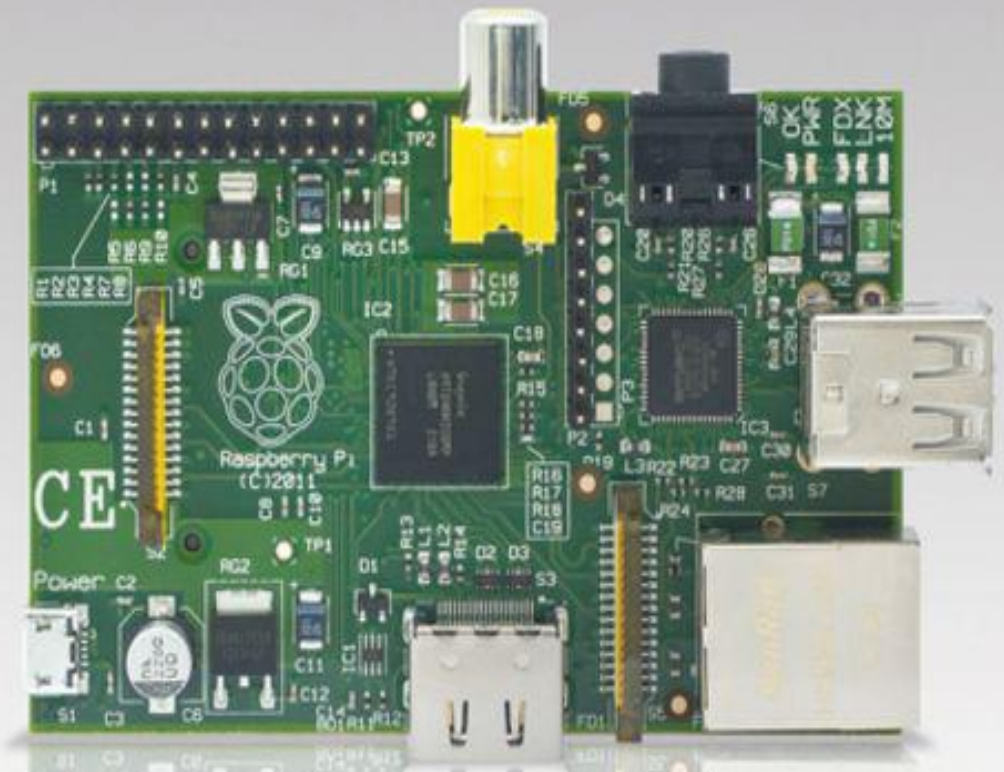



# Raspberry Pi®

## User Guide

**Eben Upton**
Co-creator of the Raspberry Pi

**Gareth Halfacree**

# Raspberry Pi® User Guide

**Table of Contents**

# Raspberry Pi® User Guide

Eben Upton and Gareth Halfacree

**Raspberry Pi® User Guide**

This edition first published 2012



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.



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

ISBN 978-1-118-46446-5 (pbk); ISBN 978-1-118-46448-9 (ebk); ISBN 978-1-118-46447-2 (ebk); ISBN 978-1-118-46449-6 (ebk)

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 Consumer and Technology Publishing Director**

Michelle Leete

**Associate Director–Book Content Management**

Martin Tribe

**Associate Publisher**

Chris Webb

**Executive Commissioning Editor**

Craig Smith

**Assistant Editor**

Ellie Scott

**Project Editor**

Kathryn Duggan

**Copy Editor**

Kathryn Duggan

**Technical Editor**

Omer Kilic

**Editorial Manager**

Jodi Jensen

**Senior Project Editor**

Sara Shlaer

**Editorial Assistant**

Leslie Saxman

**Marketing**

**Associate Marketing Director**

Louise Breinholt

**Marketing Manager**

Lorna Mein

**Senior Marketing Executive**

Kate Parrett

**Composition Services**

**Compositor**

Erin Zeltner

**Proofreader**

Wordsmith Editorial

**Indexer**

BIM Indexing & Proofreading Services

# 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.

For Liz, who made it all possible.
—Eben

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

# Introduction

**"Children today are** digital natives", said a man I got talking to at a fireworks party last year. "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% 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 and ICT is being 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 about six years ago, 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.**

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 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.

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, 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 decided against microSD as the storage medium, because the little fingernail-sized cards are so flimsy in the hands of children and so easy to lose. 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 have a small number of very dedicated volunteers, but we need 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.

**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 20,000 members—between them they've contributed more than 100,000 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 around one request every single second of the day.**

**There are now hundreds of fan sites out there. There's also a fan magazine called The MagPi (a free download from www.themagpi.com), which is produced monthly by community members, with type-in listings, lots of articles, project guides, tutorials and more. 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!**

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.

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". I'm writing this in early June 2012, and orders in the three months since we opened for business have topped half a million units, even though we're still at a point when neither company will sell you more than one Raspberry Pi (they're trying to get rid of their order backlogs before they turn on the ability to multiorder). At this point, if we'd gone with our original plans, we'd have made 100 or so of these devices for University open days, and that would have been it.

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, 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 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's any children's computing movement at the moment, it's 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. 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.

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. 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: Connecting the Board

# Chapter 1: Meet the Raspberry Pi

**Your Raspberry Pi** board is a miniature marvel, packing considerable computing power into a footprint no larger than a credit card. It's capable of some amazing things, but there are a few things you're going to need to know before you plunge head-first into the bramble patch.

**TIP**

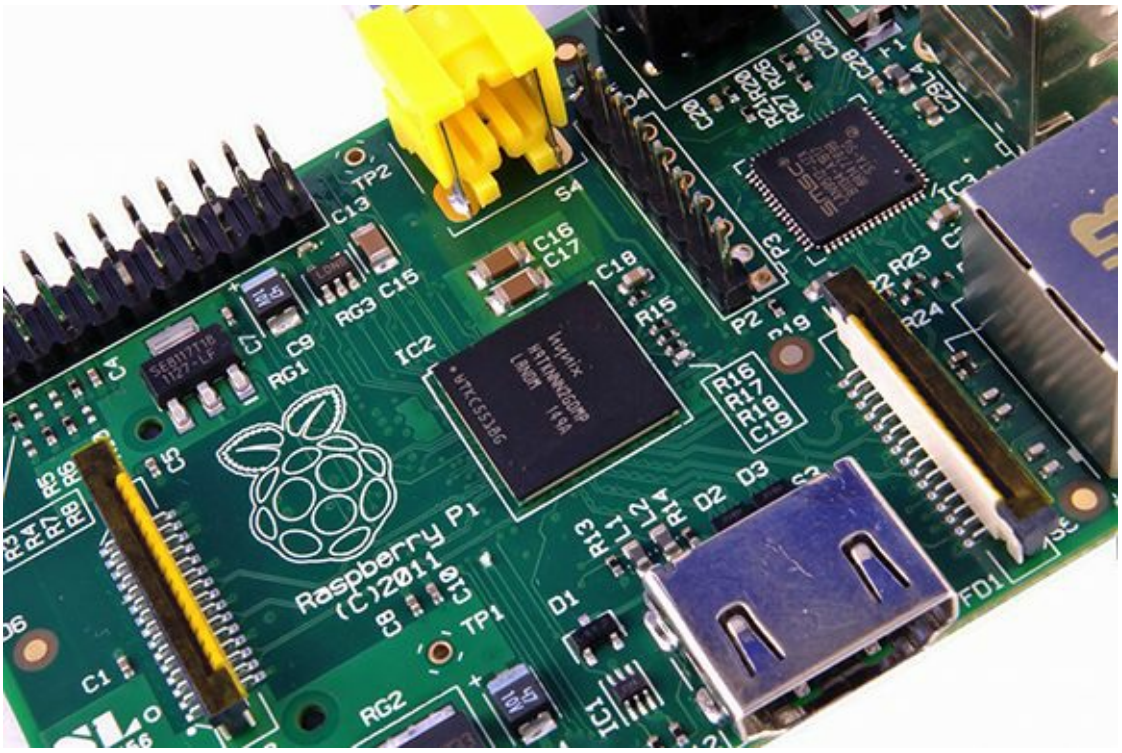
**If you're eager to get started, skip ahead a couple of pages to find out how to connect your Raspberry Pi to a display, keyboard and mouse.**

# ARM vs. x86

The processor at the heart of the Raspberry Pi system is a Broadcom BCM2835 system-on-chip (SoC) multimedia processor. This means that the vast majority of the system's components, including its central and graphics processing units along with the audio and communications hardware, are built onto that single component hidden beneath the 256 MB memory chip at the centre of the board (see Figure 1-1).

It's not just this SoC design that makes the BCM2835 different to the processor found in your desktop or laptop, however. It also uses a different instruction set architecture (ISA), known as ARM.

Figure 1-1: The BCM2835 SoC, located beneath a Hynix memory chip




Developed by Acorn Computers back in the late 1980s, the ARM architecture is a relatively uncommon sight in the desktop world. Where it excels, however, is in mobile devices: the phone in your pocket almost certainly has at least one ARM-based processing core hidden away inside. Its combination of a simple reduced instruction set (RISC) architecture and low power draw make it the perfect choice over desktop chips with high power demands and complex instruction set (CISC) architectures.

The ARM-based BCM2835 is the secret of how the Raspberry Pi is able to operate on just the 5V 1A power supply provided by the onboard micro-USB port. It's also the reason why you won't find any heat-sinks on the device: the chip's low power draw directly translates into very little waste heat, even during complicated processing tasks.

It does, however, mean that the Raspberry Pi isn't compatible with traditional PC software. The majority of software for desktops and laptops is built with the x86 instruction set architecture in mind, as found in processors from the likes of AMD, Intel and VIA. As a result, it won't run on the ARM-based Raspberry Pi.

The BCM2835 uses a generation of ARM's processor design known as ARM11, which in turn is designed around a version of the instruction set architecture known as ARMv6. This is worth remembering: ARMv6 is a lightweight and powerful architecture, but has a rival in the more advanced ARMv7 architecture used by the ARM Cortex family of processors. Software developed

for ARMv7, like software developed for x86, is sadly not compatible with the Raspberry Pi's BCM2835—although developers can usually convert the software to make it suitable.

That's not to say you're going to be restricted in your choices. As you'll discover later in the book, there is plenty of software available for the ARMv6 instruction set, and as the Raspberry Pi's popularity continues to grow, that will only increase. In this book, you'll also learn how to create your own software for the Pi even if you have no experience with programming.

# Windows vs. Linux

Another important difference between the Raspberry Pi and your desktop or laptop, other than the size and price, is the operating system—the software that allows you to control the computer.

The majority of desktop and laptop computers available today run one of two operating systems: Microsoft Windows or Apple OS X. Both platforms are closed source, created in a secretive environment using proprietary techniques.

These operating systems are known as closed source for the nature of their source code, the computer-language recipe that tells the system what to do. In closed-source software, this recipe is kept a closely-guarded secret. Users are able to obtain the finished software, but never to see how it's made.

The Raspberry Pi, by contrast, is designed to run an operating system called GNU/Linux—hereafter referred to simply as Linux. Unlike Windows or OS X, Linux is open source: it's possible to download the source code for the entire operating system and make whatever changes you desire. Nothing is hidden, and all changes are made in full view of the public. This open source development ethos has allowed Linux to be quickly altered to run on the Raspberry Pi, a process known as porting. At the time of this writing, several versions of Linux—known as distributions—have been ported to the Raspberry Pi's BCM2835 chip, including Debian, Fedora Remix and Arch Linux.

The different distributions cater to different needs, but they all have something in common: they're all open source. They're also all, by and large, compatible with each other: software written on a Debian system will operate perfectly well on Arch Linux and vice versa.

Linux isn't exclusive to the Raspberry Pi. Hundreds of different distributions are available for desktops, laptops and even mobile devices; and Google's popular Android platform is developed on top of a Linux core. If you find that you enjoy the experience of using Linux on the Raspberry Pi, you could consider adding it to other computing devices you use as well. It will happily coexist with your current operating system, allowing you to enjoy the benefits of both while giving you a familiar environment when your Pi is unavailable.

As with the difference between ARM and x86, there's a key point to make about the practical difference between Windows, OS X and Linux: software written for Windows or OS X won't run on Linux. Thankfully, there are plenty of compatible alternatives for the overwhelming majority of common software products—better still, the majority are free to use and as open source as the operating system itself.

# Getting Started with the Raspberry Pi

Now that you have a basic understanding of how the Pi differs from other computing devices, it's time to get started. If you've just received your Pi, take it out of its protective anti-static bag and place it on a flat, non-conductive surface before continuing with this chapter.

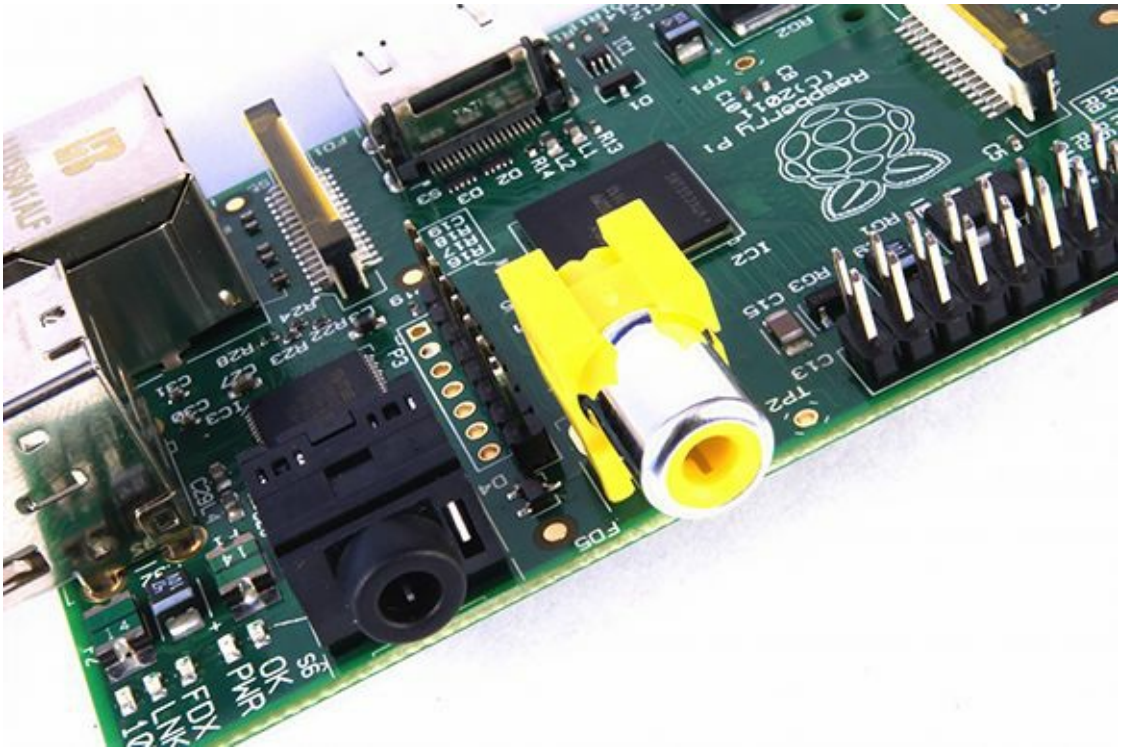
## Connecting a Display

Before you can start using your Raspberry Pi, you're going to need to connect a display. The Pi supports three different video outputs: composite video, HDMI video and DSI video. Composite video and HDMI video are readily accessible to the end user, as described in this section, while DSI video requires some specialised hardware.

### Composite Video

Composite video, available via the yellow-and-silver port at the top of the Pi known as an RCA phono connector (see Figure 1-2), is designed for connecting the Raspberry Pi to older display devices. As the name suggests, the connector creates a composite of the colours found within an image—red, green and blue—and sends it down a single wire to the display device, typically an old cathode-ray tube (CRT) TV.

Figure 1-2: The yellow RCA phono connector, for composite video output



When no other display device is available, a composite video connection will get you started with the Pi. The quality, however, isn't great. Composite video connections are significantly more prone to interference, lack clarity and run at a limited resolution, meaning that you can fit fewer icons and lines of text on the screen at once.

## HDMI Video

A better-quality picture can be obtained using the HDMI (High Definition Multimedia Interface) connector, the only port found on the bottom of the Pi (see Figure 1-3). Unlike the analogue composite connection, the HDMI port provides a high-speed digital connection for pixel-perfect pictures on both computer monitors and high-definition TV sets. Using the HDMI port, a Pi can display images at the Full HD 1920x1080 resolution of most modern HDTV sets. At this resolution, significantly more detail is available on the screen.

If you're hoping to use the Pi with an existing computer monitor, you may find that your display doesn't have an HDMI input. That's not a disaster: the digital signals present on the HDMI cable map to a common computer monitor standard called DVI (Digital Video Interconnect). By purchasing an HDMI-to-DVI cable, you'll be able to connect the Pi's HDMI port to a monitor with DVI-D connectivity.

Figure 1-3: The silver HDMI connector, for high-definition video output


If your monitor has a VGA input—a D-shaped connector with 15 pins, typically coloured silver and blue—the Raspberry Pi can't connect to it. Adapters are available that will take in a digital DVI signal and convert it to an analogue VGA signal, but these are expensive and bulky. The best option here is simply to buy a more-modern monitor with a DVI or HDMI input.

### DSI Video

The final video output on the Pi can be found above the SD card slot on the top of the printed circuit board—it's a small ribbon connector protected by a layer of plastic. This is for a video standard known as Display Serial Interface (DSI), which is used in the flat-panel displays of tablets and smartphones. Displays with a DSI connector are rarely available for retail purchase, and are typically reserved for engineers looking to create a compact, self-contained system. A DSI display can be connected by inserting a ribbon cable into the matched connector on the Pi, but for beginners, the use of a composite or HDMI display is recommended.

## Connecting Audio

If you're using the Raspberry Pi's HDMI port, audio is simple: when properly configured, the HDMI port carries both the video signal and a digital audio signal. This means that you can connect a single cable to your display device to enjoy both sound and pictures.

Assuming you're connecting the Pi to a standard HDMI display, there's very little to do at this point. For now, it's enough to simply connect the cable.

If you're using the Pi with a DVI-D monitor via an adapter or cable, audio will not be included. This highlights the main difference between HDMI and DVI: while HDMI can carry audio signals, DVI cannot.

For those with DVI-D monitors, or those using the composite video output, a black 3.5 mm audio jack located on the top edge of the Pi next to the yellow phono connector provides analogue audio (see Figure 1-2). This is the same connector used for headphones and microphones on consumer audio equipment, and it's wired in exactly the same way. If you want, you can simply connect a pair of headphones to this port for quick access to audio.

**TIP**

**While headphones can be connected directly to the Raspberry Pi, you may find the volume a little lacking. If possible, connect a pair of powered speakers instead. The amplifier inside will help boost the signal to a more audible level.**

If you're looking for something more permanent, you can either use standard PC speakers that have a 3.5 mm connector or you can buy some adapter cables. For composite video users, a 3.5 mm to RCA phono cable is useful. This provides the two white-and-red RCA phono connections that sit alongside the video connection, each carrying a channel of the stereo audio signal to the TV.

For those connecting the Pi to an amplifier or stereo system, you'll either need a 3.5 mm to RCA phono cable or a 3.5 mm to 3.5 mm cable, depending on what spare connections you have on your system. Both cable types are readily and cheaply

available at consumer electronics shops, or can be purchased even cheaper at online retailers such as Amazon.

# Connecting a Keyboard and Mouse

Now that you've got your Raspberry Pi's output devices sorted, it's time to think about input. As a bare minimum, you're going to need a keyboard, and for the majority of users, a mouse or trackball is a necessity too.

First, some bad news: if you've got a keyboard and mouse with a PS/2 connector—a round plug with a horseshoe-shaped array of pins—then you're going to have to go out and buy a replacement. The old PS/2 connection has been superseded, and the Pi expects your peripherals to be connected over the Universal Serial Bus (USB) port.

Depending on whether you purchased the Model A or Model B, you'll have either one or two USB ports available on the right side of the Pi (see Figure 1-4). If you're using Model B, you can connect the keyboard and mouse directly to these ports. If you're using Model A, you'll need to purchase a USB hub in order to connect two USB devices simultaneously.

Figure 1-4: Model B's two USB ports


A USB hub is a good investment for any Pi user: even if you've got a Model B, you'll use up both your available ports just connecting your keyboard and mouse, leaving nothing free for additional devices such as an external optical drive, storage device or joystick. Make sure you buy a powered USB hub: passive models are cheaper and smaller, but lack the ability to run current-hungry devices like CD drives and external hard drives.

**TIP**

**If you want to reduce the number of power sockets in use, connect the Raspberry Pi's USB power lead to your powered USB hub. This way, the Pi can draw its power directly from the hub, rather than needing its own dedicated power socket and mains adapter. This will only work on hubs with a power supply capable of providing 700mA to the Pi's USB port, along with whatever power is required by other peripherals.**

Connecting the keyboard and mouse is as simple as plugging them in to the USB ports, either directly in the case of a Model B or via a USB hub in the case of a Model A.

## A Note on Storage

As you've probably noticed, the Raspberry Pi doesn't have a traditional hard drive. Instead it uses a Secure Digital (SD) memory card, a solid-state storage system typically used in digital cameras. Almost any SD card will work with the Raspberry Pi, but because it holds the entire operating system, it is necessary for the card to be at least 2 GB in capacity to store all the required files.

SD cards with the operating system preloaded are available from the official Raspberry Pi Store along with numerous other sites on the Internet. If you've

purchased one of these, or received it in a bundle with your Pi, you can simply plug it in to the SD card slot on the bottom side of the left-hand edge. If not, you'll need to install an operating system—known as flashing—onto the card before it's ready to go.

Some SD cards work better than others, with some models refusing to work at all with the Raspberry Pi. For an up-to-date list of SD card models known to work with the Pi, visit the eLinux Wiki page: http://www.elinux.org/RPi_VerifiedPeripherals#SD_cards

# Flashing the SD Card

To prepare a blank SD card for use with the Raspberry Pi, you'll need to flash an operating system onto the card. While this is slightly more complicated than simply dragging and dropping files onto the card, it shouldn't take more than a few minutes to complete.

Firstly, you'll need to decide which Linux distribution you would like to use with your Raspberry Pi. Each has its advantages and disadvantages. Don't worry if you change your mind later and want to try a different version of Linux: an SD card can be flashed again with a new operating system at any point.

The most up-to-date list of Linux releases compatible with the Pi is available from the Raspberry Pi website at http://www.raspberrypi.org/downloads.

The Foundation provides BitTorrent links for each distribution. These are small files that can be used with BitTorrent software to download the files from other users. Using these links is an efficient and fast way to distribute large files, and keeps the Foundation's download servers from becoming overloaded.

To use a BitTorrent link, you'll need to have a compatible client installed. If you don't already have a BitTorrent client installed, download one and install it before trying to download the Raspberry Pi Linux distribution. One client for Windows, OS X and Linux is µTorrent, available from http://www.utorrent.com/downloads.

Which distribution you choose to download is up to you. Instructions in the rest of the book will be based on the Debian Raspberry Pi distribution, a good choice for beginners. Where possible, we'll give you instructions for other distributions as well.

Linux distributions for the Raspberry Pi are provided as a single image file, compressed to make it faster to download. Once you've downloaded the Zip archive (a compressed file, which takes less time to download than the uncompressed files would) for your chosen distribution, you'll need to decompress it somewhere on your system. In most operating systems, you can simply double-click the file to open it, and then choose Extract or Unzip to retrieve the contents.

After you've decompressed the archive, you'll end up with two separate files. The file ending in `sha1` is a hash, which can be used to verify that the download hasn't been corrupted in transit. The file ending in `img` contains an exact copy of an SD card set up by the distribution's creators in a way that the Raspberry Pi understands. This is the file that needs to be flashed to the SD card.

**WARNING**

**During the following, you'll be using a software utility called dd. Used incorrectly `dd` will happily write the image to your main hard drive, erasing your operating system and all your stored data. Make sure you read the instructions in each section thoroughly and note the device address of your SD card carefully. Read twice, write once!**

## Flashing from Linux

If your current PC is running a variant of Linux already, you can use the `dd` command to write the contents of the image file out to the SD card. This is a text-interface program operated from the command prompt, known as a terminal in Linux parlance. Follow these steps to flash the SD card:

**1.** Open a terminal from your distribution's applications menu.

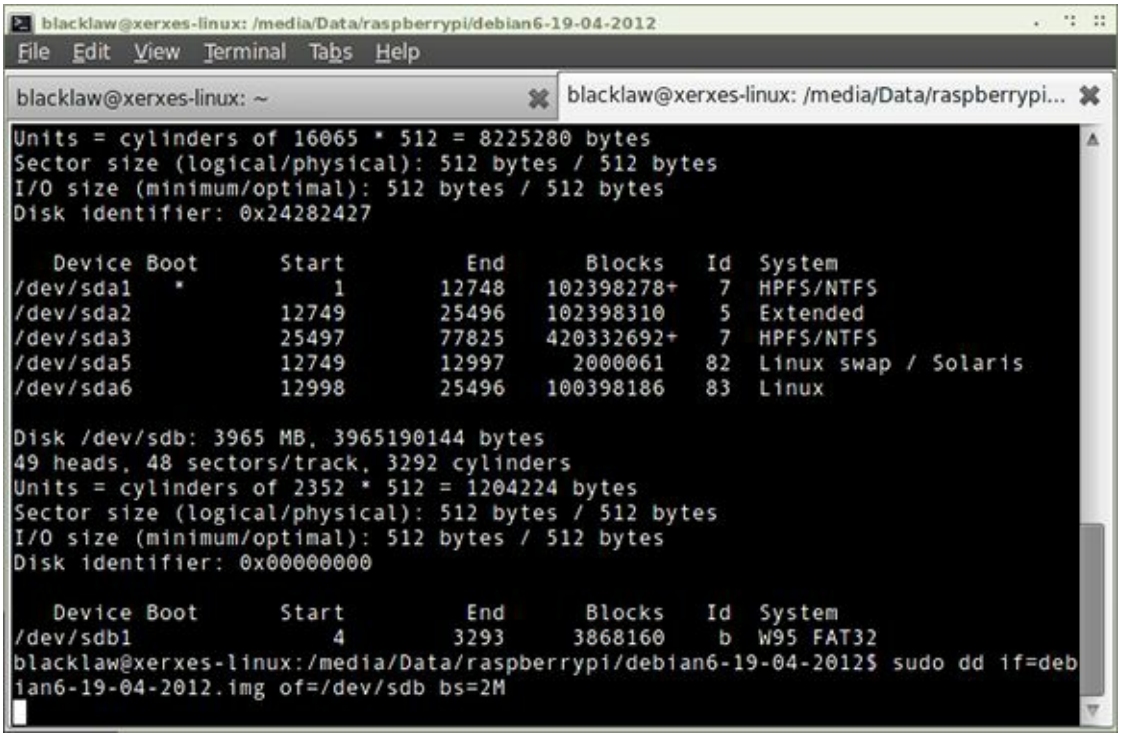
**2.** Plug your blank SD card into a card reader connected to the PC.

**3.** Type `sudo fdisk -l` to see a list of disks. Find the SD card by its size, and note the device address (`/dev/sd`X, where X is a letter identifying the storage device. Some systems with integrated SD card readers may use the alternative format `/dev/mmcblk`X—if this is the case, remember to change the target in the following instructions accordingly).

**4.** Use `cd` to change to the directory with the `.img` file you extracted from the Zip archive.

**5.** Type `sudo dd if=`imagefilename`.img of=/dev/sd`X `bs=2M` to write the file imagefilename`.img` to the SD card connected to the device address from step 3. Replace imagefilename.img with the actual name of the file extracted from the Zip archive. This step takes a while, so be patient! During flashing, nothing will be shown on the screen until the process is fully complete (see Figure 1-5).

Figure 1-5: Flashing the SD card using the dd command in Linux




## Flashing from OS X

If your current PC is a Mac running Apple OS X, you'll be pleased to hear that things are as simple as with Linux. Thanks to a similar ancestry, OS X and Linux both contain the `dd` utility, which you can use to flash the system image to your blank SD card as follows:

1. Select Utilities from the Application menu, and then click on the Terminal application.
2. Plug your blank SD card into a card reader connected to the Mac.
3. Type `diskutil list` to see a list of disks. Find the SD card by its size, and note the device address (`/dev/diskX`, where X is a letter identifying the storage device).
4. If the SD card has been automatically mounted and is displayed on the desktop, type `diskutil unmountdisk /dev/diskX` to unmount it before proceeding.
5. Use `cd` to change to the directory with the `.img` file you extracted from the Zip archive.
6. Type `dd if=`imagefilename`.img of=/dev/diskX bs=2M` to write the file imagefilename`.img` to the SD card connected to the device address from step 3. Replace imagefilename.img with the actual name of the file extracted from the Zip archive. This step takes a while, so be patient!

## Flashing from Windows

If your current PC is running Windows, things are slightly trickier than with Linux or OS X. Windows does not have a utility like `dd`, so some third-party software is required to get the image file flashed onto the SD card. Although it's possible to install a Windows-compatible version of `dd`, there is an easier way: the Image Writer for Windows. Designed specifically for creating USB or SD card images of Linux distributions, it features a simple graphical user interface that makes the creation of a Raspberry Pi SD card straightforward.

The latest version of Image Writer for Windows can be found at the official website: https://launchpad.net/win32-image-writer. Follow these steps to download, install and use the Image Writer for Windows software to prepare the SD card for the Pi:

1. Download the binary (not source) Image Writer for Windows Zip file, and extract it to a folder on your computer.
2. Plug your blank SD card into a card reader connected to the PC.
3. Double-click the `Win32DiskImager.exe` file to open the program, and click the blue folder icon to open a file browse dialogue box.
4. Browse to the imagefilename`.img` file you extracted from the distribution archive, replacing imagefilename.img with the actual name of the file extracted from the Zip archive, and then click the Open button.
5. Select the drive letter corresponding to the SD card from the Device drop-down dialogue box. If you're unsure which