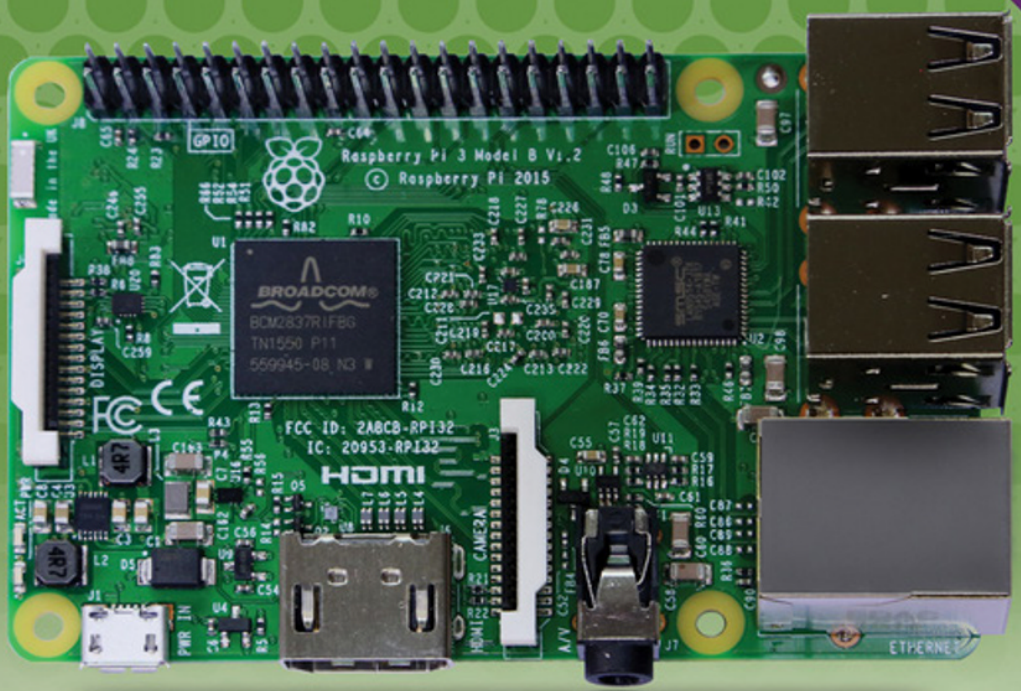
Fourth Edition
Updated for Raspberry Pi 3



Raspberry Pi®
User Guide

Eben Upton
Co-creator of the Raspberry Pi

Gareth Halfacree

WILEY

Raspberry Pi®
User Guide

Raspberry Pi® User Guide

4th Edition

Eben Upton and Gareth Halfacree

WILEY

This edition first published 2016

© 2016 Eben Upton and Gareth Halfacree

Registered office

John Wiley & Sons Ltd., The Atrium, Southern Gate, Chichester, West Sussex, PO19 8SQ, United Kingdom

For details of our global editorial offices, for customer services and for information about how to apply for permission to reuse the copyright material in this book please see our website at `www.wiley.com`.

The right of the authors to be identified as the authors of this work has been asserted in accordance with the Copyright, Designs and Patents Act 1988.

All rights reserved. No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, electronic, mechanical, photocopying, recording or otherwise, except as permitted by the UK Copyright, Designs and Patents Act 1988, without the prior permission of the publisher.

Wiley also publishes its books in a variety of electronic formats. Some content that appears in print may not be available in electronic books.

Designations used by companies to distinguish their products are often claimed as trademarks. All brand names and product names used in this book are trade names, service marks, trademarks or registered trademarks of their respective owners. The publisher is not associated with any product or vendor mentioned in this book. This publication is designed to provide accurate and authoritative information in regard to the subject matter covered. It is sold on the understanding that the publisher is not engaged in rendering professional services. If professional advice or other expert assistance is required, the services of a competent professional should be sought.

Library of Congress Control Number: 2016946656

Trademarks: Wiley and the Wiley logo are trademarks or registered trademarks of John Wiley & Sons, Inc. and/or its affiliates in the United States and/or other countries, and may not be used without written permission. Raspberry Pi and the Raspberry Pi logo are registered trademarks of the Raspberry Pi Foundation. All other trademarks are the property of their respective owners. John Wiley & Sons, Ltd. is not associated with any product or vendor mentioned in the book.

Google Drive™ is a registered trademark of Google™.

A catalogue record for this book is available from the British Library.

ISBN 978-1-119-26436-1 (Pbk); ISBN 978-1-119-26438-5 (ePDF); ISBN 978-1-119-92437-8 (ePub)

Set in 10 pt. Chaparral Pro by Indianapolis Composition Services

Printed simultaneously in Great Britain and the United States

Publisher's Acknowledgements

Some of the people who helped bring this book to market include the following:

Editorial and Production

VP - Professional Technology Strategy
Barry Pruett

Associate Director–Book Content Management
Martin Tribe

Executive Commissioning Editor
Jody Lefevere

Project Editor
John Sleeva

Technical Editor
Andrew Scheller

Manager of Content Development and Assembly
Mary Beth Wakefield

Editorial Assistant
Matthew Lowe

Marketing

Marketing Manager
Lorna Mein

Associate Marketing Manager
Carrie Sherrill

About the Authors

EBEN UPTON is a founder and trustee of the Raspberry Pi Foundation, and serves as its Executive Director. He is responsible for the overall software and hardware architecture of the Raspberry Pi, and for the Foundation's relationships with its key suppliers and customers. In an earlier life, he founded two successful mobile games and middleware companies, Ideaworks 3d Ltd. and Podfun Ltd., and held the post of Director of Studies for Computer Science at St John's College, Cambridge. He holds a BA, a PhD, and an MBA from the University of Cambridge.

In his day job, Eben works for Broadcom as an ASIC architect and general troublemaker.

GARETH HALFACREE is a freelance technology journalist and the co-author of the *Raspberry Pi User Guide* alongside project co-founder Eben Upton. Formerly a system administrator working in the education sector, Gareth's passion for open source projects has followed him from one career to another, and he can often be seen reviewing, documenting, or even contributing to projects including GNU/Linux, LibreOffice, Fritzing, and Arduino. He is also the creator of the Sleepduino and Burnduino open hardware projects, which extend the capabilities of the Arduino electronics prototyping system. A summary of his current work can be found at `http://freelance.halfacree.co.uk`.

About the Technical Editor

ANDREW SCHELLER is a freelance software developer based in Cambridge in the UK. He has worked on many projects for a variety of clients, including updating the Raspberry Pi NOOBS software to run on the Raspberry Pi 2. Prior to going freelance, he served as a tools programmer for Sony Computer Entertainment, working on a number of titles, including MediEvil Resurrection for the PlayStation Portable and 24: The Game for the PlayStation 2. While there he developed a localisation system which has become adopted as the standard by all Sony game studios worldwide. He spends some of his free time working on open source software, and has been using Linux for more than 20 years. He enjoys walking, cycling, and climbing, and holds a BSc (Hons) in Computer Science from Durham University.

For Liz, who made it all possible.

—Eben

For my father, the enthusiastic past, and my daughters, the exciting future.

—Gareth

Table of Contents

Introduction 1

Programming Is Fun! 1

A Bit of History 3

So What Can You Do with the Raspberry Pi? 8

Part I The Board 11

CHAPTER 1

Meet the Raspberry Pi 13

A Trip Around the Board 13

Model A/B 16

Model A+/B+ 16

Raspberry Pi 2 17

Raspberry Pi 3 18

Raspberry Pi Zero 19

A Bit of Background 20

ARM Versus x86 20

Windows Versus Linux 21

CHAPTER 2

Getting Started with the Raspberry Pi 23

Connecting a Display 23

Composite Video 24

HDMI Video 25

DSI Video 26

Connecting Audio 26

Connecting a Keyboard and Mouse 27

Installing NOOBS on an SD Card 29

Connecting External Storage 30

Connecting the Network 31

Wired Networking 32

Wireless Networking 33

Connecting Power 34

Installing the Operating System 35

Installing Using NOOBS 35

Installing Manually 37

Connecting Bluetooth Devices 41

CHAPTER 3
Linux System Administration 43
Linux: An Overview 43
Linux Basics 46
Introducing Raspbian 46
About Raspbian's Parent, Debian 51
Alternatives to Raspbian. 51
Using External Storage Devices 52
Creating a New User Account 54
File System Layout 54
Logical Layout 55
Physical Layout 57
Installing and Uninstalling Software 57
Managing Software Graphically. 57
Managing Software at the Command Line 58
Finding the Software You Want 60
Installing Software. 61
Uninstalling Software 62
Upgrading Software. 62
Shutting the Pi Down Safely 63

CHAPTER 4
Troubleshooting 65
Keyboard and Mouse Diagnostics 65
Power Diagnostics. 66
Display Diagnostics 68
Boot Diagnostics 69
Network Diagnostics 69

CHAPTER 5
Network Configuration. 73
Wired Networking 73
Connecting to a Wired Network via the GUI. 73
Connecting to a Wired Network via the Console 75
Testing Your Connectivity 76
Wireless Networking 76
Connecting to a Wireless Network via the GUI 77
Connecting to a Wireless Network via the Console 79

CHAPTER 6
The Raspberry Pi Configuration Tool 85
Running the Tool 85
The System Tab 86
Filesystem 86
Password 87
Hostname 88
Boot 88
Auto Login 88
Network at Boot 89
Overscan 89
Rastrack 89
The Interfaces Tab 90
Camera 91
SSH 91
SPI 91
I^2C 91
Serial 91
1-Wire 92
Performance 92
Overclock 92
GPU Memory 94
Localisation 94
Locale 94
Timezone 96
Keyboard 96

CHAPTER 7
Advanced Raspberry Pi Configuration 99
Editing Configuration Files via NOOBS 99
Hardware Settings: config.txt 101
Modifying the Display 102
Boot Options 105
Overclocking the Raspberry Pi 106
Disabling L2 Cache 110
Enabling Test Mode 110
Memory Partitioning 111
Software Settings: cmdline.txt 112

Part II Building a Media Centre or Productivity Machine 115

CHAPTER 8

The Pi as a Home Theatre PC. 117

Playing Music at the Console 117
Dedicated HTPC with OSMC 119
Streaming Internet Media 122
Streaming Local Network Media 123
Configuring OSMC 124

CHAPTER 9

The Pi as a Productivity Machine 127

Using Cloud-Based Apps 127
Using LibreOffice 130
Image Editing with the Gimp 131

Part III Programming the Pi 135

CHAPTER 10

An Introduction to Scratch. 137

Introducing Scratch 137
Example 1: Hello World 138
Example 2: Animation and Sound 141
Example 3: A Simple Game 144
Interfacing Scratch with Hardware. 149
Further Reading 152

CHAPTER 11

An Introduction to Python 153

Introducing Python 153
Example 1: Hello World 154
Example 2: Comments, Inputs, Variables, and Loops 159
Example 3: Gaming with pygame 164
Example 4: Python and Networking 172
Further Reading 179

CHAPTER 12

Minecraft Pi Edition 181

Introducing Minecraft Pi Edition 181
Installing Minecraft 182

Running Minecraft 182
Exploration 184
Hacking Minecraft 185

Part IV Hardware Hacking 191

CHAPTER 13
Learning to Hack Hardware 193
Electronic Equipment 193
Reading Resistor Colour Codes 195
Sourcing Components 197
Online Sources 197
Offline Sources 198
Hobby Specialists 199
Moving Up from the Breadboard 199
A Brief Guide to Soldering 202

CHAPTER 14
The GPIO Port 207
Identifying Your Board Revision 207
GPIO Pinout Diagrams 208
GPIO Features 210
UART Serial Bus 211
I^2C Bus 211
SPI Bus 211
Using the GPIO Port in Python 212
GPIO Output: Flashing an LED 212
GPIO Input: Reading a Button 216
Soldering the Raspberry Pi Zero's GPIO Header 220

CHAPTER 15
The Raspberry Pi Camera Module 223
Why Use the Camera Module? 224
Choosing a Camera Module 224
Installing the Camera Module 225
Enabling Camera Mode 228
Capturing Stills 230
Recording Video 232
Command-Line Time-Lapse Photography 233

CHAPTER 16
Add-On Hardware 237
Official Raspberry Pi Case 238
Installation 239
Raspberry Pi 7" Touchscreen Display 240
Installation 241
Sense HAT 244
Installation 245
Programming the Sense HAT 247

Part V Appendixes 251
APPENDIX A
Python Recipes 253
Raspberry Snake (Chapter 11, Example 3) 253
IRC User List (Chapter 11, Example 4) 255
GPIO Input and Output (Chapter 14) 257

APPENDIX B
Raspberry Pi Camera Module Quick Reference 259
Shared Options 259
Raspistill Options 264
Raspivid Options 266

APPENDIX C
HDMI Display Modes 269

Index 277

Introduction

"CHILDREN TODAY ARE digital natives", said a man I got talking to at a fireworks party. "I don't understand why you're making this thing. My kids know more about setting up our PC than I do."

I asked him if they could program, to which he replied: "Why would they want to? The computers do all the stuff they need for them already, don't they? Isn't that the point?"

As it happens, plenty of kids today aren't digital natives. We have yet to meet any of these imagined wild digital children, swinging from ropes of twisted-pair cable and chanting war songs in nicely parsed Python. In the Raspberry Pi Foundation's educational outreach work, we do meet a lot of kids whose entire interaction with technology is limited to closed platforms with graphical user interfaces (GUIs) that they use to play movies, do a spot of word-processed homework, and play games. They can browse the web, upload pictures and video, and even design web pages. (They're often better at setting the satellite TV box than Mum or Dad, too.) It's a useful toolset, but it's shockingly incomplete, and in a country where 20 percent of households still don't have a computer in the home, even this toolset is not available to all children.

Despite the most fervent wishes of my new acquaintance at the fireworks party, computers don't program themselves. We need an industry full of skilled engineers to keep technology moving forward, and we need young people to be taking those jobs to fill the pipeline as older engineers retire and leave the industry. But there's much more to teaching a skill like programmatic thinking than breeding a new generation of coders and hardware hackers. Being able to structure your creative thoughts and tasks in complex, non-linear ways is a learned talent, and one that has huge benefits for everyone who acquires it, from historians to designers, lawyers, and chemists.

Programming Is Fun!

It's enormous, rewarding, creative fun. You can create gorgeous intricacies, as well as (much more gorgeous, in my opinion) clever, devastatingly quick, and deceptively simple-looking routes through, under, and over obstacles. You can make stuff that'll have other people looking on jealously, and that'll make you feel wonderfully smug all afternoon. In my day job, where I design the sort of silicon chips that we use in the Raspberry Pi as a processor and work on the low-level software that runs on them, I basically get paid to sit around all day playing. What could be better than equipping people to be able to spend a lifetime doing that?

It's not even as if we're coming from a position where children don't want to get involved in the computer industry. A big kick up the backside came a few years ago, when we were moving quite slowly on the Raspberry Pi project. All the development work on Raspberry Pi was done in the spare evenings and weekends of the Foundation's trustees and volunteers—we're a charity, so the trustees aren't paid by the Foundation, and we all have full-time jobs to pay the bills. This meant that occasionally, motivation was hard to come by when all I wanted to do in the evening was slump in front of the *Arrested Development* boxed set with a glass of wine. One evening, when not slumping, I was talking to a neighbour's nephew about the subjects he was taking for his General Certificate of Secondary Education (GCSE, the British system of public examinations taken in various subjects from the age of about 16), and I asked him what he wanted to do for a living later on.

"I want to write computer games", he said.

"Awesome. What sort of computer do you have at home? I've got some programming books you might be interested in."

"A Wii and an Xbox."

On talking with him a bit more, it became clear that this perfectly smart kid had never done any real programming at all; that there wasn't any machine that he could program in the house; and that his information and communication technology (ICT) classes—where he shared a computer and was taught about web page design, using spreadsheets and word processing—hadn't really equipped him to use a computer even in the barest sense. But computer games were a passion for him (and there's nothing peculiar about wanting to work on something you're passionate about). So that was what he was hoping the GCSE subjects he'd chosen would enable him to do. He certainly had the artistic skills that the games industry looks for, and his maths and science marks weren't bad. But his schooling had skirted around any programming—there were no Computing options on his syllabus, just more of the same ICT classes, with its emphasis on end users rather than programming. And his home interactions with computing meant that he stood a vanishingly small chance of acquiring the skills he needed in order to do what he really wanted to do with his life.

This is the sort of situation I want to see the back of, where potential and enthusiasm is squandered to no purpose. Now, obviously, I'm not monomaniacal enough to imagine that simply making the Raspberry Pi is enough to effect all the changes that are needed. But I do believe that it can act as a catalyst. We've already seen big changes in UK schools, where a revamped Computing curriculum arrived on the syllabus in 2014, and we've seen a massive change in awareness of a gap in our educational and cultural provision for kids just in the four years since the first Raspberry Pi was launched.

Too many of the computing devices a child will interact with daily are so locked down that they can't be used creatively as a tool—even though computing *is* a creative subject. Try using your iPhone to act as the brains of a robot, or getting your PS4 to play a game you've written. Sure, you can program the home PC, but there are significant barriers in doing that which a lot of children don't overcome: the need to download special software, and having the sort of parents who aren't worried about you breaking something that they don't know how to fix. And plenty of kids aren't even aware that doing such a thing as programming the home PC is possible. They think of the PC as a machine with nice clicky icons that give you an easy way to do the things you need to do so you don't need to think much. It comes in a sealed box, which Mum and Dad use to do the banking and which will cost lots of money to replace if something goes wrong!

The Raspberry Pi is cheap enough to buy with a few weeks' pocket money, and you probably already have all the equipment you need to make it work: a TV, an SD card that can come from an old camera, a mobile phone charger, a keyboard, and a mouse. It's not shared with the family; it belongs to the kid; and it's small enough to put in a pocket and take to a friend's house. If something goes wrong, it's no big deal—you just swap out a new SD card and your Raspberry Pi is factory-new again. And all the tools, environments, and learning materials that you need to get started on the long, smooth curve to learning how to program your Raspberry Pi are right there, waiting for you as soon as you turn it on.

A Bit of History

I started work on a tiny, affordable, bare-bones computer in 2006, when I was a Director of Studies in Computer Science at Cambridge University. I'd received a degree at the University Computer Lab as well as studying for a PhD while teaching there, and over that period, I'd noticed a distinct decline in the skillset of the young people who were applying to read Computer Science at the Lab. From a position in the mid-1990s, when 17-year-olds wanting to read Computer Science had come to the University with a grounding in several computer languages, knew a bit about hardware hacking, and often even worked in assembly language, we gradually found ourselves in a position where, by 2005, those kids were arriving having done some HTML—with a bit of PHP and Cascading Style Sheets if you were lucky. They were still fearsomely clever kids with lots of potential, but their experience with computers was entirely different from what we'd been seeing before.

The Computer Science course at Cambridge includes about 60 weeks of lecture and seminar time over three years. If you're using the whole first year to bring students up to speed, it's harder to get them to a position where they can start a PhD or go into industry over the next two years. The best undergraduates—the ones who performed the best at the end of their three-year course—were the ones who weren't just programming when they'd been told to for their weekly assignment or for a class project. They were the ones who were programming

in their spare time. So the initial idea behind the Raspberry Pi was a very parochial one with a very tight (and pretty unambitious) focus: I wanted to make a tool to get the small number of applicants to this small university course a kick start. My colleagues and I imagined we'd hand out these devices to schoolkids at open days, and if they came to Cambridge for an interview a few months later, we'd ask what they'd done with the free computer we'd given them. Those who had done something interesting would be the ones that we'd be interested in having in the program. We thought maybe we'd make a few hundred of these devices, or best case, a lifetime production run of a few thousand.

Of course, once work was seriously underway on the project, it became obvious that there was a lot more we could address with a cheap little computer like this. What we started with is a long way indeed from the Raspberry Pi you see today. I began by soldering up the longest piece of breadboard you can buy at Maplin with an Atmel chip at our kitchen table, and the first crude prototypes used cheap microcontroller chips to drive a standard-definition TV set directly. With only 512 K of RAM, and a few MIPS of processing power, these prototypes were very similar in performance to the original 8-bit microcomputers. It was hard to imagine these machines capturing the imaginations of kids used to modern games consoles and iPads.

There had been discussions at the University Computer Lab about the general state of computer education, and when I left the Lab for a non-academic job in the industry, I noticed that I was seeing the same issues in young job applicants as I'd been seeing at the University. So I got together with my colleagues Dr Rob Mullins and Professor Alan Mycroft (two colleagues from the Computer Lab), Jack Lang (who lectures in entrepreneurship at the University), Pete Lomas (a hardware guru), and David Braben (a Cambridge games industry leading light with an invaluable address book), and over beers (and, in Jack's case, cheese and wine), we set up the Raspberry Pi Foundation—a little charity with big ideas.

In my new role as a chip architect at Broadcom, a big semiconductor company, I had access to inexpensive but high-performing hardware produced by the company with the intention of being used in what were then very high-end mobile phones—the sort with the HD video and the 14-megapixel cameras. I was amazed by the difference between the chips you could buy for $10 as a small developer, and what you could buy as a cell-phone manufacturer for roughly the same amount of money: general purpose processing, 3D graphics, video, and memory bundled into a single BGA package the size of a fingernail. These microchips consume very little power, and have big capabilities. They are especially good at multimedia, and were already being used by set-top box companies to play high-definition video. A chip like this seemed the obvious next step for the shape the Raspberry Pi was taking, so I worked on taping out a low-cost variant that had an ARM microprocessor on board and could handle the processing grunt we needed.

Why "Raspberry Pi"?

We get asked a lot where the name "Raspberry Pi" came from. Bits of the name came from different trustees. It's one of the very few successful bits of design by committee I've seen, and to be honest, I hated it at first. (I have since come to love the name, because it works really well—but it took a bit of getting used to since I'd been calling the project the "ABC Micro" in my head for years.) It's "Raspberry" because there's a long tradition of fruit names in computer companies (besides the obvious, there are the old Tangerine and Apricot computers—and we like to think of the Acorn as a fruit as well). "Pi" is a mangling of "Python", which we thought early on in development would be the only programming language available on a much less powerful platform than the Raspberry Pi we ended up with. As it happens, we still recommend Python as our favourite language for learning and development, but there is a world of other language options you can explore on the Raspberry Pi too.

We felt it was important to have a way to get kids enthusiastic about using a Raspberry Pi even if they didn't feel very enthusiastic about programming. In the 1980s, if you wanted to play a computer game, you had to boot up a box that went "bing" and fed you a command prompt. It required typing a little bit of code just to get started, and most users didn't ever go beyond that—but some did, and got beguiled into learning how to program by that little bit of interaction. We realised that the Raspberry Pi could work as a very capable, very tiny, very cheap modern media centre, so we emphasised that capability to suck in the unwary—with the hope that they'd pick up some programming while they're at it.

After about five years' hard grind, we had created a very cute prototype board, about the size of a thumb drive. We included a permanent camera module on top of the board to demonstrate the sort of peripherals that can easily be added (there was no camera when we launched because it brought the price up too much, but we've now made a separate, cheap camera module available for photography projects), and brought it along to a number of meetings with the BBC's R&D department. Those of us who grew up in the UK in the 1980s had learned a lot about 8-bit computing from the BBC Microcomputer and the ecosystem that had grown up around it—with BBC-produced books, magazines and TV programmes—so I'd hoped that they might be interested in developing the Raspberry Pi further. But as it turned out, something has changed since we were kids: various competition laws in the UK and the EU meant that "the Beeb" couldn't become involved in the way we'd hoped. In a last-ditch attempt to get *something* organised with them, we ditched the R&D department idea and David (he of the giant address book) organised a meeting with Rory Cellan-Jones, a senior tech journalist, in May 2011. Rory didn't hold out much hope for partnership with the BBC, but he did ask if he could take a video of the little prototype board with his phone, to put on his blog.

The next morning, Rory's video had gone viral, and I realised that we had accidentally promised the world that we'd make everybody a $25 computer.

While Rory went off to write another blog post on exactly what it is that makes a video go viral, we went off to put our thinking caps on. That original, thumb-drive-sized prototype didn't fit the bill: with the camera included as standard, it was way too expensive to meet the cost model we'd suggested (the $25 figure came from my statement to the BBC that the Raspberry Pi should cost around the same as a text book, and is a splendid demonstration of the fact that I had no idea how much text books cost these days), and the tiny prototype model didn't have enough room around its periphery for all the ports we needed to make it as useable as we wanted it to be. So we spent a year working on engineering the board to lower cost as much as possible while retaining all the features we wanted (engineering cost down is a harder job than you might think), and to get the Raspberry Pi as useable as possible for people who might not be able to afford much in the way of peripherals.

We knew we wanted the Raspberry Pi to be used with TVs at home, just like the ZX Spectrum in the 1980s, saving the user the cost of a monitor. But not everybody has access to an HDMI television, so we added a composite port to make the Raspberry Pi work with an old cathode-ray television instead. Since SD cards are cheap and easy to find, we chose them as our storage medium: the original Model A and Model B used full-size cards, while more recent iterations have moved to the now more common microSD standard. And we went through several iterations of power supply, ending up with a micro USB cable. Recently, micro USB became the standard charger cable for mobile telephones across the EU (and it's becoming the standard everywhere), which means the cables are becoming more and more ubiquitous, and in many cases, people already have them at home.

By the end of 2011, with a projected February release date, it was becoming obvious to us that things were moving faster, and demand was higher, than we were ever going to be able to cope with. The initial launch was always aimed at developers, with the educational launch planned for later in 2012. We had a small number of very dedicated volunteers, but we needed the wider Linux community to help us prepare a software stack and iron out any early-life niggles with the board before releasing into the educational market. We had enough capital in the Foundation to buy the parts for and build 10,000 Raspberry Pis over a period of a month or so, and we thought that the people in the community who would be interested in an early board would come to around that number. Fortunately and unfortunately, we'd been really successful in building a big online community around the device, and interest wasn't limited to the UK, or to the educational market. Ten thousand was looking less and less realistic.

There were 100,000 people on our mailing list wanting a Raspberry Pi—and they all put an order in on day one! Not surprisingly, this brought up a few issues.

Our Community

The Raspberry Pi community is one of the things we're proudest of. We started with a very bare-bones blog at `www.raspberrypi.org` just after Rory's May 2011 video, and put up a forum on the same website shortly after that. That forum now has more than 170,000 members—between them they've contributed nearly a million posts of wit and wisdom about the Raspberry Pi. If there's any question, no matter how abstruse, that you want to ask about the Raspberry Pi or about programming in general, someone there will have the answer (if it's not in this book, you'll find it in the forums).

Part of my job at Raspberry Pi involves giving talks to hacker groups, computing conferences, teachers, programming collectives, and the like, and there's always someone in the audience who has talked to me or to my wife Liz (who runs the community) on the Raspberry Pi website—and some of these people have become good friends of ours. The Raspberry Pi website gets more than one request every single second of the day.

There are now hundreds of fan sites out there. For several years, there was a fan magazine called *The MagPi*, which was produced monthly by community members, with type-in listings, lots of articles, project guides, tutorials, and more. This became so successful that we brought it in-house at the Foundation, which makes it available in print or as a free download from `www.raspberrypi.org/magpi`. Type-in games in magazines and books provided an easy route into programming for me—my earliest programming experience with the BBC Micro was of modifying a type-in helicopter game to add enemies and pick-ups.

We blog something interesting about the device at `www.raspberrypi.org` at least once every day. Come and join in the conversation!

First off, there are the inevitable paper cuts you're going to get boxing up 100,000 little computers and mailing them out—and the fact was that we had absolutely no money to hire people to do this for us. We didn't have a warehouse—we had Jack's garage. There was no way we could raise the money to build 100,000 units at once—we'd envisaged making them in batches of 2,000 every couple of weeks, which, with this level of interest, was going to take so long that the thing would be obsolete before we managed to fulfil all the orders. Clearly, manufacturing and distribution were something we were going to have to give up on and hand over to somebody else who already had the infrastructure and capital to do that, so we got in touch with element14 and RS Components, both UK microelectronics suppliers with worldwide businesses, and contracted with them to do the actual manufacture and distribution side of things worldwide so we could concentrate on development and the Raspberry Pi Foundation's charitable goals.

Demand on the first day was still so large that RS and element14's websites both crashed for most of the day—at one point in the day, element14 were getting seven orders a second, and for a couple of hours on February 29, Google showed more searches were made worldwide for "Raspberry Pi" than were made for "Lady Gaga". We made and sold more than a million Raspberry Pis in the first year of business, making Raspberry Pi the fastest-growing computer company in the world, ever. Things aren't slowing down: we make more than 300,000 Pis every month and have sold more than ten million in a little over four, with no hint of a slowdown. If we'd stuck with our original plans, we'd have made 100 or so of these devices for University open days, and that would have been it.

NOTE The first production Pis were made in Chinese factories, but in 2012 we managed to repatriate all of the production to the UK. Your Raspberry Pi is now made in South Wales, in an area of the country with a proud manufacturing heritage, but few remaining factories. Amazingly, it costs us less to manufacture in Wales as it did in China, and we're able to do that manufacture without a language or cultural barrier, and with the ability to jump in the car and be on the factory floor in a few hours if necessary.

There is nothing that affects the blood pressure quite like accidentally ending up running a large computer company!

So What Can You Do with the Raspberry Pi?

This book explores a number of things you can do with your Raspberry Pi, from controlling hardware with Python, to using it as a media centre, setting up camera projects, or building games in Scratch. The beauty of the Raspberry Pi is that it's just a very tiny general-purpose computer (which may be a little slower than you're used to for some desktop applications, but much better at some other stuff than a regular PC), so you can do anything you could do on a regular computer with it. In addition, the Raspberry Pi has powerful multimedia and 3D graphics capabilities, so it has the potential to be used as a games platform, and we very much hope to see more people starting to write games for it.

We think physical computing—building systems using sensors, motors, lights, and microcontrollers—is something that gets overlooked in favour of pure software projects in a lot of instances, and it's a shame, because physical computing is *massive fun*. To the extent that there was any children's computing movement when we began this project, it was a physical computing movement. The LOGO turtles that represented physical computing when we were kids are now fighting robots, quadcopters, or parent-sensing bedroom doors, and we love it. However, the lack of General Purpose Input/Output (GPIO) on home PCs is a real handicap for many people getting started with robotics projects. The Raspberry Pi exposes GPIO so you can get to work straight away.

I keep being surprised by ideas the community comes up with which wouldn't have crossed my mind in a thousand years: the Australian school meteor-tracking project; the Boreatton Scouts in the UK and their robot, which is controlled via an electroencephalography headset (the world's first robot controlled by Scouting brain waves); the family who are building a robot vacuum cleaner; Manuel, the talking Christmas moose. And I'm a real space cadet, so reading about the people sending Raspberry Pis into near-earth orbit on rockets and balloons gives me goosebumps.

In the first edition of this book, I said that success for us would be another 1,000 people every year taking up Computer Science at the university level in the UK. That would not only be beneficial for the country, the software and hardware industries, and the economy; but it would be even more beneficial for every one of those 1,000 people, who, I hope, discover that there's a whole world of possibilities and a great deal of fun to be had out there. In the second edition and third editions, I was a little more ambitious, saying that we'd like to see that replicated throughout the developed world. As Raspberry Pi has grown, however, I've become even more ambitious: I want every child, everywhere, to have access to an open, programmable, general-purpose computer, and to have the opportunity to learn to program in the same way that I did on my BBC Microcomputer back in the 1980s. It's a lofty goal, but we've already seen Raspberry Pi labs spring up in the most unlikely places, like a village lab in a part of Cameroon with no electricity network where the Pis run off solar power, generators, and batteries, or a school high in the mountains in Bhutan.

Building a robot when you're a kid can take you to places you never imagined—I know because it happened to me!

—Eben Upton

Part I

The Board

Chapter 1 Meet the Raspberry Pi

Chapter 2 Getting Started with Raspberry Pi

Chapter 3 Linux System Administration

Chapter 4 Troubleshooting

Chapter 5 Network Configuration

Chapter 6 The Raspberry Pi Configuration Tool

Chapter 7 Advanced Raspberry Pi Configuration