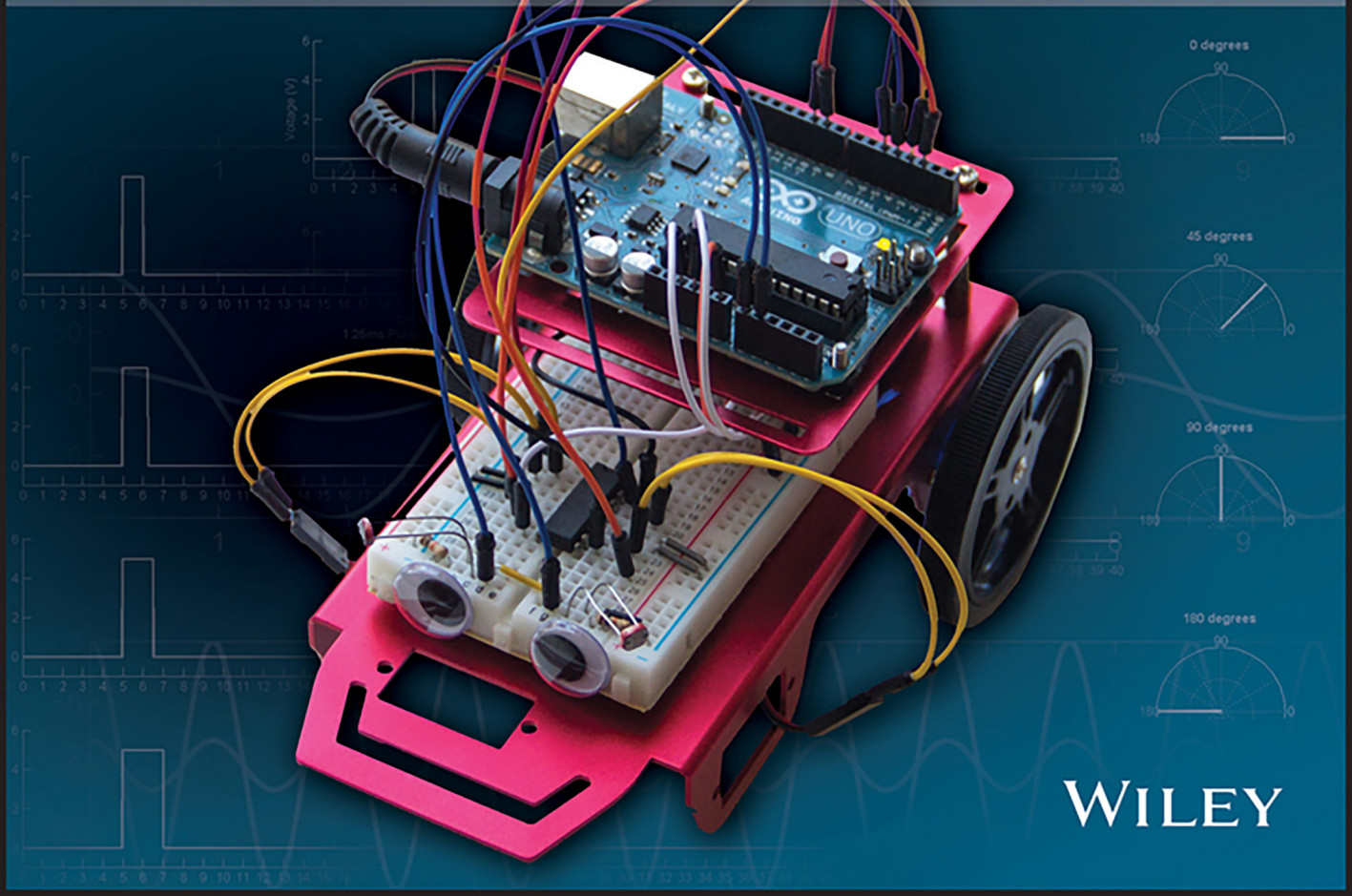
SECOND EDITION

New color photographs and diagrams

JEREMY BLUM

EXPLORING ARDUINO®

TOOLS AND TECHNIQUES FOR ENGINEERING WIZARDRY



WILEY

EXPLORING ARDUINO®

EXPLORING ARDUINO®

Tools and Techniques for Engineering Wizardry

Second Edition

Jeremy Blum

WILEY

Exploring Arduino®: Tools and Techniques for Engineering Wizardry

Published by
John Wiley & Sons, Inc.
10475 Crosspoint Boulevard
Indianapolis, IN 46256
www.wiley.com

Published simultaneously in Canada

ISBN: 978-1-119-40537-5
ISBN: 978-1-119-40535-1 (ebk)
ISBN: 978-1-119-40530-6 (ebk)

Manufactured in the United States of America

10 9 8 7 6 5 4 3 2 1

Library of Congress Control Number: 2019948860

To Leah, for helping me to see every challenge as an opportunity.

About the Author

Jeremy Blum is currently the director of engineering at Shaper (`shapertools.com`), where he is using computer vision to reinvent the way people use handheld power tools. Prior to joining Shaper, Jeremy was a lead electrical architect/engineer for confidential products at Google [x], including Google Glass.

Jeremy received his master's and bachelor's degrees in Electrical and Computer Engineering from Cornell University. At Cornell, he co-founded and led Cornell University Sustainable Design, he launched a first-of-its-kind entrepreneurial co-working space for students, and he conducted robotics and machine learning research.

Jeremy has designed prosthetic hands, fiber-optic LED lighting systems, home-automation systems, 3D printers and scanners, self-assembling robots, wearable computing platforms, augmented reality devices, and learning robots. His work has been featured in international conferences, peer-reviewed journals, and popular media outlets such as Discovery Channel, the *Wall Street Journal*, and *Popular Science* magazine. *Forbes* magazine named him to their annual "30 Under 30" list in recognition of his work that has "helped America make things and get stuff done." He is the co-author of several patents in the fields of wearable computing and augmented reality fabrication.

When not building products, Jeremy is teaching. His written and video tutorials have been utilized by millions of people to learn electrical engineering and embedded software design. His book, *Exploring Arduino*, has been translated into multiple languages and is used as an engineering textbook around the world, including at his alma mater, Cornell. Jeremy's passion is using engineering to improve people's lives, and giving people the tools they need to do the same. You can learn more about Jeremy at his website, `jeremyblum.com`.

About the Technical Editor

Dr. Derek Molloy is an associate professor in the Faculty of Engineering and Computing's School of Electronic Engineering at Dublin City University, Ireland. He lectures at undergraduate and postgraduate levels in object-oriented programming with embedded systems, digital and analog electronics, and connected embedded systems. His research contributions have largely been in the fields of computer and machine vision, embedded systems, 3D graphics/visualization, and e-learning. Derek produces

a popular YouTube video series that has introduced millions of people to embedded Linux and digital electronics topics. In 2013, he launched a personal web/blog site that is visited by thousands of people every day and that integrates his YouTube videos with support materials, source code, and user discussion. He has published other books in this Wiley mini-series: *Exploring BeagleBone* in 2015, followed in 2016 by *Exploring Raspberry Pi*. The second edition of *Exploring BeagleBone* was released earlier this year. You can learn more about Derek, his work, and his other publications at his personal website, `derekmolloy.ie`.

Acknowledgments

In the several years since the first edition of this book was released, I've received so many notes from readers who have told me about the many ways that they've learned from *Exploring Arduino*. I've also received plenty of constructive criticism—little things that I can adjust to improve the book's utility. I've taken all these comments to heart and have tracked them carefully over the past few years. It is my intention to make this second edition even more useful than the first, while still maintaining the approachability that many readers told me that they appreciated. So, THANK YOU to everybody who has given me feedback about the first edition of *Exploring Arduino*!

Second, I must extend my thanks again to Wiley. They've been amazing partners through this journey, and I'm glad to have them to continue to see this book through to a second edition. In particular, I'd like to thank Jim Minatel, Adaobi Obi Tulton, Dr. Derek Molloy, Marylouise Wiack, and Athiyappan Lalith Kumar.

Thanks also to the wonderful folks at Adafruit, who have collaborated with me on ensuring that parts kits are easy to obtain for this book. Adafruit contributes heavily to the open source hardware and software communities, and I certainly would not be the engineer that I am today without their excellent products and guides.

Back when I wrote the first edition of *Exploring Arduino*, I was still getting my master's degree. I've long since graduated, but now I've got my work at Shaper to focus on. I owe a big thanks to all my co-workers both at Shaper and at Google (my previous employer) for always encouraging me, and for building awesome hardware with me!

I want to give a special shout-out to my professors at Cornell, especially Professor François Guimbretière, who taught the course where I was first introduced to Arduino. He has since used the first edition of this book as a textbook for that course, and it makes me so happy to know that I've been able to give back to Cornell in that capacity.

Finally, I want to thank my parents, my brother, my wife, and my friends for putting up with me, and for always encouraging me. I feel so fortunate to have such wonderful people in my life.

Contents at a Glance

Contents

Figure Credits

Introduction

When the first edition of this book came out in 2013, I opened it with the following greeting:

You have excellent timing. As I often like to say, "We're living in the future."

I think I backed myself into a corner with that introduction, because if 2013 was "the future," then I'm not quite sure what to call the present! The *far future*? The *future-future*? My point is, the march of progress has been swift, and the possibilities for what you can do with even a cursory knowledge of embedded electronics and software continue to expand every day.

Since the first edition of this book was released, electronics and software have continued to become increasingly accessible with every passing day. In 2013, I was hesitant to include a chapter about connecting your hardware projects to the internet because the process for doing so was still quite fussy. The "Internet of Things" (IoT) was just an emerging nerdy buzzword in 2013. Now, it's a key part of the global vernacular. It seems like every product for sale nowadays contains a microcontroller. Everything is "smart" and most of those things also feature phone or web connectivity. I bet you didn't think you'd be buying a Bluetooth-enabled toothbrush back when "Bluetooth" just referred to people talking to themselves through their wireless cellphone headsets.

Considering all this, I felt it was time to release a new edition of *Exploring Arduino*. This second edition expands upon everything that was covered in the first edition. It updates all the projects with new challenges and details, clarifies questions that people had from the first edition, and adds a plethora of new content, including a lot more details on wireless connectivity, new Arduino hardware, changes to the Arduino ecosystem and software, and more.

Why Arduino?

With the tools available to you today, many of which you'll learn about in this book, you have the opportunity and the ability to bend the physical world to your whim. Until very recently, it has not been possible for someone to pick up a microcontroller and use it to control their world within minutes. A *microcontroller* is a programmable integrated circuit (IC) that gives you the power to define the operation of complex mechanical, electrical, and software systems using relatively simple commands. The possibilities are endless, and the Arduino microcontroller platform will become your new favorite tool as you explore the world of electronics, programming, human-computer interaction,

art, control systems, and more. Throughout the course of this book, you'll use the Arduino to do everything from detecting motion to creating wireless control systems to communicating over the internet.

Whether you are completely new to any kind of engineering or are a seasoned veteran looking to get started with embedded systems design, the Arduino is a great place to start. Are you looking for a general reference for Arduino development? This book is perfect for you, too. It walks you through a number of separate projects, but you'll also find it easy to return to the book for code snippets, best practices, system schematics, and more. The electrical engineering, systems design, and programming practices that you'll learn while reading this book are widely applicable beyond the Arduino platform and will prepare you to take on an array of engineering projects, whether they use the Arduino or some other platform.

Who This Book Is For

This book is for Arduino enthusiasts of all experience levels. Chapters build upon each other, utilizing concepts and project components from previous chapters to develop more complex ideas. But don't worry. Whenever you face new, complex ideas, a cross-reference reminds you where you first encountered any relevant building-block concepts so that you can easily refresh your memory.

This book assumes that you have little or no previous experience working with programming or electrical engineering. Using feedback from readers of the first edition of this book, I've taken special care to be very detailed in my explanation of the more confusing topics you may encounter. To effectively support readers of various experience levels, the book features several optional sections and *sidebars*, or short excerpts, that explain a particular concept in greater detail. Although these sidebars are not necessary for you to gain a good understanding of how to use the Arduino, they do provide a closer look at technical topics for the more curious reader.

What You'll Learn in This Book

This book is not a recipe book. If you want to follow step-by-step instructions that tell you exactly how to build a particular project without actually explaining why you are doing what you are doing, this book is not for you. You can think of this book as an introduction to electrical engineering, computer science, product design, and high-level thinking using the Arduino as a vehicle to help you experience these concepts in a hands-on manner.

When building hardware components of the Arduino projects demonstrated in this book, you'll learn not just how to wire things together, but also how to read schematics,

why particular parts are used for particular functions, and how to read datasheets that will allow you to choose appropriate parts to build your own projects. When writing software, I provide complete program code, but you will first be stepped through several iterative processes to create the final program. This will help to reinforce specific program functions, good code-formatting practices, and algorithmic understanding.

This book will teach physics concepts, algorithms, digital design principles, and Arduino-specific programming concepts. It is my hope that working through the projects in this book will not just make you a well-versed Arduino developer, but also give you the skills you need to develop more-complex electrical systems, and to pursue engineering endeavors in other fields, and with different platforms.

Features Used in This Book

The following features and icons are used in this book to help draw your attention to some of the most important or useful information in the book:

WARNING Be sure to take heed when you see one of these asides. They appear when particular steps could cause damage to your electronics if performed incorrectly.

TIP These asides contain quick hints about how to perform the task at hand more easily and effectively.

NOTE These asides contain additional information that may be of importance to you, including links to videos and online material that will make it easier to follow along with the development of a particular project.

SAMPLE HEADING

These asides go into additional depth about the current topic or a related topic.

Getting the Parts

In preparing the projects outlined in this book, I've taken special care to use components that are readily available through a variety of retailers, both in the United States and internationally. I've also partnered with Adafruit (`adafruit.com`), a popular retailer

of hobbyist electrical components. You can purchase all the components required for completing the projects in this book from Adafruit. A convenient listing of Adafruit parts for each chapter is available at `exploringarduino.com/kits`.

At the beginning of each chapter, you'll find a detailed list of parts that you need to complete that chapter—all of these parts are available from many sources. The companion website for this book, `www.wiley.com/go/exploringarduino2e`, also provides links to multiple sources where you can find the parts for each chapter.

What You'll Need

In addition to the actual parts that you'll use to build your Arduino projects, there are a few other tools and materials that you'll need on your Arduino adventures. Most importantly, you'll need a computer that is compatible with the Arduino integrated development environment (IDE) (Mac OS X 10.7 Lion or newer, Windows XP or later, or a Linux distro). I will provide instructions for all operating systems when warranted.

Arduino now also has an entirely web-based editor, but this book will generally focus on the desktop IDE. All the instructions for the desktop software generally apply to the online IDE as well. The first version of this book was read by people all over the world, representing a wide range of internet speeds and reliability. To ensure that Arduino remains easily accessible to all, I'll mostly provide instructions that use the offline IDE, as constant internet access isn't always an option for everybody.

You may also want some additional tools that will be used throughout the book to debug and assemble hardware. These tools are not only necessary to complete the projects in this book. As you develop your electrical engineering skillset, they will come in handy for other projects, too. I recommend the following:

- A soldering iron and solder (Note: A few shields and microcontroller boards used in the final chapters of this book may be sold with some soldering required—this usually involves easy soldering of thru-hole pins to a circuit board.)
- A multimeter (This will be useful for debugging concepts within this book, but is not required.)
- A set of small screwdrivers
- Tweezers
- Wire cutters and wire strippers
- A hot glue gun
- A magnifying glass (Electronics are small, and sometimes it's necessary to read the tiny, laser-etched markings on integrated circuits in order to look up their datasheets online.)

Source Code and Digital Content

The primary companion site for this book is `exploringarduino.com`, and it is maintained by the author. You will find code downloads for each chapter on this site (along with videos, links, and other useful materials). Note that both 1st edition and 2nd edition content is available at this URL—ensure that you are visiting the pages for this edition of the book. Digital content for the first edition is located at `exploringarduino.com/content1/` . . . and digital content for the second edition of this book is located at `exploringarduino.com/content2/` . . . . The website clearly differentiates between content for the two editions of the book and is easy to navigate.

Wiley also maintains a repository of digital content that accompanies this book at `wiley.com/go/exploringarduino2e`. You can also search for the book at `wiley.com` by ISBN (the ISBN for this book is 9781119405375) to find links to book resources.

The code for this book is hosted on `GitHub.com` (a popular platform for sharing open source software). Throughout each chapter, you can find references to the names of code files as needed in listing titles and text. Each chapter's code packages will be linked from `exploringarduino.com` and `wiley.com`. You can download the code as a compressed ZIP archive from either source. After you download the code, just decompress it with an appropriate decompression tool—all operating systems have one built in. You can also pull code directly from this book's GitHub repository (which is linked from `exploringarduino.com`) if you are comfortable working with Git-based version control.

NOTE Because many books have similar titles, you may find it easiest to search by ISBN; this book's ISBN is 9781119405375.

NOTE Some URLs (especially the ones that I don't control) may change or be very long. To make it easier to type in long URLs that I may reference throughout the book, I will list a "shortened URL" using my personal domain name shortener: `blum.fyi`. For example, `blum.fyi/jarvis` redirects to a longer URL on my website about a project called "JARVIS."

Errata

We make every effort to ensure that there are no errors in the text or in the code. However, no one is perfect, and mistakes do occur. If you find an error in this book, such as a spelling mistake or faulty piece of code, we would be grateful for your feedback. By

sending in errata, you may save another reader hours of frustration, and at the same time, you can help us provide even higher-quality information.

To find the errata page for this book, go to `wiley.com/go/exploringarduino2e` and click the Errata link. On this page, you can view all errata that has been submitted for this book and posted by Wiley editors. I also review all errata reports and post errata notes to `exploringarduino.com` on each chapter page.

Supplementary Material and Support

During your adventures with your Arduino, you'll inevitably have questions and perhaps run into problems. One of the best aspects of using the Arduino is the excellent support community that you can find on the web. This extremely active base of Arduino users will readily help you along your journey. I maintain a list of updated resources for getting help with Arduino, electrical engineering, and embedded software on the Exploring Arduino Resources page:

`exploringarduino.com/resources`

I used to try to answer people's individual Arduino questions directly, but that's unfortunately no longer possible due to the sheer volume of questions that I receive through my website, Twitter, Facebook, YouTube, and other channels. I highly encourage you to seek help through the forums linked from the Resources page listed here. I can almost guarantee that their response times will be faster than mine.

What Is an Arduino?

The best part about the Arduino prototyping platform is that it's whatever you want it to be. The Arduino could be an automatic plant-watering control system. It could be a web server. It could even be a quadcopter autopilot.

The Arduino is a microcontroller development platform paired with an intuitive programming language that you develop using the Arduino integrated development environment. By equipping the Arduino with sensors, actuators, lights, speakers, add-on modules (called *shields*), and other integrated circuits, you can turn the Arduino into a programmable "brain" for just about any control system.

It's impossible to cover everything that the Arduino is capable of, because the possibilities are limited only by your imagination. Hence, this book serves as a guide to get you acquainted with the Arduino's functionality by executing several projects that will give you the skills you need to develop your own projects.

You'll learn more about the Arduino and the available variations of the board in Chapter 1, "Getting Started and Understanding the Arduino Landscape." If you're eager to know all the inner workings of the Arduino, you're in luck: It is completely open source, and all the schematics and documentation are freely available on the Arduino website. Appendix A, "Deciphering Datasheets and Schematics," covers some of the Arduino's technical specifications.

An Open Source Platform

If you're new to the world of open source, you are in for a treat. This book does not go into detail about the open source hardware movement, but it is worth knowing a bit about the ideologies that make working with the Arduino so wonderful. If you want a full rundown of what open source hardware is, check out the official definitions on the Open Source Hardware Association website (`blum.fyi/OSHW-Definition`).

Because the Arduino is open source hardware, all the design files, schematics, and source code are freely available to everybody. This means that you can more easily hack the Arduino to serve a very particular function, and also integrate the Arduino platform into your designs, make and sell Arduino clones, and use the Arduino software libraries in other projects. There are hundreds of Arduino derivative boards available (often with specialized functions added on to them).

The Arduino open source license also permits commercial reuse of their designs (so long as you don't utilize the Arduino trademark on your designs). So, if you use an Arduino to prototype an exciting project and you want to turn it into a commercial product, you can do that.

Be sure to respect the licenses of the source code and hardware that you use throughout this book. Some licenses require that you provide attribution to the original author when you publish a design based on their previous work. Others require that you always share improvements that you make under an equivalent license. This sharing helps the community grow, and leads to all the amazing online documentation and support that you'll often refer to during your Arduino adventures. All code examples that I've written for this book (unless otherwise specified) are licensed under the MIT License, enabling you to use them for anything you want.

Beyond This Book

You may already be familiar with my popular series of YouTube Arduino and electronics tutorials (`youtube.com/sciguy14`). I refer to them throughout this book as a way

to see more-detailed walkthroughs of the topics covered here. If you're curious about some of the remarkable things that you can do with clever combinations of electronics, microcontrollers, computer science, and creativity, check out my portfolio (`jeremyblum.com/portfolio`) for a sampling of projects. Like Arduino, most of what I do is released via open source licenses that allow you to easily duplicate my work for your own needs.

I'm anxious to hear about what you do with the skills you acquire from this book. I encourage you to share them with me and with the rest of the world (use the tag *#ExploringArduino* on social media). Good luck on your Arduino adventures!

Arduino Engineering Basics

1 Getting Started and Understanding the Arduino Landscape

What You'll Need for This Chapter:

Arduino Uno or Adafruit METRO 328

USB cable (Type A to B for Uno, Type A to Micro-B for METRO)

CODE AND DIGITAL CONTENT FOR THIS CHAPTER
Code downloads, videos, and other digital content for this chapter can be found at:
`exploringarduino.com/content2/ch1`

Code for this chapter can also be obtained from the Downloads tab on this book's Wiley web page:
`wiley.com/go/exploringarduino2e`

Whether you are a weekend warrior looking to learn something new, an aspiring electrical engineer, or a software developer looking to better understand the hardware that runs your code, getting your first Arduino project up and running is sure to be an energizing experience. The preface of this book should have already given you some perspective on the Arduino platform and its capabilities; now it's time to explore your options in the world of Arduino. In this chapter, you will examine the available hardware, learn about the programming environment and language, and get your first program up and running. Once you understand the functionality that the Arduino can provide, you'll write your first program and get the Arduino to blink!

NOTE To follow along with a video that introduces the Arduino platform, visit the Chapter 1 content page for the second edition of this book at `exploringarduino.com/content2/ch1`.

Exploring the Arduino Ecosystem

In your adventures with the Arduino, you'll depend on three main components for your projects:

- First-party or third-party Arduino boards
- External hardware (including both shields and manually created circuits, which you'll explore throughout this book)
- The Arduino integrated development environment, or Arduino IDE

All these system components work in tandem to enable you to accomplish just about anything with your Arduino.

You have a lot of options when it comes to Arduino development boards. Most of this book uses Arduino boards designed by Arduino. Some of the final chapters leverage Arduino-compatible hardware that is designed by third parties to add features like Bluetooth and Wi-Fi to the standard Arduino offerings. Many third-party Arduino boards are directly compatible with Arduino software, libraries, hardware, etc. Some of these boards are designed to be exact clones of official Arduino boards, while others add their own features or capabilities. All the boards used in this book are programmable via the same IDE. When relevant, this book will list caveats about using different boards for various projects. Most of the projects in this book use the Arduino Uno because it has become the de facto introductory board for learning Arduino. You can freely substitute the Adafruit METRO 328 board in places where the Uno is called for—it is functionally identical. You'll see it used in place of the Uno in some of the photos and videos that accompany this book. Most introductory tutorials that you'll find on the web use the Uno or a variant of it. If you do use the Adafruit METRO 328, you may need to install the drivers for it to be detected as an Arduino Uno on Windows. Download and run the installer from `blum.fyi/adafruit-windows-drivers`.

WARNING **Beware of Counterfeits.** Only buy Arduino boards and Arduino-compatible boards from reputable sources (such as those listed throughout this book). There are many companies that manufacture clones of popular Arduino boards with inferior components.

You will start by exploring the basic functionality that is found in every Arduino board. Then you will examine the differences between each modern board so that you can make an informed decision when choosing a board to use for your next project.

THE GREAT ARDUINO SCHISM AND REFORMATION

Before you jump into understanding the available options in the Arduino ecosystem, I need to talk about the elephant in the room: the Great Arduino Schism and Reformation (not an official name). Between the release of the first edition of this book and the release of the second edition, the people behind the Arduino hardware and software had a falling out. I won't go into the details here, or pick a side. Basically, Arduino split into two entities represented by two websites: `Arduino.cc` and `Arduino.org`. Each group started producing slightly different hardware offerings, forked the codebase, and made conflicting claims about which hardware was genuine. Thankfully, the two sides of this battle have since reconciled their differences and we're back to one Arduino again. Throughout this book, I'll generally talk about the hardware offerings from Arduino.cc, though by the time you get this book, the two Arduinos should be one again. If you'd like to learn more about this nerdy drama, `Hackaday.com` did a series of reports on it. You can read about the resolution at `blum.fyi/arduino-vs-arduino`.

Arduino Functionality

All Arduino boards have a few key capabilities and functions. Take a moment to examine the Arduino Uno shown in Figure 1-1; it will be your base configuration. These are some functional groups that you'll be concerning yourself with:

- **Microcontroller:** At the heart of every Arduino is a microcontroller. This is the brain of your Arduino.
- **Programming:** Programming interfaces enable you to load software onto your Arduino.
- **I/O:** Input/Output (I/O) circuitry is what enables your Arduino interface with sensors, actuators, etc.
- **Power:** There are a variety of ways to supply power to an Arduino. Most Arduino boards can automatically switch between power from multiple sources (such as USB and a battery).

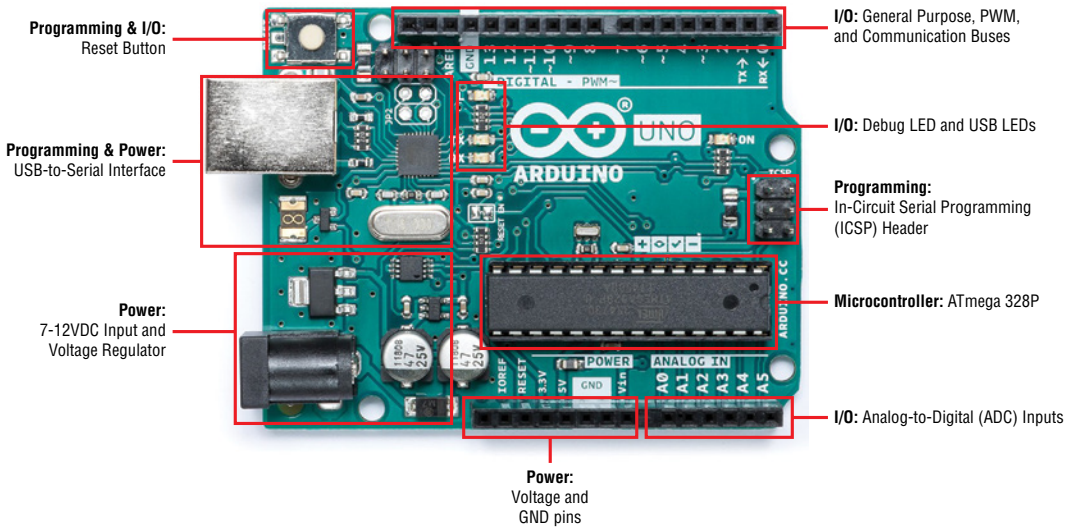


Figure 1-1: Arduino Uno components

Credit: Arduino, `arduino.cc`; annotations by author

The Microcontroller

At the heart of every Arduino is a microcontroller unit (MCU). All the original Arduino boards, including the Arduino Uno, used an 8-bit Atmel® ATmega microcontroller based on the AVR® architecture. The Arduino Uno in Figure 1-1, for example, uses an ATmega 328P. For most projects that you'll want to build, a simple 8-bit MCU like this one will be more than enough for your needs, so that's what you'll use throughout most of the exercises in this book.

NOTE MICROCHIP AND ATMEL Microchip, a chip manufacturer famous for making the PIC series of microcontrollers, recently acquired Atmel. ATmega chip production has continued under this new brand. Therefore, you may see Microchip and Atmel used interchangeably in reference to the manufacturer of ATmega microcontroller chips. The chips are functionally identical if they have the same part number.

NON-AVR MICROCONTROLLER ARCHITECTURES

But what about when you want to start doing crazy things like synthesizing music, running a web server, and driving massive LED displays? Though possible with clever and efficient programming on an 8-bit MCU, some of these needs are better served by faster and more capable processors.

As an answer to this, in recent years, Arduino has been expanding the range of available Arduino boards to include some that run on Intel (x86 and ARC—Argonaut RISC Core) architectures, and some that use Arm® (Advanced RISC Machine) architectures. The Arduino Due, for example, uses a 32-bit Arm Cortex®-M3 microprocessor. This Cortex processor is faster and contains more peripherals than the 8-bit AVR MCU, thus enabling the Due to do things like play music. Other new Arduino boards add functionality like built-in Wi-Fi and Bluetooth, which is also facilitated by faster and more capable processors. I'll touch on some of these boards later in this chapter, and you'll also get the opportunity to build projects with them later in this book.

You don't need to understand the intricacies of processor architectures to program or use an Arduino—it's all abstracted away for you. However, some people like to know what underlies their hardware projects. The following list will help clarify the buzzwords you just read:

- 8-bit architecture—An MCU architecture type where all addresses, integers, and other key data types are represented as 8-bit numbers.
- 32-bit architecture—An MCU architecture type where all the addresses, integers, and other key data types are represented as 32-bit numbers.

(Continued)

(Continued)

- Microchip (previously Atmel)—A company that makes microcontrollers. Microchip/Atmel makes both AVR MCUs and Arm processors. Most Arduinos use processors that are made by Microchip/Atmel.
- AVR—A microcontroller architecture developed by Atmel for their ATmega MCUs.
- Arm—A collection of 32/64-bit processor architectures developed by a company of the same name. Arm licenses its embedded architecture designs to be used by companies like Microchip and others.
- Cortex-M Series—Cortex M0, M3, and so on represent microprocessor Arm architectures.

The Arduino's microcontroller is responsible for holding all your compiled code and executing the commands you specify. The Arduino programming language gives you access to microcontroller peripherals, including analog-to-digital converters (ADCs), general-purpose input/output (GPIO or just I/O) pins, communication buses (including I^2C, SPI, UART, and others), and serial/USB interfaces. Utilizing copper wires etched into the Arduino's printed circuit board, all of this useful functionality is routed from the tiny pins on the microcontroller to accessible headers on the Arduino that you can plug wires or shields into. In the case of the Uno, a 16 MHz ceramic resonator or oscillating crystal is wired to the ATmega's clock pins, which serves as the reference by which all program commands execute. You can use the Reset button to restart the execution of your program. Most Arduino boards come with a debug LED already connected to pin 13, which enables you to run your first program (blinking an LED) without connecting any additional circuitry.

Programming Interfaces

Ordinarily, microcontroller programs are written in C or assembly, and programmed via the In-Circuit Serial Programming™ (ICSP™) interface using a dedicated programmer (see Figure 1-2). Perhaps the most important characteristic of an Arduino is that you can program it directly using only an ordinary USB cable. This functionality is made possible by the Arduino bootloader. The bootloader is loaded onto the microcontroller at the factory (using the ICSP header), which allows a serial USART (Universal Synchronous/Asynchronous Receiver/Transmitter) to load your program on the Arduino without using a separate programmer. (You can learn more about how the bootloader functions in the sidebar, "The Arduino Bootloader and Firmware Setup.")

In the case of the Arduino Uno and Mega 2560, a secondary microcontroller (an ATmega16U2 or ATmega8U2, depending on your revision) serves as an interface

between a USB cable and the serial USART pins on the main microcontroller. In the Adafruit METRO 328, a Silicon Labs bridge chip is used in place of the ATmega 16U2, but its function is equivalent. The Arduino Leonardo, which uses an ATmega32U4 as the main microcontroller, has USB incorporated, so a secondary microcontroller is not needed. The Arduino M0 uses a Cortex M0 that also includes USB functionality, so it doesn't need a secondary USB chip. In older Arduino boards, an FTDI brand USB-to-serial chip was used as the interface between the ATmega's serial USART port and a USB connection. It's still a popular solution when creating your own Arduino-compatible product.

Figure 1-2: AVRISP mkII programmer

Credit: © Microchip Technology Incorporated. Used with permission.

Input/Output: GPIO, ADCs, and Communication Busses

The part of the Arduino that you'll care most about during your projects is the general-purpose Input/Output (GPIO) and ADC pins. All of these pins can be individually addressed via the programs you'll write. These pins can serve as digital inputs and outputs. The ADC pins can also act as analog inputs that can measure voltages between 0V and 5V (usually from sensors). Many of these pins are also multiplexed to serve special functions, which you will explore later in this book. These special functions include various communication interfaces, serial interfaces, pulse-width-modulated outputs, and external interrupts.

Power

For most of your projects, you will simply use the 5V power that is provided over your USB cable. However, when you're ready to untether your project from a computer, you have other power options. Most Arduinos can accept between 6V and 20V (7V to 12V is the recommended voltage supply range) via the direct current (DC) barrel jack connector,

or into the VIN pin. Some Arduinos operate at 5V logic levels, and others operate at 3.3V logic levels. For 5V Arduinos, like the Uno, the power is configured as follows:

- 5V is used for all the logic on the Uno board. In other words, when you toggle a digital I/O pin, you are toggling it between 5V and 0V.
- 3.3V is broken out to a pin to accommodate 3.3V shields and external circuitry.

For most projects in this book, you can generally assume the use of a 5V Arduino, unless I explicitly specify otherwise.

THE ARDUINO BOOTLOADER AND FIRMWARE SETUP

A *bootloader* is a chunk of code that lives in a reserved space in the program memory of the Arduino's main MCU. In general, AVR microcontrollers are programmed with an ICSP, which talks to the microcontroller via a Serial Peripheral Interface (SPI). Programming via this method is straightforward, but necessitates the user having a hardware programmer such as an STK500 or an AVRISP mkII (see Figure 1-2).

When you first boot the Arduino board, it enters the bootloader, which runs for a few seconds. If it receives a programming command from the IDE over the MCU's UART (serial interface) in that time period, it loads the program that you are sending it into the rest of the MCU's program memory. If it does not receive a programming command, it starts running your most recently uploaded sketch, which resides in the rest of the program memory.

When you send an "upload" command from the Arduino IDE, it instructs the USB-to-serial chip (an ATmega 16U2 or 8U2 in the case of the Arduino Uno) to reset the main MCU, thus forcing it into bootloader mode. Then, your computer immediately begins to send the program contents, which the MCU is ready to receive over its UART connection (facilitated by the USB-to-serial converter).

Bootloaders are great because they enable simple programming via USB with no external hardware. However, they do have two downsides:

- They take up valuable program space. If you have written a complicated sketch, the approximately 2 KB of space taken up by the bootloader might be really valuable.
- Using a bootloader means that your program will always be delayed by a few seconds at bootup as the bootloader checks for a programming request.

If you have a programmer (or another Arduino that can be programmed to act as a programmer), you can remove the bootloader from your ATmega and program it directly by connecting your programmer to the ICSP header and using the *Sketch ➢ Upload Using Programmer* command from within the IDE.

Arduino Boards

This book cannot possibly cover all the available Arduino boards; there are many, and manufacturers are constantly releasing new ones with various features. I will focus on a subset of the most commonly used Arduino boards. The following section highlights some of the features in these boards.

The Uno (see Figure 1-3) is the flagship introductory-level Arduino and will be used heavily in this book. It uses an ATmega328P as the main MCU.

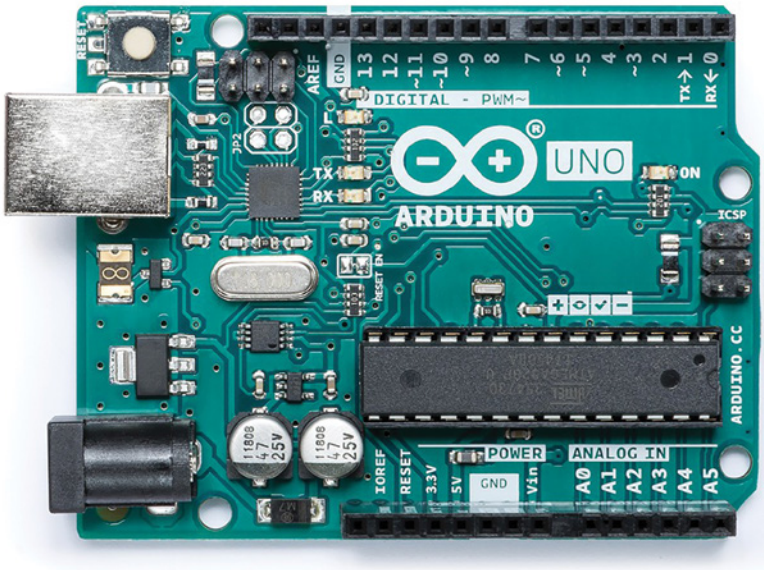


Figure 1-3: The Arduino Uno

Credit: Arduino, arduino.cc

The Mega 2560 (see Figure 1-4) employs an Microchip/Atmel ATmega2560 as the main MCU, which has 54 general I/Os to enable you to interface with many more devices. Think of the Mega as a supercharged version of the Uno—it's faster, has more memory, exposes more ADC channels, and has four hardware serial interfaces (unlike the one serial interface found on the Uno). It costs approximately 50 percent more than the Uno.

The Arduino Leonardo and Arduino Micro (see Figure 1-5 and Figure 1-6) both use the ATmega32U4 as the main microcontroller, which has a USB interface built in. Therefore, they don't need a secondary MCU to perform the serial-to-USB conversion. This cuts down on the cost and enables you to do unique things like emulate a joystick or a keyboard instead of a simple serial device. You will learn how to use these features in Chapter 8, "Emulating USB Devices". The Micro is functionally identical to the Leonardo, but is a smaller form factor that is designed to be plugged into a solderless or soldered breadboard.

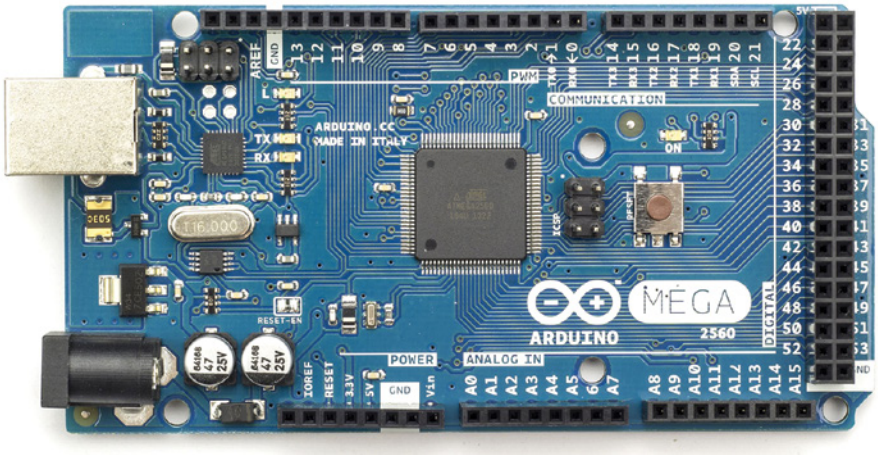


Figure 1-4: The Arduino Mega 2560

Credit: Arduino, arduino.cc

Figure 1-5: The Arduino Leonardo

Credit: Pololu Robotics & Electronics, pololu.com

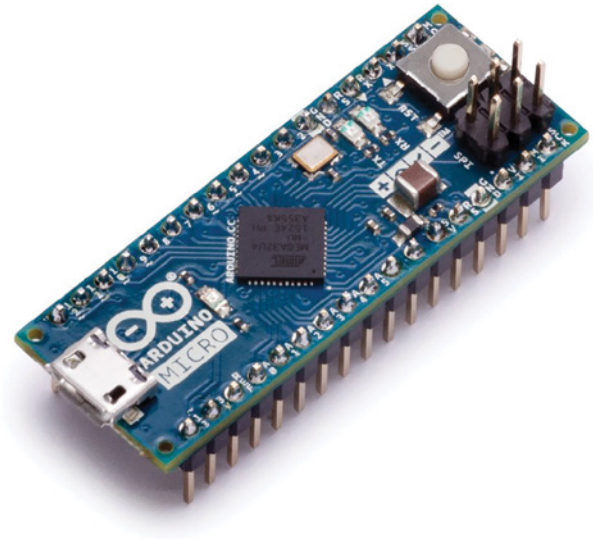


Figure 1-6: The Arduino Micro

Credit: Arduino, `arduino.cc`

The Due (see Figure 1-7) was Arduino's first foray into using the Arm microarchitecture. It uses a 32-bit Arm Cortex-M3 SAM3X. The Due offers higher-precision ADCs, selectable-resolution pulse-width modulation (PWM), digital-to-analog converters (DACs), a USB host connector, and an 84 MHz clock speed.

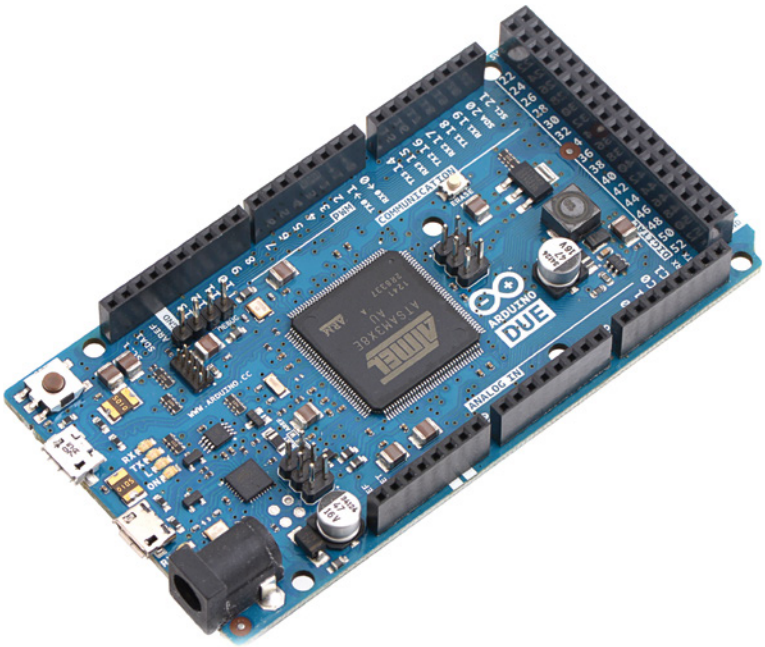


Figure 1-7: The Arduino Due

Credit: Pololu Robotics & Electronics, `pololu.com`

There are a variety of other Arduino boards as well. As you go through the chapters of this book, you may want to consider using some of those boards for more sophisticated projects that you dream up. As your needs get more specific, you may consider using some of the third-party Arduino-compatible boards that are available from companies like SparkFun, Adafruit, Pololu, and others. Because Arduino is an open-source platform, literally hundreds of clones and derivatives are available. The products and companies that I specifically call out in this book are ones that I have tested personally and can confirm work well. Use caution when buying generic Arduino clones online; read the reviews to find out if they work the way they are intended to. When in doubt, buy official Arduino products, or products from well-trusted companies like the ones I've mentioned.

When it comes to things like Bluetooth and Wi-Fi interoperability, the official Arduino offerings are a bit lacking at the time of this writing, so my recommended route is to check out the extremely well-documented Arduino-compatible Feather boards from `Adafruit.com`. You'll learn how to use these boards for building wireless Bluetooth and Wi-Fi projects in the final chapters of this book. Figure 1-8 shows a Bluetooth-enabled Arduino board from Adafruit.

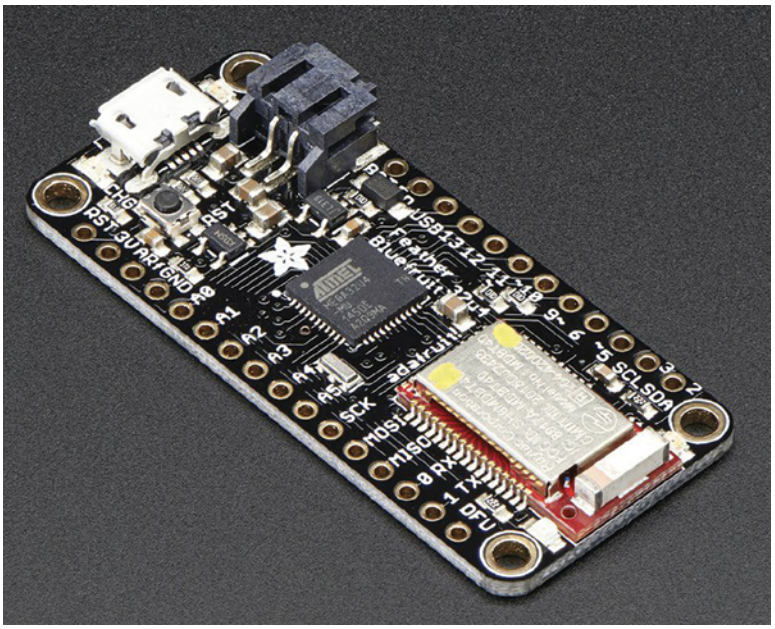


Figure 1-8: The Adafruit Feather 32u4 Bluefruit LE

Credit: Adafruit, `adafruit.com`

The skills you learn from this book will also easily transfer to a variety of Arduino-inspired platforms that use an Arduino-like programming interface coupled with their own hardware. The Photon (see Figure 1-9) from Particle is a great example of a Wi-Fi enabled microcontroller that uses a programming language inspired by the Arduino language. I use Particle Photons in my apartment to control my reading lamps and window shades from my phone.

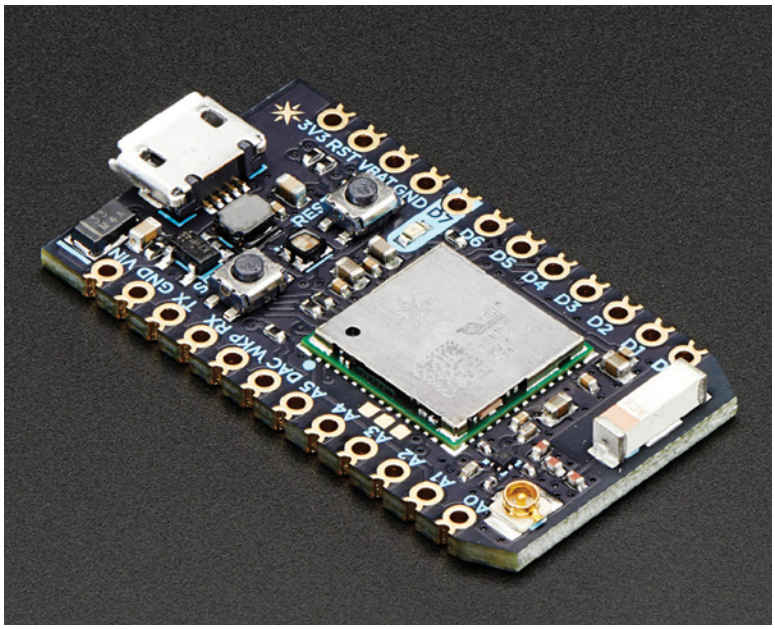


Figure 1-9: The Particle Photon

Credit: Adafruit, `adafruit.com`

Creating Your First Program

Now that you understand the hardware you'll be using throughout this book, you can install the software or access the Arduino web IDE and run your first program. Throughout this book, you'll generally use the downloaded desktop IDE. Start by downloading the Arduino software to your computer.

THE ARDUINO CLOUD IDE

The Arduino Cloud IDE is not explicitly used in this book's tutorials, but you can use it instead of the desktop IDE if you prefer. Simply set up an account at `arduino.cc`, and navigate to the editor, at `create.arduino.cc/editor`. Follow the instructions to install the plug-in and to start uploading code.

Downloading and Installing the Arduino IDE

Go to the Arduino website at `arduino.cc` and click the Software tab to display the Software page (see Figure 1-10). From there, you can download the newest version of the IDE that corresponds to your operating system.

Figure 1-10: The Arduino.cc page where you can download the Arduino IDE

If you're on Windows, download the installer instead of the Zip file. The installer will handle loading the necessary drivers for you. Run the installer and follow the onscreen directions. All the default options should be fine. For macOS or Linux, download the

compressed folder and extract it. On Mac OS X, simply drag the application into your Applications folder.

Running the IDE and Connecting to the Arduino

Now that you have the IDE downloaded and ready to run, you can connect the Arduino to your computer via USB, as shown in Figure 1-11. Linux and macOS machines usually install the drivers automatically.

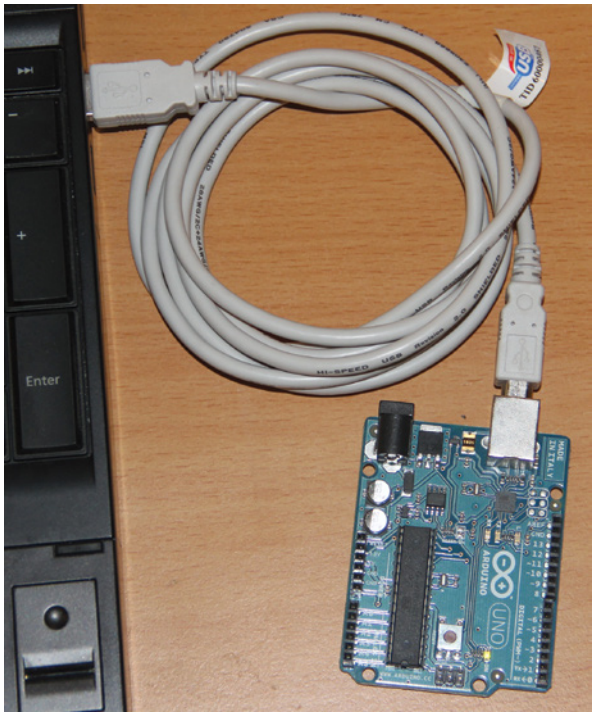


Figure 1-11: Arduino Uno connected to a computer via USB

NOTE Having trouble getting the IDE installed, or connecting to your board? Arduino.cc provides great troubleshooting instructions for all operating systems and Arduino hardware. Check out `blum.fyi/install-arduino`.

Now, launch the Arduino IDE. You're ready to load your first program onto your Arduino. To ensure that everything is working as expected, you'll load the Blink example program, which will blink the onboard LED. Most Arduinos have an onboard LED