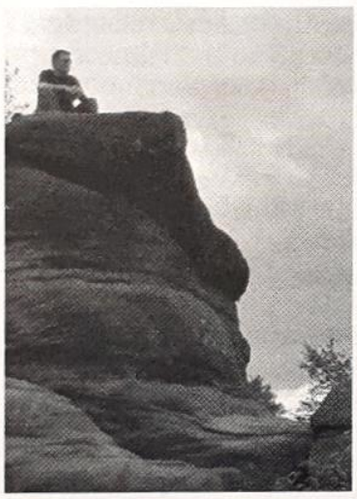


Brian Evans

Apress®

Contents

About the Author



Brian Evans is an artist working in electronic media and Assistant Professor at Metropolitan State College of Denver, where he teaches multidisciplinary courses in art and design on topics that include spatial media, electronics, and 3D fabrication. Many of his classes use open-source hardware, including MakerBot or RepRap 3D printers and the Arduino electronics platform, in the creation of new works in art and design.

His work has been shown at the Los Angeles Municipal Art Gallery at Barnsdall Park, the Orange County Center for Contemporary Art, and the University Art Museum at California State University, Long Beach. Evans was a resident and contributor to the Grounding Open Source Hardware residency and summit at the Banff New Media Institute in Alberta, Canada, in 2009 and contributor to the Open Hardware Summit in New York, in 2011. He received an MFA at California State University, Long Beach, in 2008 and a BFA at Arizona State University in 2005.

Introduction

This book will help you to develop working source code for the Arduino microcontroller. In these pages, we will primarily concern ourselves with the software aspect of physical computing—designing code to work with physical objects that exhibit behavior or interactivity through software. Starting with the basic context of the Arduino platform to getting up and running with our first code, we will discuss the structure and syntax of Arduino's C-based programming language, looking at variables, control structures, arrays, and memory. This book will then go into many of the functions unique to Arduino development for controlling digital and analog input and output, timing, randomness, writing functions, and using many of the Arduino libraries for working with different kinds of hardware and communication protocols.

Arduino, like Processing before it, adopted the idea of a code sketchbook. We will carry on this metaphor as we talk about the process of sketching in code as an intuitive method for quickly testing out new ideas in code. Most of this book is written around this idea of developing programming skills through sketching. We will also provide some suggestions for new projects and hardware, new languages to try out, and ways to contribute back to the community. This book intentionally does not dwell too long on electronics theory, circuit design, hacking, or other specifically hardware-based practices, although we'll revisit the hardware side of things in our last chapter to provide a small foundation for physical computing.

This book in many ways picks up where the *Arduino Programming Notebook* left off, with even more in-depth discussions about the Arduino environment; simple, no-frills code samples; and clear, easy-to-read schematics and illustrations. The *Notebook*, a little PDF booklet, was my first experience writing about the Arduino and was never meant to be more than a brief guide for my students when I first introduced a class of 15 college art and design majors to the Arduino in 2007. Best laid plans and all, this little booklet has now been translated into Spanish, Russian, and Dutch (that I know of), is hosted in so many different places that it is impossible to keep track of, and it's been used in workshops and classes around the world. I haven't updated the *Notebook* over the last few years, and in all honesty I am not entirely sure what to do with it now, so hopefully this new book will fill a void and find a similar, widespread adoption that the little booklet has enjoyed all these years.

Who This Book is For

This book is written for the primary audience of the Arduino platform: artists, designers, students, tinkerers, and the makers of things. While you might have some programming experience that you want to bring to the Arduino platform, we will assume no prior knowledge of writing code. With that said, a healthy familiarity of the computer is helpful, as is the willingness and inquisitive curiosity to look beyond this book for certain answers.

The majority of Arduino users just want to get things done and often don't care about the little details—they just want their projects to work. I understand this, as I am one of those people. I first discovered programmable microcontrollers when I was an art student, and at the time, art school was not generally the most conducive environment for learning how to write code and wire up motors—at

least it wasn't before the Arduino came along. Likewise, I was never one for a love of mathematics, which thankfully is not a prerequisite to deeply enjoy the process of writing code.

Reading This Book

Our process in each chapter will be to focus on some fundamental projects that build on the primary concepts presented in that chapter. For each project, we will begin with a project description and discuss the specific hardware needed for that project. We will also provide diagrams and illustrations for making these simple circuits and interfacing them to the Arduino board. As you read through each project, you should take notes and write in the margins—we won't be offended. Experiment, try new things, and see what happens.

The projects demonstrated in this book are meant to be prototypes, or fundamental proof-of-concept designs for a new device. We will adhere to a degree of minimalism, keeping to simple and easily obtainable hardware that supports the development of sophisticated written code. Once you have built the prototype, it can be incorporated into a final project later. We won't actually be doing that here so that we can focus on actually writing and developing code. Our examples will borrow and build on each other throughout the book, revisiting past examples when we need to as our understanding of writing code develops.

The intent with our code samples is to write compartmentalized or modular code wherever possible to allow for easy adaptability and future development. We will spend a lot of time developing our coding skills so that when it comes time to develop a new project independently, you will know where to begin. The sketches are meant to be fluid—you are encouraged to hack them—changing values, timing, pin assignments, ranges, and so on—until it no longer works. Then try to fix it. We will stick to a particular style of writing code in our samples, although we urge you to develop your own writing style that reflects the way you think and the way you want to see your code.

Wiring up the circuits for our projects is as simple and straightforward as possible, with little to no understanding of electronics necessary. As a way to reconnect our discussions of programming to the physical electronics used throughout the book, Chapter 12 will provide a brief review of some basic electronics, including how circuits work, reading schematics, and an introduction to soldering. If you find that you are struggling with hooking up the projects in the earlier chapters, you might want to jump to Chapter 12 for a refresher. Otherwise, this chapter will serve as a good summary that could help answer some questions you might not even know you had. While this might at first seem a little backwards, it has worked pretty well in my classes over the last few years.

Arduino 1.0

At the time of this writing, the Arduino developers are hard at work on a more stable, more efficient, and generally improved version of the Arduino software called Arduino 1.0. The final release version of Arduino 1.0 should be available right about the same time that this book is published. This is important because in the process of making things better, some things had to be broken. This means that some older code written under the alpha release of the Arduino software will no longer work on Arduino 1.0.

Conversely, the code in this book and images of the Arduino development environment have all been prepared using a beta release of Arduino 1.0 (`http://code.google.com/p/arduino/wiki/Arduino1`), so images of the Arduino software may appear different from the final version, some of the code in this book may not work on older versions of the software, and still other features of 1.0 were not yet fully implemented—so I couldn't write about them. There may also be other growing pains with this upgrade that we are not fully aware of at this time, so if an unusual problem crops up, then you might want to blame 1.0 and start there to figure out what's wrong.

Conventions

We will use several conventions in this book, including `fixed width fonts` in line to denote specific code examples, **bold text** highlights new concepts or definitions, and anything with a parenthesis after it—as in `setup()` or `loop()`—will denote something called a function. Anytime there is a block of fixed-width font separated from the main text, it is a multiline code example, as in the following:

```
// this is a mulitline
// code example
```

Note Occasionally there will be areas separated as this sentence is, as a side note, general tip, or caution about something you will want to pay careful attention to.

Downloading the Code

The source code for this book is available from the Apress web site (`www.apress.com`) in the Source Code / Downloads section. If you are publishing examples that use code from this book, using attribution that includes the title, author, publisher, year, and ISBN is generally a nice thing to do.