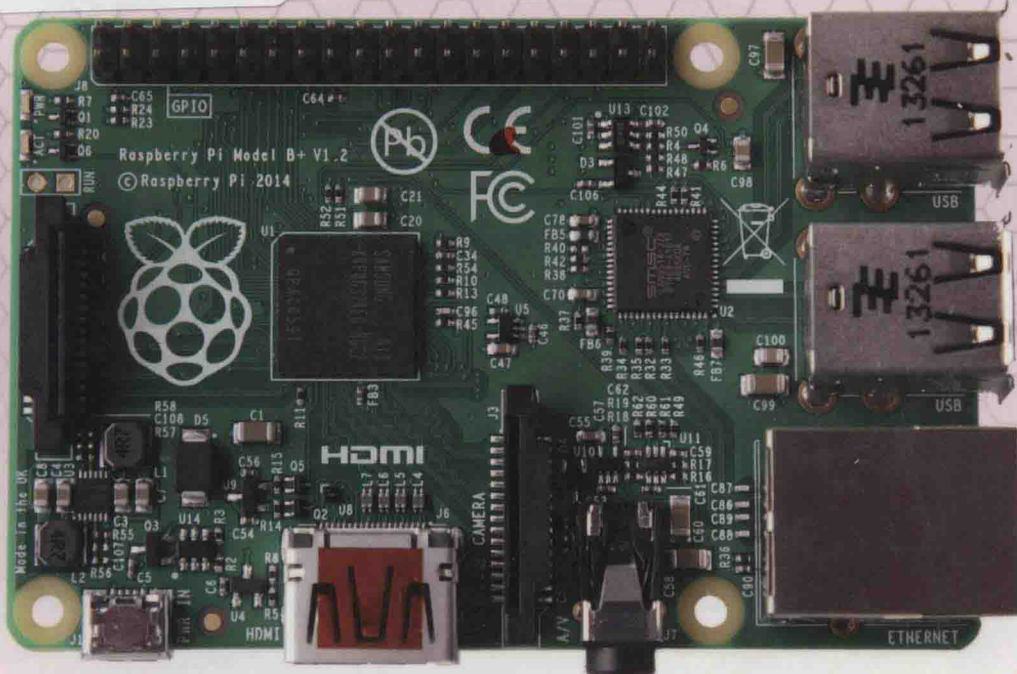
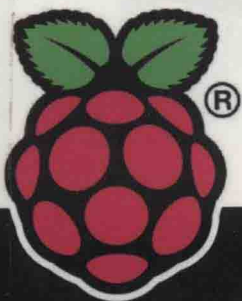


Third Edition
Updated for Model B+ Board



Raspberry Pi[®]

User Guide



Eben Upton

Co-creator of the Raspberry Pi

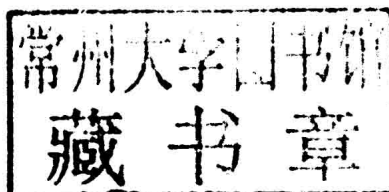
Gareth Halfacree

WILEY

Raspberry Pi[®] User Guide

Third Edition

Eben Upton and Gareth Halfacree



WILEY

This edition first published 2014

© 2014 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.

Reprinted April 2015

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.

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 978-1-118-92166-1 (Pbk); ISBN 978-1-118-92168-5 (ePDF); ISBN 978-1-118-92167-8 (ePub)

Set in 10 pt. Chaparral Pro by TCS/SPS

Printed simultaneously in Great Britain and the United States

Printed by TJ International Ltd, Padstow, Cornwall

Publisher's Acknowledgements

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

Editorial and Production

VP Consumer and Technology Publishing Director: Michelle Leete

Associate Director–Book Content Management: Martin Tribe

Associate Publisher: Chris Webb

Executive Commissioning Editor: Craig Smith

Project Editor: John Sleeva

Copy Editor: Melba Hopper

Technical Editor: Andrew Scheller

Senior Project Editor: Sara Shlaer

Editorial Manager: Rev Mengle

Editorial Assistant: Claire Johnson

Marketing

Marketing Manager: Lorna Mein

Associate Marketing Manager: Carrie Sherrill

Assistant Marketing Manager: Dave Allen

About the Authors

EBEN UPTON is a founder of the Raspberry Pi Foundation and serves as the CEO of Raspberry Pi (Trading), its commercial arm. He is a Technical Director with Broadcom, a Fortune 500 semiconductor company. In an earlier life, he founded two successful mobile games and middleware companies, Ideaworks 3d (now Marmalade) and Podfun, held the post of Director of Studies for Computer Science at St John's College, Cambridge, and wrote the Oxford Rhyming Dictionary with his father, Professor Clive Upton. He holds a BA, a PhD and an MBA from the University of Cambridge.

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 such as 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>.

For Liz, who made it all possible.

—Eben

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

—Gareth

Contents

Introduction	1
-------------------------------	----------

Part I

CHAPTER 1

Meet the Raspberry Pi	13
--	-----------

A Trip Around the Board	13
Model A	15
Model B	16
Model B+	16
A History of Model B PCB Revisions	18
Revision 1	18
Revision 2	18
Model B+	18
A Bit of Background	19
ARM versus x86	19
Windows versus Linux	20

CHAPTER 2

Getting Started with the Raspberry Pi	23
--	-----------

Connecting a Display	24
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	35
Installing the Operating System	36
Installing Using NOOBS	36
Installing Manually	38
Flashing from Linux	39

Flashing from OS X	40
Flashing from Windows	40

CHAPTER 3

Linux System Administration	43
Linux: An Overview	43
Linux Basics	45
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	55
Logical Layout	55
Physical Layout	57
Installing and Uninstalling Software	57
Obtaining Software from the Pi Store	57
Obtaining Software from Elsewhere	59
Finding the Software You Want	61
Installing Software	62
Uninstalling Software	63
Upgrading Software	63
Shutting the Pi Down Safely	64

CHAPTER 4

Troubleshooting	65
Keyboard and Mouse Diagnostics	65
Power Diagnostics	66
Display Diagnostics	68
Boot Diagnostics	68
Network Diagnostics	69
The Emergency Kernel	72

CHAPTER 5

Network Configuration	75
Wired Networking	75
Wireless Networking	78
Installing Firmware	79

Connecting to a Wireless Network via wpa_gui	82
Connecting to a Wireless Network via the Terminal	85
No Encryption	90
WEP Encryption	90
WPA/WPA2 Encryption	90
Connecting to the Wireless Network	91

CHAPTER 6

The Raspberry Pi Software Configuration Tool 93

Running the Tool	94
The Setup Options Screen	94
1 Expand Filesystem	95
2 Change User Password	95
3 Enable Boot to Desktop/Scratch	96
4 Internationalisation Options	96
I1 Change Locale	97
I2 Change Timezone	97
I3 Change Keyboard Layout	98
5 Enable Camera	98
6 Add to Rastrack	98
7 Overclock	99
8 Advanced Options	100
A1 Overscan	101
A2 Hostname	101
A3 Memory Split	102
A4 SSH	103
A5 SPI	103
A6 Audio	103
A7 Update	103
9 About raspi-config	104

CHAPTER 7

Advanced Raspberry Pi Configuration 105

Editing Configuration Files via NOOBS	105
Hardware Settings—config.txt	107
Modifying the Display	108
Boot Options	111

Overclocking the Raspberry Pi	112
Overclocking Settings	113
Overvoltage Settings	114
Disabling L2 Cache	115
Enabling Test Mode	116
Memory Partitioning	117
Software Settings—cmdline.txt	117

Part II

CHAPTER 8

The Pi as a Home Theatre PC. 123

Playing Music at the Console	123
Dedicated HTPC with Raspbmc	126
Streaming Internet Media	127
Streaming Local Network Media	129
Configuring Raspbmc	131

CHAPTER 9

The Pi as a Productivity Machine 133

Using Cloud-Based Apps	134
Using LibreOffice	136
Image Editing with the Gimp	138

CHAPTER 10

The Pi as a Web Server 141

Installing a LAMP Stack	142
Installing WordPress	145

Part III

CHAPTER 11

An Introduction to Scratch. 153

Introducing Scratch	153
Example 1: Hello World	154
Example 2: Animation and Sound	158
Example 3: A Simple Game	161
Robotics and Sensors	167
Sensing with the PicoBoard	167
Robotics with LEGO	167
Further Reading	168

CHAPTER 12**An Introduction to Python 169**

Introducing Python	169
Example 1: Hello World	170
Example 2: Comments, Inputs, Variables and Loops	175
Example 3: Gaming with pygame	179
Example 4: Python and Networking	188
Further Reading	194

CHAPTER 13**Minecraft Pi Edition 195**

Introducing Minecraft Pi Edition	195
Installing Minecraft	196
Running Minecraft	197
Exploration	199
Hacking Minecraft	200

Part IV**CHAPTER 14****Learning to Hack Hardware 207**

Electronic Equipment	208
Reading Resistor Colour Codes	210
Sourcing Components	210
Online Sources	211
Offline Sources	212
Hobby Specialists	213
Moving Up from the Breadboard	214
A Brief Guide to Soldering	217

CHAPTER 15**The GPIO Port 223**

Identifying Your Board Revision	223
GPIO Pinout Diagrams	224
GPIO Features	226
UART Serial Bus	227
I ² C Bus	227
SPI Bus	228

Using the GPIO Port in Python	228
GPIO Output: Flashing an LED.....	228
GPIO Input: Reading a Button.....	233
 CHAPTER 16	
The Raspberry Pi Camera Module.....	237
Why Use the Camera Module?	238
Installing the Camera Module.....	239
Enabling Camera Mode	242
Capturing Stills	244
Recording Video	246
Command-Line Time-Lapse Photography	247
 CHAPTER 17	
Add-On Boards.....	255
Ciseco Slice of Pi	255
Adafruit Prototyping Pi Plate	259
Fen Logic Gertboard.....	262
 Part V	
APPENDIX A	
Python Recipes.....	269
Raspberry Snake (Chapter 12, Example 3)	269
IRC User List (Chapter 12, Example 4)	272
GPIO Input and Output (Chapter 15)	273
 APPENDIX B	
Raspberry Pi Camera Module Quick Reference.....	275
Shared Options	275
Raspistill Options	278
Raspivid Options.....	279
Raspiyuv Options	280
 APPENDIX C	
HDMI Display Modes.....	281
 Index.....	 287

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're already seeing big changes in the UK schools' curriculum, where Computing is arriving on the syllabus this year and ICT is being entirely reshaped, and we've seen a massive change in awareness of a gap in our educational and cultural provision for kids just in the short time since the 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 PS3 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 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.

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.