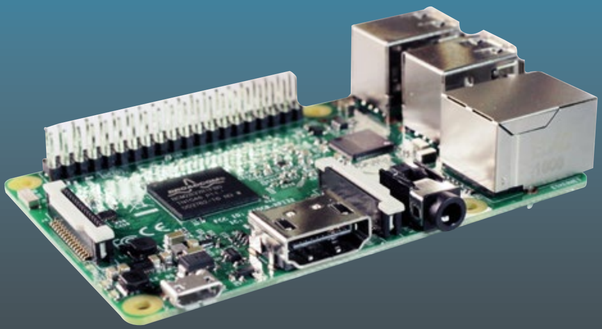
TECHNOLOGY IN ACTION™



Learn Raspberry Pi Programming with Python

Learn to Program on the World's Most Popular Tiny Computer

Second Edition

Wolfram Donat

Apress®

Learn Raspberry Pi Programming with Python

Learn to Program on the World's Most Popular Tiny Computer

Second Edition

Wolfram Donat

Apress®

Learn Raspberry Pi Programming with Python: Learn to Program on the World's Most Popular Tiny Computer

Wolfram Donat
Palmdale, California, USA

ISBN-13 (pbk): 978-1-4842-3768-7 ISBN-13 (electronic): 978-1-4842-3769-4
https://doi.org/10.1007/978-1-4842-3769-4

Library of Congress Control Number: 2018951226

Managing Director, Apress Media LLC: Welmoed Spahr
Acquisitions Editor: Aaron Black
Development Editor: James Markham
Coordinating Editor: Jessica Vakili

Cover designed by eStudioCalamar

Cover image designed by Freepik (www.freepik.com)

Distributed to the book trade worldwide by Springer Science+Business Media New York, 233 Spring Street, 6th Floor, New York, NY 10013. Phone 1-800-SPRINGER, fax (201) 348-4505, email orders-ny@springer-sbm.com, or visit www.springeronline.com. Apress Media, LLC is a California LLC and the sole member (owner) is Springer Science + Business Media Finance Inc (SSBM Finance Inc). SSBM Finance Inc is a **Delaware** corporation.

For information on translations, please email rights@apress.com or visit http://www.apress.com/rights-permissions.

Apress titles may be purchased in bulk for academic, corporate, or promotional use. eBook versions and licenses are also available for most titles. For more information, reference our Print and eBook Bulk Sales web page at http://www.apress.com/bulk-sales.

Any source code or other supplementary material referenced by the author in this book is available to readers on GitHub via the book's product page, located at www.apress.com/978-1-4842-3768-7. For more detailed information, please visit http://www.apress.com/source-code.

Printed on acid-free paper

To Becky and Reed

*Thank you for your patience and support
when I disappear for hours, days, and weeks
at a time to build all manner of off-the-wall
things and then write about them.*

Table of Contents

About the Author

Wolfram Donat is a writer, engineer, and maker who has been futzing with computers and electronics for longer than he cares to admit. He firmly believes that if something is worth doing, it's worth overdoing; everything needs a self-destruct button; and digital watches are still a pretty neat idea. He has a degree in computer engineering from the University of Alaska, and—despite several warnings—currently lives in Southern California with his wife, son, and a small menagerie.

About the Technical Reviewer

Massimo Nardone has more than 24 years of experience in security, web/mobile development, cloud, and IT architecture. His true IT passions are security and Android.

He has been programming and teaching how to program with Android, Perl, PHP, Java, VB, Python, C/C++, and MySQL for more than 20 years.

He holds a Master of Science degree in Computing Science from the University of Salerno, Italy.

He has worked as a project manager, software engineer, research engineer, chief security architect, information security manager, PCI/SCADA auditor, and senior lead IT security/cloud/SCADA architect over the years.

Technical skills include security, Android, cloud, Java, MySQL, Drupal, Cobol, Perl, web and mobile development, MongoDB, D3, Joomla, Couchbase, C/C++, WebGL, Python, Pro Rails, Django CMS, Jekyll, Scratch, and more.

He has previously worked as visiting lecturer and supervisor for exercises at the Networking Laboratory of the Helsinki University of Technology (Aalto University). He holds four international patents (PKI, SIP, SAML, and Proxy areas).

He currently works as Chief Information Security Office (CISO) for Cargotec Oyj and is a member of the ISACA Finland Chapter Board.

Acknowledgments

Even though writing is a solitary activity, no author writes in a void, and I would like to acknowledge those who have helped this book become a reality. Rebecca and Reed, your support—as always—is invaluable. Oliver makes sure the door to the office works. Chloe ensures that all objects coming out of the workshop or garage have evasive-maneuver capabilities. Smudge gives and receives emotional support. Doofus and Pericles supervise.

Couldn’t do it without you guys.

Preface

It is difficult to believe that it's been four years since I wrote the first edition of this book. In 2014, there was one version of the Raspberry Pi, a comparatively underpowered board with only one core in its ARM processor and only twenty GPIO pins to play around with. I was excited to order my first Pi, and I actually had to get on a waiting list to be on the second shipment list.

In addition, it seemed that every time you turned around, someone else was introducing a single-board computer (SBC) that was trying to appeal to the same niche that the Pi did—mainly hobbyists and makers who were ready to step up from the Arduino to something a bit more powerful.

The Pi resisted all attacks on its throne, however (not that it was ever interested in competing), and thrived. There are now seven models of the Pi: the model 1, the model 2, the model 2B, the model 3, the model 3B, the Pi Zero, and the Zero W. The Pi 3 is a computing powerhouse compared to the original model; its quad-core architecture lets it perform tasks like computer vision and machine learning, and overclocking it can give you speeds up to 1.5GHz, compared to the original's 700MHz. Meanwhile, the Zero and the Zero W have such a low price point ($5US and $10US, respectively) that I often have to field questions from readers like "Why should I use an Arduino? The Pi Zero is cheaper!"

And the Pi is not the only game in town. Depending on how much you're willing to spend, there are quite a few other SBCs that can be used for whatever project you've got in mind, ranging from the $30 BeagleBoard to the $550 NVidia Jetson TX2. I still like the Pi, however; it's the board that first got me started playing around with embedded computers and the hobby projects that you can do with them. It’s inexpensive, so when I burn it up or brick it (as I’ve done quite a few times) I can replace it without breaking the bank. And it’s still powerful enough for quite a lot of things.

Thanks for reading this new book with me. If you’re a fan of my original book, thanks for sticking with me and putting up with the several mistakes that made it through to publication, and if you’re a new reader and a new Pi user, welcome! I hope to use the following pages to introduce you to an exciting new world of projects and computing.

Introduction

In 2006, when Eben Upton and the other founders of the Raspberry Pi Foundation looked at the state of computer science (CS) programs at universities, they were dismayed. Such programs were being reduced to "CS 101: How to Operate Microsoft Word" and "CS 203: Optimize Your Facebook Page." Nobody, they realized, was learning how to *program* anymore, least of all before they entered college. So they hatched a plan—create a small, cheap computer that kids could learn to program on, like the Amigas, Spectrums, and Commodore 64s of yesteryear. They put an ARM processor on a board, gave it (eventually) 512MB of RAM and a VideoCore GPU, and allowed users to interface with it using a USB keyboard and mouse and an HDMI output. To make it easy to program, they designed it with Python in mind—a powerful, easy-to-learn language. And thus the Raspberry Pi was born.

I wrote my first program in BASIC on a Commodore VIC20, longer ago than I care to admit. At 5KB of RAM, it had less processing power than many of today's microcontrollers, but I was still able to write a simple maze game on it, saving my progress as I went along to a cassette-tape drive. In the years since, I've traversed my way through the different computing platforms, from Windows 3.1 to Macintosh OS 8 to Linux, my OS of choice. It had been a long time since I was truly excited by a computer; the Pi was a breath of fresh air in a stale computing environment. Not only was it small and cheap, but it was also easy to get it to interact with the physical world—a real boon for anybody like me who was interested in designing and building physical systems. So, when I heard about its release, I signed up for the shipment like about a trillion other hobbyists/hackers/engineers and waited impatiently for mine to be delivered. Then, I started building stuff with it and writing about it, and I never looked back.

If you've bought (or were gifted) a Pi, but aren't sure how to get started with it, this book is for you.

If you've got a Pi but aren't sure about what you can or want to do with it, this book is for you.

If you're even *considering* buying a Pi, for yourself or someone else, but haven't yet because you keep wondering "What's it good for?" or "Why not buy an Arduino?" then this book is *definitely* for you.

This book isn't meant to be a textbook on Python, nor is it an exhaustive exploration of the Raspberry Pi and everything it can do. But it *is* meant to be a fun getting-started guide to this nifty little computer, in all of its possible permutations. I hope that after working through this book you'll have an understanding of everything you can do with this little board when you mix it with some ingenuity and creativity.

If you want to work through the projects in order, feel free. If you'd rather skip around, doing only those projects that interest you, do that. Along the way, I hope you develop a familiarity with Python and Linux and the Pi that will enable you to continue on, building projects as you go, and perhaps inspiring others the way I hope to inspire you. Above all, I hope you enjoy the book and its projects. It was truly a blast to write. I always love hearing about your projects; you can reach me through the publisher or via Twitter: `@wolfram_donat`.

Happy computing!

CHAPTER 1

Introducing the Raspberry Pi

So, you've got yourself a Raspberry Pi mini-computer and are thinking to yourself: "Now what?" Maybe it was a gift. Maybe you'd heard about this "Raspberry Pie thingamabob" and decided to find out what all of the ruckus was about. Perhaps you're already experienced with computers, but not with Linux or Python. Maybe you're a Linux geek who's never made a servo move or lit up an LED with just a few lines of code and the correct hardware and software installed. Or maybe you're familiar with computers only to the point of checking your email and surfing the web but are eager to learn more. Or perhaps (one of my favorite scenarios) you're an educator who's interested in teaching the next generation about computers and programming and technology in general.

Whatever the case may be, welcome! You're about to join a club—not a particularly exclusive one, I'm afraid, as the cost of joining is only about $35 US and a spark of creativity, but a club nonetheless—populated by students and teachers and hobbyists and artists and engineers. As a member of this club, you'll be able to discuss *package managers*, *ARM processors*, and *dot config files* intelligently with whomever will listen to your babble. You'll become familiar with servos, LEDs, and cameras-on-a-chip. And, perhaps most important, you'll be able to connect to your new mini-computer, program it using one of many programming languages

© Wolfram Donat 2018
W. Donat, *Learn Raspberry Pi Programming with Python*,
https://doi.org/10.1007/978-1-4842-3769-4_1

(although this book deals solely with Python), build projects, and interface those projects with the Pi, enabling it to interact with the physical world and do some pretty cool things.

With this book, I hereby induct you into this club. Your experience doesn't matter, because I'll lead you step-by-step through the process of setting up your Pi so you can work with it with a minimum of headaches. I'll give you a solid background in Linux so you have an idea of what's going on behind the scenes, and I'll devote an entire chapter to Python, the deceptively powerful scripting language used by tech companies like Facebook, Google, and even NASA. I also plan to introduce you to the basic nuts and bolts of electronics, something that many tech-project books either gloss over or neglect completely. There are safety factors to consider (I've nearly had several small explosions as a result of shorting out LiPo batteries, for instance) as well as good building practices; you'll learn the difference between a good and a bad solder joint, how to avoid slicing off your finger with an X-ACTO knife, and the difference between a 40Ω and a 40KΩ resistor.

Of course, if you're already familiar with all of these introductory items, feel free to skip ahead to the good stuff—the projects. Most of them can be constructed in a weekend or so, and I've tried to keep the costs within reason as well. All are programmed in Python. At the beginning of each chapter, I'll give you a shopping list and suggestions on where to buy the parts, and then we'll start building. The projects don't necessarily build on each other, nor do you have to do them in order, though they do tend to increase in complexity from the first project to the last.

What sorts of projects can you do with a Pi? A better question might be what sorts of projects *can't* you do with a Pi. It's been used for everything from web servers to car computers (*carputers*) to cluster computing to embedded machine vision devices to CNC controllers . . . the list just goes on and on. I hope that after finishing this book you'll have some ideas of your own as well as the skills required to put those ideas into practice.

Whatever reason you have for picking up this book, your main objectives should be to have fun and to learn something. I hope I can help along the way!

The History of the Raspberry Pi

It may seem to some readers that the Raspberry Pi is new; there are a surprising number of people who have no idea what it is. Even now, seven years after the first Pi was produced, a large number of online articles begin with something along the lines of "The Raspberry Pi is a small, credit-card sized computer that hobbyists have begun using for . . ." This is in stark contrast to, say, the Arduino; most people who are up on current events have at least heard of it, even if they have no idea what it is or what it's used for, since it's been around since 2005 and has gained an immense, vocal following among hobbyists, geeks, and do-it-yourselfers around the world.

THE ARDUINO

For those who *don't* know, the Arduino is a microcontroller platform, available in many different form factors and sizes, mounted on a PCB that plugs easily into most computers' USB ports. It allows the user to program the onboard Atmega chip to do various things via a C-like programming language in programs called *sketches*. A typical Arduino sketch might look something like this:

```
#include <Servo.h>

void setup()
{
     myservo.attach(9);
}
```

```
void loop()
{
    myservo.write(95);
    delay(1000);
    myservo.write(150);
    delay(1000);
}
```

This program will move a connected servomotor (a small motor that can be controlled precisely via software) back and forth, with one-second delays between movements.

The Arduino is not as powerful as the Pi when it comes to computing power, but it's also a completely different animal, as it's a microcontroller, not a computer, so comparing them is a bit like comparing zebras and avocados. The two machines do, however, complement each other well, and I will discuss how to do that in Chapter 14.

As I said, the Raspberry Pi has been around for a few years—seven, to be exact. There are several different models available, with a new, improved version being released about once every other year.

The Pi's creators—Eben Upton, Rob Mullins, Jack Lang, and Alan Mycroft—first floated the idea of a cheap PC for teaching purposes in 2006. They were all based at the University of Cambridge in the United Kingdom, and they were concerned that the demise of cheap personal computers like the Commodore 64, the Amiga, and the Spectrum was adversely affecting young people's ability to program. With desktop and laptop machines costing hundreds or thousands of dollars, kids and teenagers were forbidden to practice programming on the family's main computer for fear of breaking it.

At the same time, Upton and the others realized that many university computer science curricula had been reduced to "Microsoft Word 101" and "How to Create a Web Page Using HTML." The four creators wanted to raise the programming knowledge bar of incoming students, and thus perhaps computer science and engineering courses would become a bit more robust and applicable to STEM fields in the real world.

Obviously, a cheaper computer was necessary. They played around with microcontrollers and various chips, breadboards, and PCBs (see Figure 1-1), but it wasn't until 2008 that the idea became more feasible. Chips were becoming smaller, cheaper, and more powerful, thanks to the explosion in mobile devices. These chips enabled them to plan a device that would be capable of supporting multimedia, not just command-line programming, which they felt was important in order to attract all ages of students. Young people were more likely to be interested in a media-capable device, and thus would be more likely to try programming on one.

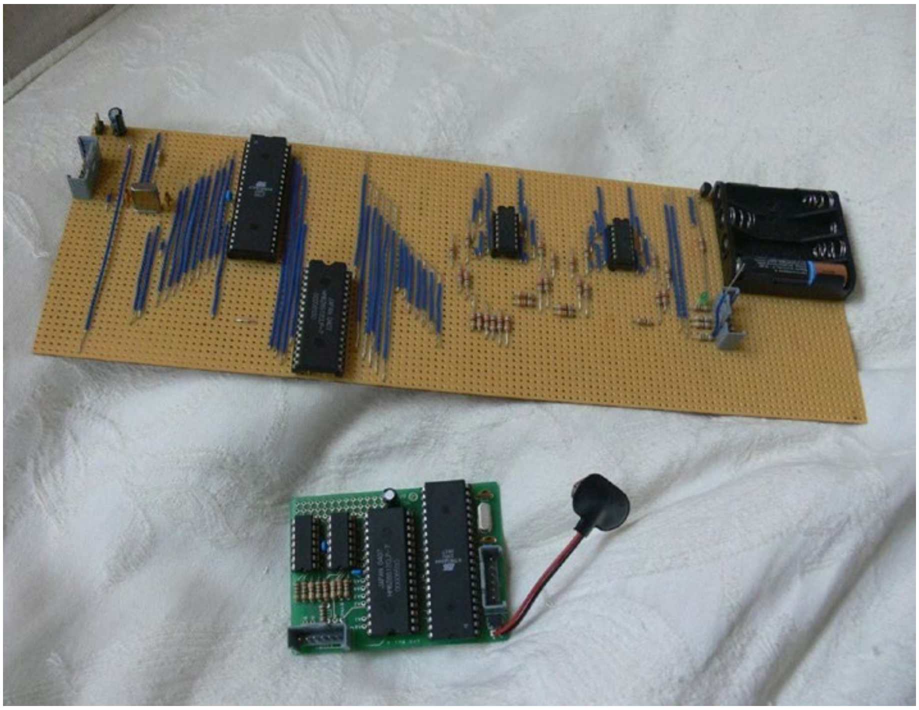


Figure 1-1. *Eben Upton's 2006 Raspberry Pi prototype (image ©Raspberry Pi Foundation)*

In 2008, the original four creators, along with Pete Lomas and David Braben, formed the Raspberry Pi Foundation (the Foundation), and three years later the first mass-produced Pi rolled off the assembly line.

Note The name *Raspberry Pi* is a nod to the number of microcomputers named after fruit in the early days, such as the Apple and the Tangerine, and the *Pi* comes from the Python scripting language, which has always been an integral part of the Pi's design.

Within a year, the Foundation had sold over one million units. The founding members have spoken many times about how they were dumbfounded by the explosive interest in their device. Their original goal of putting a cheap, programmable device in the hands of educators and their students has come to fruition, but the Pi has also become much more than that. Apparently, they were not the only people who were missing the ability to program on a cheaper machine; hobbyists around the world (including yours truly) flooded element14, Premier Farnell, and RS Electronics with orders—to the point that people who had pre-ordered their Pi had to wait up to six months for supply to catch up with demand. (As of this writing, one of the latest models of the Pi, the Pi Zero W, is still only available on a one-per-customer basis.) Many customers may have been current or former programmers, eager to play with a new, small, powerful computer. I, for instance, first learned to program in BASIC on the Commodore VIC-20, a computer with an impressive 5KB of RAM.

Aside from education, there are an almost infinite number of other uses for the Pi, as it states on the Raspberry Pi Foundation's *About Us* page:

> *We've had enormous interest, support, and help from the educational community, and we've been delighted and a little humbled by the number of enquiries from agencies and people far away from our original targets for the device. Developing countries are interested in the Raspberry Pi as productivity devices in areas that simply can't afford the power and hardware needed to run a traditional desktop PC; hospitals and museums have contacted us to find out about using the Raspberry Pi to drive display devices. Parents of severely disabled kids have talked to us about monitoring and accessibility applications; and there seem to be a million and one people out there with hot soldering irons who want to make a robot.*

Luckily, for the most part, supply has securely caught up with demand. There is no waiting period to buy a Pi anymore, and with the exception of the Zero W there is no longer a limit of one per customer. There are countless "hats" available (form-fitting aftermarket add-on boards with various capabilities), as well as a camera board and an official touchscreen display that both plug into the ports on the Pi. The founders have also actively encouraged other companies to copy their paradigm, and that has probably been largely responsible for the current number of small single-board computers available today.

Exploring the Pi

It is no longer possible to write just a single section that claims to exhaustively illustrate the Pi's built-in parts and design, as there are many different designs available. I will, however, keep this section small by addressing only the three most recent releases: the Pi version 3, the Zero, and the Zero W. As it happens, the Zero and the Zero W have almost identical setups, so we need only describe one of the two. The price point of all of these boards has remained low; the version 3 is currently about $35US, the Zero is about $5, and the Zero W is $10. On March 14, 2018, also known as Pi Day, the Raspberry Pi Foundation released an update to the Pi version 3, the 3 B+. This newer version offers a few upgrades to the original version 3, including dual-band WiFi, a slightly faster CPU (1.4GHz), and power-over-Ethernet (PoE) capabilities. As this version is still *very* new, as its form factor is almost identical to the original version 3, and as its upgrades won't affect any of the projects in this book, I won't mention it beyond this point.

The size of the Pi hasn't changed over the years; the Pi 3 measures the same as the Pi 1: 85.6mm x 56mm x 21mm. The Pi Zero and Zero W are a bit smaller: 30mm x 65mm x 3.5mm (not having USB and Ethernet ports makes a huge difference in thickness). The newest Pi is a bit heavier—45 grams versus the original's 31 grams—but luckily weight probably doesn't factor in when you're trying to fit the new Pi into your old case or project design.

Take a look at Figure 1-2, and I'll take you on a short clockwise tour of the board, starting with the GPIO pins.

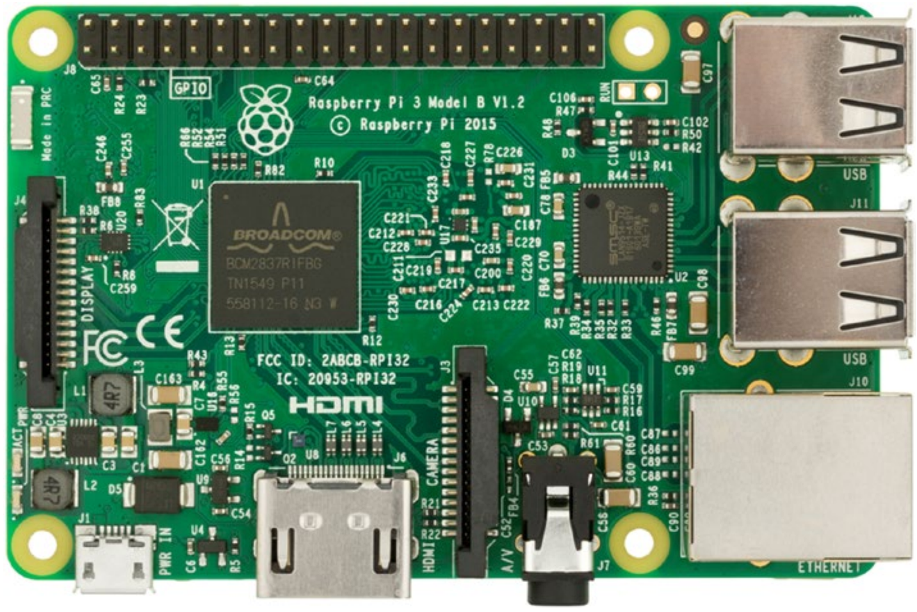


Figure 1-2. *The Raspberry Pi 3*

The GPIO Pins

As you can see in the figure, there's a lot packed onto the board's small space. You can see, running along the top, one of the biggest improvements from the Pi's early version to the current models: the increase from 26 to 40 GPIO (General Purpose Input/Output) pins. These pins allow you to connect the Pi to any number of physical extensions, from LEDs and servomotors to motor controllers and extension boards (often referred to as "hats"). With a normal desktop or laptop, interfacing with physical devices like those is nigh impossible, as the serial port has all but disappeared on newer devices, and not everybody is capable of writing low-level device drivers for the USB port. The Pi, however, comes with libraries pre-installed that allow you to access the pins using Python, C, or C++, and there are additional libraries (e.g., PiGPIO and ServoBlaster) available if you don't care for the preinstalled versions.