Make: Sensors

Projects and Experiments to Measure the World with Arduino and Raspberry Pi

Tero Karvinen, Kimmo Karvinen & Ville Valtokari

Make: Sensors

Tero Karvinen, Kimmo Karvinen, and
Ville Valtokari

Make: Sensors

by Tero Karvinen, Kimmo Karvinen, and Ville Valtokari

Printed in Canada.

Published by Maker Media, Inc., 1005 Gravenstein Highway North, Sebastopol, CA 95472.

Maker Media books may be purchased for educational, business, or sales promotional use. Online editions are also available for most titles (*http://safaribooksonline.com*). For more information, contact O'Reilly Media's corporate/institutional sales department: 800-998-9938 or *corporate@oreilly.com*.

Editors: Brian Jepson and Emma Dvorak
Production Editor: Kristen Brown
Technical Editor: Jason Tavares
Proofreader: Julie Van Keuren
Indexer: Judith McConville
Cover Designers: Juliann Brown and Brian Jepson
Interior Designer: Nellie McKesson
Illustrators: Tero Karvinen, Kimmo Karvinen, and Ville Valtokari
Photographer: Kimmo Karvinen
Cover Photo: Kimmo Karvinen

May 2014: First Edition

Revision History for the First Edition:

2014-05-05: First release

2015-01-23: Second release

See *http://oreilly.com/catalog/errata.csp?isbn=9781449368104* for release details.

ISBN: 978-1-449-36810-4

[TI]

Table of Contents

Preface

Welcome to *Make: Sensors*. Soon you'll be making gadgets that can sense it all—from dangerous gases to acceleration. In this book, you'll use sensors to measure the physical world, represent the result as a numeric value, and take some action based on that value.

For example, a sensor could measure heat, pressure, light, or acceleration and report a value such as 22 C, 1015 millibars, light is detected, or 2.3 g acceleration (in the case of light, notice that we represented it as a Boolean or yes/no value rather than a numeric quantity; you'll see examples of this later).

A microcontroller board is the brain of the robot, system, or gadget you're building. You'll write your own software to run on the microcontroller. In this book, you'll work with two very popular boards: Arduino and Raspberry Pi. Either of these makes it easy to write software code to work with electronics.

It's About Your Ideas

If your interest in electronics started with a desire to quickly learn some basics and then design your own robots, gadgets, or projects, you're in the right place. This book will show you how to go from idea to reality quickly.

Theory, skills, and basics are useful—as long as they serve your creativity. Feel free to experiment with your ideas, and have the courage to publish your results on the Web.

Each chapter presents a mini project to show how you can combine different technologies. For example, you'll build a wooden box that you open with a fingerprint and a color-changing chameleon dome. These are fun projects, but also good starting points for things you invent later yourself.

The skills you learn with Arduino are easily applicable to real-life projects. For example, we used Arduino to build the sun sensor prototype for Finland's first satellite (Figure P-1).

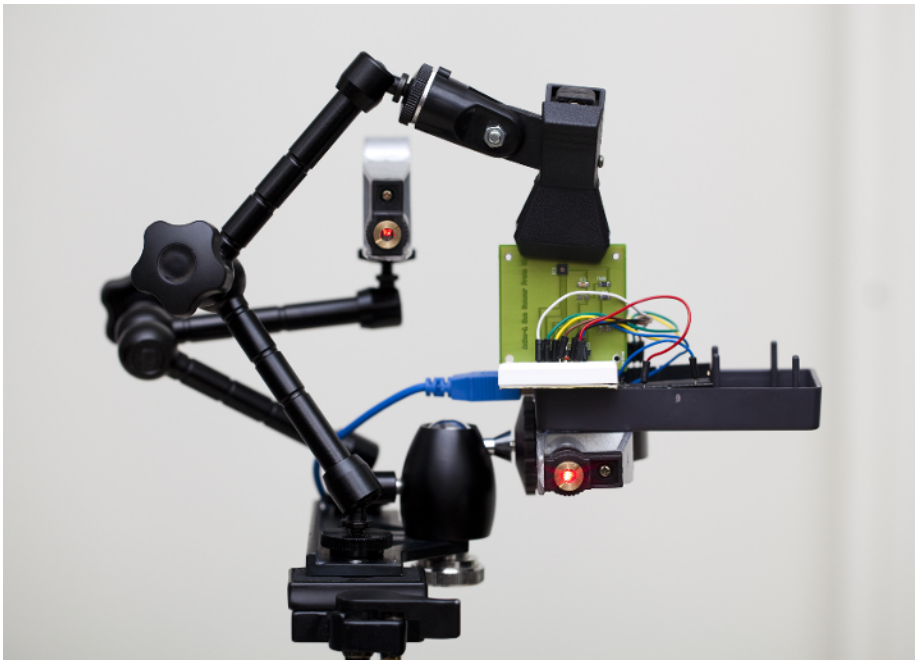

Figure P-1. *Finland launches its first satellite in 2014. We designed and built the sun sensor prototype with Arduino.*

How to Read This Book

When you get an idea, you can quickly build your first prototype with the help of this book. Instead of spending hours with component data sheets, you can simply pick a sensor and use ready-made breadboard diagrams and code. You can use sensors as building blocks for your project, but unlike construction kits such as Meccano or Lego, the possibilities with Arduino and Raspberry Pi are nearly endless.

If you know what you want to measure, you can easily find a sensor for it. The book is arranged by the real-life phenomena you can measure:

- Distance (Chapter 3)
- Smoke and gas (Chapter 4)
- Touch (Chapter 5)
- Movement (Chapter 6)
- Light (Chapter 7)

- Acceleration and angular momentum (Chapter 8)
- Identity (Chapter 9)
- Electricity (Chapter 10)
- Sound (Chapter 11)
- Weather and climate (Chapter 12)

You can also use *Make: Sensors* as a maker's coffee-table book: browse it to get ideas of what's available, and look for inspiration for new projects.

If you want to understand how sensors are connected to Arduino and Raspberry Pi, you'll enjoy the in-depth explanations. All the sensor code examples are fully self-contained, completely showing the interaction with the sensor. Understanding the sensors in the book helps you apply your skills to new sensors, even ones that aren't on the market yet.

When we chose the sensors for you, we picked a variety of useful and interesting sensors. We didn't just pick easy or difficult ones. This means you'll get to see solutions to the wide variety of challenges involved in connecting sensors to Arduino and Raspberry Pi.

In each chapter you'll find experiments, environmental experiments, and a test project:

1. Experiments give you quick instructions on how to use a single sensor with Arduino and Raspberry Pi. You can easily use these as building blocks for your own projects or just to see how the sensor works.
2. Environmental experiments let you play with sensors and monitor changes in the surrounding environment. This gives you insight into how sensors see the world and how they really work.
3. Sensors are more fun when you actually do something with the readings they give you. In test projects you'll build a device or gadget around one sensor. You'll learn how to use different outputs such as RGB LEDs, e-paper, and servo motors. Test projects also work as quick starting points for your own innovations.

Input, Processing, Output

Any robot or gadget you build must handle three things: *input*, *processing*, and *output*.

1. Because most of the devices you build won't have a keyboard or a mouse, sensors are your inputs. Take a quick look at the table of contents, and keep in mind that this is just a fraction of what's out there. There are countless sensors to measure everything you could imagine.
2. Processing happens in your program, running in Arduino or Raspberry Pi. In your program, you get to decide what happens next.

3. Outputs affect the world around the device. You could light an LED, turn on a servo motor, or play a sound. Those are three of the most common types of output, but there are others (for example, haptic feedback such as vibration, displaying something on an e-paper screen, or turning on a household appliance).

Protocols

A *protocol* defines how a sensor talks to the microcontroller board, such as Arduino or Raspberry Pi. The protocol defines how the wires should be connected and how your code should ask for measurements.

Even though there is a staggering amount of different sensors, there is a limited number of popular protocols. You'll learn each of the protocols as you work through experiments and projects, but here's an overview of what you'll be seeing.

You can get an overview of common sensor protocols in Table P-1.

Digital resistance
: Some sensors work like a button and have two states, on or off. These sensors are easy to read. The on state is represented when a voltage referred to as *HIGH* is applied to the microcontroller input pin. This is usually either 3.3 volts or 5 volts depending on the microcontroller board you're using.

Analog resistance
: Analog resistance sensors change their resistance in response to a physical change (such as turning the knob of a dial). Arduino and Raspberry Pi measure the changes in resistance by measuring the voltage level that passes through the sensor. For example, you can turn a potentiometer to make its resistance larger or smaller. These analog resistance sensors are very easy to make with Arduino. Raspberry Pi needs an external chip for measuring analog values. You'll learn to use the MCP3002 analog-to-digital converter to measure resistance with Raspberry Pi in "Experiment: Follow Movement with Infrared (IR Compound Eye)" on page 50. Most analog input sensors report their value using resistance, so they are analog resistance sensors.

Pulse width
: Some sensors report their value with a pulse width, or the period of time in which the pin is held HIGH. You use functions like `pulseIn()` or `gpio.pulseInHigh()` to read the length of the pulse. Because this is handled by a function, you don't have to get into low-level microcontroller operations such as *interrupts*; it is all handled by a library.

Serial port
: A *serial port* sends text characters between two devices. It's the same technique your computer uses when talking to Arduino over USB. You'll become quite familiar with the serial port when you print some messages to the Arduino serial monitor in various projects.

I2C

I2C is a popular industry standard protocol. It is commonly found inside computers and well known from Wii Nunchuk joysticks. I2C allows 128 devices to be connected to the same wires. In *Make: Sensors*, you'll get ready-made code and circuits for two sensors using I2C.

SPI

SPI is another industry standard protocol. You'll find it easy to use the code in this book for using an analog-to-digital converter on the Raspberry Pi. But creating your own code from scratch for new devices using SPI will be a bit more work.

Bit-banging

Sometimes, a sensor is unusual enough that a standard protocol won't work with it. In those cases, you need to craft up your own code to talk to that sensor. This is often called *bit-banging*, because you're manipulating the signal from the sensor, often at the bit level. You'll see an example of that later in the book in "Experiment: Is It Humid in Here?" on page 332.

As you play with the sensors, you'll get much more familiar with these protocols. Or, if you're in a hurry to put new sensors in your robots and innovative devices, you can just use the code in this book and look at the details later.

Table P-1. Sensor protocols, easiest first

Protocol	Example value	Arduino	Raspberry Pi Python	Example sensors
Digital resistance	1 or 0	digitalRead()	botbook_gpio.read()	Button, IR sensor switch, tilt sensor, passive infrared movement
Analog resistance	5%, 10%, 23 C	analogRead()	botbook_mcp3002.readAnalog(), chip	Potentiometer, light-dependent resistor, MQ-3 alcohol, MQ X gas family (smoke, hydrocarbon, CO…), FlexiForce pressure, KY-026 flame, HDJD-S822-QR999 color, LM35 temperature, soil humidity
Pulse length	20 milliseconds	pulseIn()	gpio.pulseInHigh()	Ping and HC-SR04 ultrasonic distance, MX2125 acceleration
Serial port	A9B3C5B3C5	Serial.read()	pySerial.read()	GT-511C3 Fingerprint scanner, ELB149C5M RFID identity
I2C	(2.11 g, 0.0 g, 0.1g), very precise values	Wire.h	smbus	Wii Nunchuk, MPU 6050 accelerometer and gyro combination, GY65 atmospheric pressure
SPI	57 deg, very precise values	Bit-banging	spidev	MCP3002 analog-to-digital converter
Bits encoded to very short pulses	53%	Bit-banging	Bit-banging	DHT11 humidity

Building Things Your Way

Most users won't find raw circuit boards and components compelling to play with. Making an attractive package for your gadget or robot makes a huge difference.

This book gives you one example for each project, but there's no need to follow our instructions blindly. Try different materials and use different tools.

How about using cardboard (Figure P-2), fabric (Figure P-3) or 3D printing (Figure P-4)?

Figure P-2. *Cardboard model. Photo from Ars Electronica in Linz (not made by us)*

Figure P-3. *Fabric robot. Photo from Ars Electronica in Linz (not made by us)*

Figure P-4. *3D Bender. Photo from Ars Electronica in Linz (not made by us)*

Trying out and learning new techniques makes the process of work more interesting, such as welding or making something out of clay between all the soldering (see Figure P-5).

Figure P-5. *Base model for animatronic gorilla head and latex skin made from it.*

We also use a lot of recycled materials in our own projects. Obviously they are cheap (free!) but they also give a unique look to a project.

Buying Components

If you need high quality components without fuss, pick a well-known shop, preferably in the Western world. If you want cheap components, look to Asia.

Quality shops mainly selling to makers include Maker Shed, SparkFun, Parallax, and Adafruit. Maker Shed is the shop from the publisher of this book. SparkFun sells a lot of breakout boards, which require you to solder in headers. Parallax created Basic Stamp, the previous generation of microcontroller boards for makers. Adafruit has a lot of parts, many designed by them. The SparkFun and Adafruit websites have a lot of information about their components, including tutorials.

These days, even big-name distributors like Element14 and RS electronics have broken into the Maker market. Finding parts from their huge catalogs is becoming easier, as they've started providing clear areas for Arduino and Raspberry Pi.

For some special parts and sometimes very cheap prices, Asia is the continent to go to. DealExtreme is very popular at the moment. Its shipping is slow and quality varies, but the prices are low and the assortment is wide. AliExpress is another Asian shop worth checking out.

Conventions Used in This Book

The following typographical conventions are used in this book:

Italic
: Indicates new terms, URLs, email addresses, filenames, and file extensions.

`Constant width`
: Used for program listings, as well as within paragraphs to refer to program elements such as variable or function names, databases, data types, environment variables, statements, and keywords.

`Constant width bold`
: Shows commands or other text that should be typed literally by the user.

`Constant width italic`
: Shows text that should be replaced with user-supplied values or by values determined by context.

This icon signifies a tip, warning, or general note.

Using Code Examples

You can download all the source code for this book from *http://makesensors.botbook.com*.

You can extract the ZIP package by double-clicking it, or by right-clicking and selecting "Extract" from the pop-up menu.

This book is here to help you get your job done. In general, you may use the code in this book in your programs and documentation. You do not need to contact us for permission unless you're reproducing a significant portion of the code. For example, writing a program that uses several chunks of code from this book does not require permission. Selling or distributing a CD-ROM of examples from MAKE books does require permission. Answering a question by citing this book and quoting example code does not require permission. Incorporating a significant amount of example code from this book into your product's documentation does require permission.

We appreciate, but do not require, attribution. An attribution usually includes the title, author, publisher, and ISBN. For example: "*Make: Sensors* by Tero Karvinen, Kimmo Karvinen, and Ville Valtokari. Copyright 2014 Tero Karvinen, Kimmo Karvinen, and Ville Valtokari, 978-1-449-36810-4."

If you feel your use of code examples falls outside fair use or the permission given here, feel free to contact us at *bookpermissions@makermedia.com*.

Safari® Books Online

Safari Books Online is an on-demand digital library that delivers expert content in both book and video form from the world's leading authors in technology and business.

Technology professionals, software developers, web designers, and business and creative professionals use Safari Books Online as their primary resource for research, problem solving, learning, and certification training.

Safari Books Online offers a range of plans and pricing for enterprise, government, education, and individuals.

Members have access to thousands of books, training videos, and prepublication manuscripts in one fully searchable database from publishers like O'Reilly Media, Prentice Hall Professional, Addison-Wesley Professional, Microsoft Press, Sams, Que, Peachpit Press, Focal Press, Cisco Press, John Wiley & Sons, Syngress, Morgan Kaufmann, IBM Redbooks, Packt, Adobe Press, FT Press, Apress, Manning, New Riders, McGraw-Hill, Jones & Bartlett, Course Technology, and hundreds more.

Maker Media has uploaded this book to the Safari Books Online service. To have full digital access to this book and others on similar topics from MAKE and other publishers, sign up for free at *http://safaribooksonline.com*.

How to Contact Us

Please address comments and questions concerning this book to the publisher:

MAKE
1005 Gravenstein Highway North
Sebastopol, CA 95472
800-998-9938 (in the United States or Canada)
707-829-0515 (international or local)
707-829-0104 (fax)

MAKE unites, inspires, informs, and entertains a growing community of resourceful people who undertake amazing projects in their backyards, basements, and garages. MAKE celebrates your right to tweak, hack, and bend any technology to your will. The MAKE audience continues to be a growing culture and community that believes in bettering ourselves, our environment, our educational system—our entire world. This is much more than an audience, it's a worldwide movement that Make is leading—we call it the Maker Movement.

For more information about MAKE, visit us online:

MAKE magazine: *http://makezine.com/magazine/*
Maker Faire: *http://makerfaire.com*
Makezine.com: *http://makezine.com*
Maker Shed: *http://makershed.com/*

We have a web page for this book, where we list errata, examples, and any additional information. You can access this page at:

http://bit.ly/make-sensors

To comment or ask technical questions about this book, send email to:

bookquestions@oreilly.com

Acknowledgments

The authors would like to thank Hipsu, Marianna, Nina, Paavo Leinonen, and Valtteri.

Raspberry Pi 1

We recommend you start with the Raspberry Pi Model B, which includes wired Ethernet and enough USB ports for a mouse and keyboard. This makes it much easier to get started.

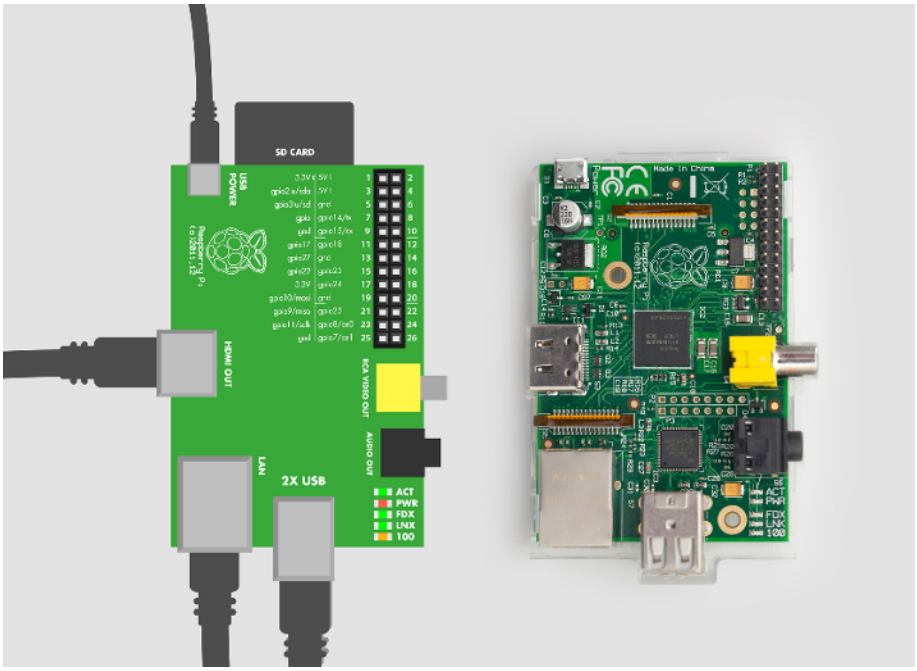

Figure 1-1. *Raspberry Pi peripheral connections*

Unless you buy your Raspberry Pi as part of a kit, it probably didn't come with an enclosure, but you can just put the bare board on your table for extra geek credibility. Or, if you have access

to a 3D printer, CNC, or laser cutter, you can find many enclosures to fabricate on *http://www.thingiverse.com*.

A 4 GB SD memory card is big enough to fit the operating system. A bigger card may be less susceptible to wearing out over time (more storage to allocate to wear-leveling), so if you have an 8 GB or bigger card, even better.

The Raspberry Pi can drive a full high-def display, and can even send sound over HDMI. Most likely, an HD television will work nicely as a display for your Pi.

Having a keyboard and a mouse will make it easy to get started. Raspberry Pi Model B has exactly two USB ports, just enough for the mouse and the keyboard.

If you want to add a USB WLAN adapter, you need a powered USB hub. See http://elinux.org/RPi_USB_Wi-Fi_Adapters for a list of WiFi adapters that are known to work with the Raspberry Pi. You'll be able to configure WiFi on your Pi by double-clicking the WiFi Config icon on the desktop after you install the operating system and boot to the graphical desktop environment.

The Most Expensive $35 (USD) Computer?

Buying all the cables, keyboard, mouse, and display can cost more than a couple of Raspberry Pis. If you don't already have all those parts gathering dust somewhere, it can be quite a lot for a tiny computer. Even so, it saves time (== money) to establish a comfortable development environment. Later, when your project is working, you can easily trim down the system to just the needed parts. As they say, Raspberry Pi is the only $35 computer that costs a hundred bucks.

If you decide to interact with your Raspberry Pi through SSH or VNC over the network, you only need to connect network and power and won't need the keyboard, mouse, or monitor except during the initial setup.

Raspberry Pi from Zero to First Boot

This chapter will get you up and running with the Raspberry Pi quickly. The first thing you need to do is to install Linux on the Raspberry Pi. It involves the following steps:

- Download and extract the installer to a formatted SD card.
- Insert the card into the Raspberry Pi and connect it to a keyboard, mouse, and monitor.
- Turn it on, choose what to install, and wait.

Once that's done, you are ready to boot the Pi into a graphical Linux desktop.

You'll need the following parts:

- Raspberry Pi Model B
- Micro USB cable and USB charger (or computer)
- 4 GB SD card
- Display with HDMI port
- HDMI cable
- USB mouse
- USB keyboard

Extract NOOBS*.zip

Download *NOOBS_vX_Y_Z.zip* (as of this writing, it was *NOOBS_v1_3_4.zip* but the filename may be different by the time you read this) from *http://raspberrypi.org/downloads*.

You can also find all the important links mentioned in this book on http://botbook.com, along with mirrored copies of some files.

Insert the SD card into your computer. Most SD cards are FAT32 formatted at the factory, so unless you're using an SD card that you've formatted yourself, extracting the NOOBS zip to the SD card is enough. After you unzip the file, make sure that the *bootcode.bin* file is in the root (top-level) directory of the SD card.

If you need to format the SD card, use the formatting tool from the SD Card Association.

In modern versions of Linux, Windows, and Mac you can just double-click or right-click the NOOBS zip file to extract it. For older versions of Windows, you can install 7zip to let you extract zip files.

Connect Cables

Connecting the cables is easy, because each cable will fit only its correct socket. Plug the mouse and the keyboard into the Raspberry Pi's USB ports. If you're using an HDMI monitor, connect an HDMI cable between the monitor and Raspberry Pi. If you're using an NTSC or PAL monitor, use a composite video cable to connect the yellow plug on the Raspberry Pi to the monitor.

Next, connect the micro USB cable to Raspberry to supply power. Plug that cable into either a computer's USB port or a 5 volt USB charger that provides at least 700 mA.

Boot and Install Raspbian

As soon as you connect power to the Raspberry Pi, it boots. No power switch is needed.

> *If nothing appears on the screen, you may need to select the right output mode for the Raspberry Pi. The default output mode is HDMI, but if you are connected via HDMI and see nothing, try pressing 2 on the keyboard connected to your Raspberry Pi to select HDMI Safe Mode. If you are connected via the composite (yellow) connector, press 3 for a PAL monitor or television, or 4 for an NTSC monitor or television.*

You are greeted with a graphical menu of different operating systems as well as language and keyboard type. Choose "Raspbian [RECOMMENDED]" (Figure 1-2) and select your language and type of keyboard you'll be using.

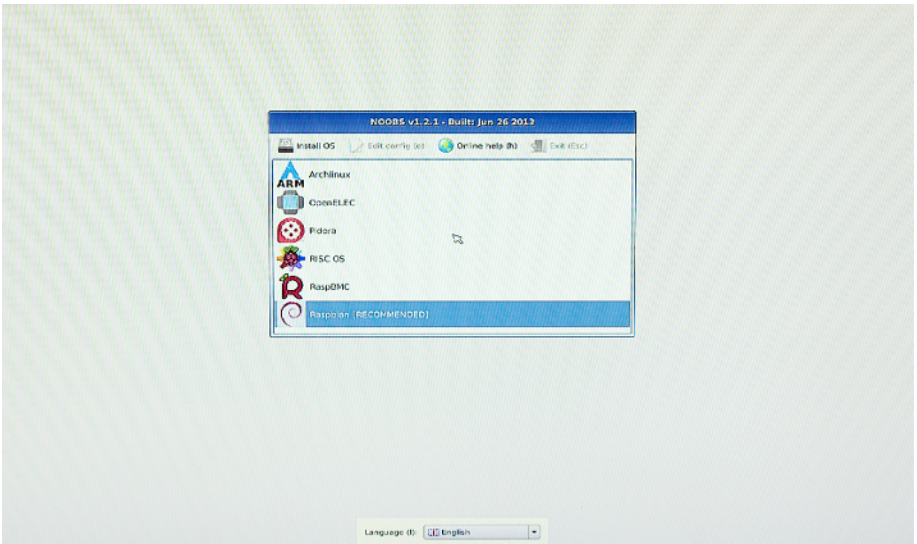

Figure 1-2. *Choosing an operating system*

If you know any Debian, Mint or Ubuntu, you will feel at home with this choice; if you don't, read on and you'll still feel at home! Raspbian takes a few minutes to finish installing (Figure 1-3). After the installer completes, it will indicate that it installed the operating system successfully. Press Enter or click OK to reboot.

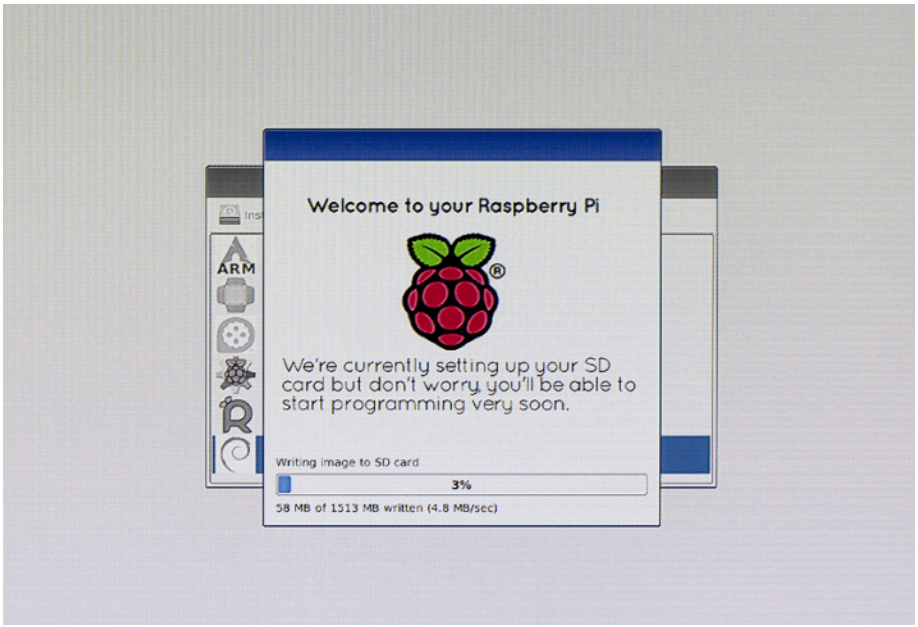

Figure 1-3. *Raspbian installs*

The Raspberry Pi configuration utility opens. Use arrow keys and Tab to navigate, and press Enter/Return to select an option, as shown in Figure 1-4.

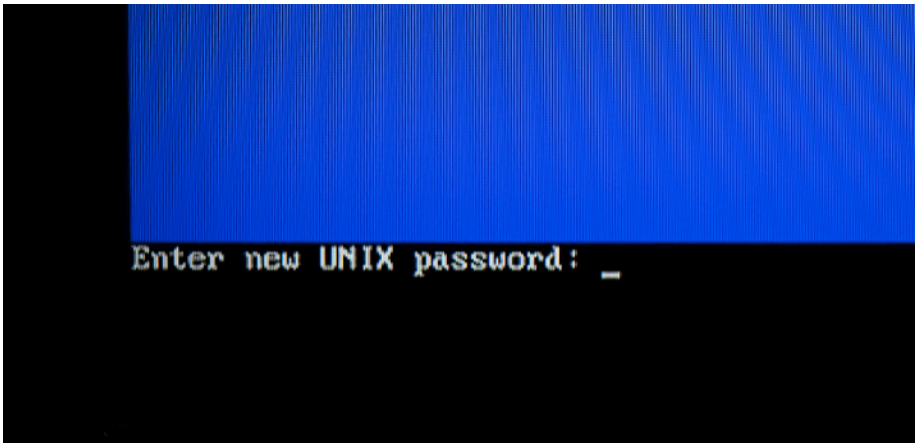

Figure 1-4. *Changing your password*

You'll want to enable the Boot to Desktop option. When you have finished changing settings, use Tab to select Finish and reboot when asked.

After the Raspberry Pi reboots, it will start up in a graphical desktop and will log you in automatically.

If you have chosen not to enable Boot to Desktop, you'll always start in the command-line interface. Log in as "raspberry" with password "pi" (unless you changed the password). After you log in, type `startx` to start the X Window System, which is the graphical desktop.

Welcome to Linux! You have now installed Raspbian on Raspberry Pi (Figure 1-5).

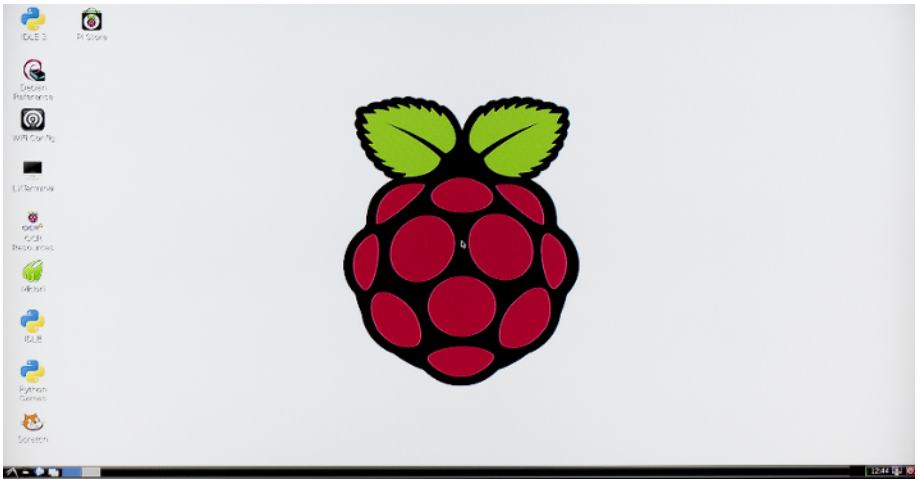

Figure 1-5. *Welcome to Linux*

> *To turn off your Raspberry Pi, double-click the Shutdown icon on the desktop. After it finishes the power down process, you should unplug the power.*

Troubleshooting Your Raspberry Pi Installation

Here are some solutions to common problems.

Is your card FAT32 formatted?

If you're having trouble booting from the SD card, it might not be formatted correctly. On Linux, use the built-in graphical partition editor (type `sudo gparted` to run it). Format the entire disk to FAT. You can run another tool with the command `sudo palimpsest` (or `sudo gnome-disks`) and for advanced users, `sudo parted` will get you into a classic command-line partitioning tool. On Windows and Mac, use the SD Association formatter (*https://www.sdcard.org/downloads/formatter_4/*). On Windows, choose the "format size adjustment" option in the formatter. On Mac, use "Overwrite Format."

Red power LED (PWR) not lit?

Power LED dim or blinking? Power LED briefly on but then goes off? Raspberry Pi is not getting enough power. Connect a USB power supply that can provide 5 volts and 1 amp or more. If you're powering the Pi via a laptop USB, switch to a desktop computer's USB or use a powerful mobile phone or tablet charger.

Black screen (but red PWR LED lit)?

It's possible that the Raspberry Pi can't read the bootloader from the SD card. Power down, remove the SD card, and reinsert it firmly. Verify that the first file used in the boot sequence, *bootcode.bin*, is on the top level of the SD card. If the problem persists, format the SD card and extract the NOOBS zip on it again. If that doesn't help, try a different SD card.

Four colored boxes on the screen?

The bootloader was read from the SD card, but the operating system *kernel.img* failed to boot. Format the SD card and extract the NOOBS zip again, or try another SD card.

Boot fails with some error messages?

Try disconnecting all USB devices, such as keyboard, mouse, and WiFi adapter (if you connected one). Leave only the SD card, display, and power. Remove and insert the SD card to ensure that it makes proper contact. If the problem persists, reformat the SD card and extract the NOOBS zip on it again.

Messed up your operating system?

If normal commands don't work, your screen is full of garbage, or Raspberry Pi stops working suddenly, don't worry. Hold down the Shift key on boot, and choose the option to reinstall Raspbian. This is quite fast and easy, but it deletes all data on the memory card. If it doesn't help, reformat the card, extract the NOOBS zip to it, and go through the installation process again.

No Internet?

If you have connected an Ethernet cable before boot, it should work like a charm in a typical network. Check that the link present (LNK) light is lit on the Raspberry Pi. If LNK is not lit, this means the Raspberry Pi thinks that one end of the Ethernet cable is not connected. Typically, you should see 100 (indicating a 100 Mbit/s or faster connection), and FDX (full duplex) should be lit. If LNK is lit and you still have problems, you may need some more advanced troubleshooting commands like the following: `ifconfig` (displays network adapter configuration), `route -n` (displays the routing table for the network connection), `cat /etc/resolv.conf` (shows the nameserver in use), and `ping -c 1 google.com` (tells you whether you can reach Google over the network).

If you get any error messages not mentioned here, try typing the exact error message into a Google search. Be sure to use the exact error message shown. If the message appears as the system is booting, take a photo of the screen with your camera or cell phone and transcribe it.

Feeling at Home in Linux

Raspberry Pi *is* Linux. Well, it's built on Linux. The name "Linux" is used to describe the operating system kernel and the operating system itself. The Linux operating system is composed of the kernel as well as thousands of utilities and applications from various sources.

Raspberry Pi is not a workstation-class device. In terms of computing power, it's more comparable to an entry-level portable tablet or a mobile phone. So even if you boot to its graphical desktop, don't expect to dump your laptop or desktop computer just yet. Extremely low computing power combined with little memory means that applications like LibreOffice and Mozilla Firefox won't be usable.

Command-Line Interface is Everywhere, Forever

Are you ready to feel the power of the *$* prompt?

The command line has stood the test of time, and it might be something you'll teach your grandchildren. The commands you use all the time, like *pwd*, *ls*, or *cat*, existed long before Linux was invented. (Linux was written by a Finnish student in Helsinki, just five kilometers from the botanical garden where I'm writing this.) Power users on both OS X and Windows dive down into the command line when they need to do something they can't do with a mouse alone.

Most of the commands you'll use with Raspberry Pi are the same you'd use on a Mac or Linux computer, and are similar even to the command-line tools on Windows.

As you may know, most of the world's servers run Linux. Google, Facebook, Amazon, and most supercomputers run Linux. Web servers don't run a graphical user interface, so that's why most programmers and system administrators must know how to use the command-line interface (CLI). You can use these commands on a Linux desktop or laptop, too.

The CLI Really Is Everywhere

If your smartphone is like many smartphones, it runs Linux, and you can run some of these commands on your phone. Android supports a limited subset of commands even without jailbreaking or rooting (you'll need a shell environment app such as ZShaolin). Like Apple's computers, the iPhone and iPad are based on a distant cousin of Linux, BSD, and you can access the CLI on a jailbroken iPhone.

The CLI is easy to automate. Whatever commands you can type on the command prompt (also known as the *shell*), you can also turn into a program. You just put each command into a text file, one command per line (we show you how to do this in the next section). Then you can run that program with `dash` *`filename`* (on Raspberry Pi, the shell is named *dash*). Whenever you realize you're typing the same 10 commands over and over, it's time to turn them into a script.

Even though CLI scripts are so easy to write that they are good for quick-and-dirty one-off scripts, they are also suitable for serious, mission-critical applications. For example, Linux uses scripts for booting and controlling daemons (server applications that run in the background).

Looking Around

After you boot your Raspberry Pi into the graphical user interface, you can start up the command-line interface by double-clicking the LXTerminal icon on your desktop.

Besides being great for browsing the Web, the Midori web browser is useful if you want to copy and paste some sample code from http://botbook.com.

The prompt *$* means Linux is waiting for your command. Type *pwd* to print your current working directory (where you are in the file system). Linux answers with a path, such as */home/pi/*.

To list the files in the working directory, type *ls* and press Return or Enter. These are the files you can readily manipulate. Whenever you get an error like *No such file or directory*, just use *pwd* to see where you are and *ls* to see what files are in the working directory.

You will always need to type Return or Enter after you type a shell command such as `ls`. Before you press Return/Enter, you can use the Backspace and arrow keys to edit the command.

To edit (or create) a file called *foo.txt*, use `nano foo.txt`. Type some text, then press Control-X to save it (when prompted whether to save, type y. When asked for the filename, just press Enter or Return). To edit the file some more, type the command `nano foo.txt` again.

Text Files for Configuration

In Linux, most things are text files. With just the commands listed in the previous section, along with the *sudo* command, a skillful hacker can change many system settings. All configuration in Linux is stored in text files. System-wide configuration files are in the */etc/* directory and per user configuration is in the user's home directory, */home/pi/*.

Even the Pi's input and output pins can be manipulated by editing text files under the */sys/* directory. Later in this book, you will learn to connect sensors and LEDs to Raspberry using GPIO pins.

If you connected the Pi to the network with an Ethernet cable before booting, your Raspberry Pi should already be connected to the Internet. Test this by typing `ping www.google.com`. You'll see a result every second. Kill the command after a while by typing Control-C. If the network is working well, it should report a packet loss of 0%. You can also download whole web pages with commands like `curl botbook.com` and `wget botbook.com`.

sudo Make Me a Sandwich

Linux is well known for its robust security model. The *separation of user privileges* is one of its key features. Normal users can make changes only to files that affect their working environment. This means that they can modify files only in their home directory (*/home/pi/*) and in temporary working directories such as */tmp/*.

The super user, or root user, is all-powerful and can change any file on the system. To use root's privileges, put `sudo` in front of a command. Putting `sudo` in front of the command runs that commands under the privileges of the root user.

For example, Raspbian's package manager makes it very easy to install additional software, but you need to use root's privileges to install anything. Before you install new software, you need to update the list of what's available with `sudo apt-get update`. This requires a network connection because all the software packages are on a file server.

Many Linux and Unix systems (such as OS X) are configured such that you need to type your user password when you use sudo. This is an extra safety step. By default, Raspberry Pi's Raspbian operating system does not ask for this. Be careful using sudo, because you can easily make mistakes that would render the operating system unbootable.

You can install any program from a repository by specifying its package name. To install ipython (an interactive tool for experimenting with the Python language and data visualization), use `sudo apt-get -y install ipython`. The `-y` parameter tells the package manager to assume a "yes" answer to any questions it asks. The package manager (apt) does everything for you.

After a moment, you can run the newly installed package by typing `ipython`. Any python command will work here, but `exit()` will get you back to the command prompt ("$"), where you can type shell commands.

Different prompts are one way of indicating which program you're talking to. When you see the shell, or command prompt ($), you can type the name of built-in shell commands as well as programs installed on your Raspberry Pi. When you see another prompt, such as the ipython prompt (`In [1]:`), you can type Python commands.

Installing daemons (also known as servers) is just as easy. Try installing the most popular web server in the planet, Apache, with this command: `sudo apt-get -y install apache2`. When it finishes, you can pull down your web server's raw home page with `curl localhost`. To browse your Apache web server from another computer, determine your IP

address with the output of the `ifconfig` command and type that address into a web browser on another computer connected to your network.

You will see at least two adapters listed in the output of `ifconfig`. Use the Ethernet adapter's (`eth0`) address, or if you're using WiFi, use that adapter's address.

Phew! That was a lot of command line if you are (were) a beginner. To give yourself a well-earned break, power off the Raspberry Pi by typing the command `sudo shutdown -P now`. Just like you do on your workstation, you must shut down properly so that data gets actually written to disk before you power off. Once it shuts down, you can unplug it from the USB power source. When you're ready to use it again, plug it back in.

Still wondering what we're talking about with the sudo sandwich? Try searching the Web for "xkcd sudo sandwich." You can use the rest of your break to read some of the other comics there.

See Appendix A for a cheat sheet of Linux commands.

One Handed Wonder

We sometimes ask Linux students to guess how long it takes me to type "supercalifragilisticexpialidocious.foo.bar.txt.botbook.com.pdf.cc" with my left hand. After some guesses, we'll test it. As a preparation, I create the file with this command (type it all on one line with no space between `foo.` and `bar`):

```
$ nano
supercalifragilisticexpialidocious.foo
.bar.txt.botbook.com.pdf.cc
```

I count down to three, and type this in less than five seconds:

```
$ ls
supercalifragilisticexpialidocious.foo
.bar.txt.botbook.com.pdf.cc
```

In reality, I just need to type `ls s` and press Tab. The shell will complete the word you're typing after you press Tab. You should use this feature all the time. It's weird how well Tab works. It can guess not only directories and files but also network servers after commands like `ssh` and `ping`. Because Tab completes only correct filenames, you won't be making many typos.

Connecting Electronics to Raspberry Pi Pins

The Pi's GPIO (general-purpose input and output) pins let you connect electronic components directly to the Raspberry Pi. These pins are called general purpose because you can decide what purpose they serve, and you can even configure the same pin to be an input or an output at different times. Throughout this book, you will learn the following:

- Digital output (turn an LED on and off)
- Digital resistance (detect whether a button is pressed or a sensor is active)

- Digital input for very short pulses (used by sensors such as a distance sensor)
- Analog resistance (analog resistance sensors for pressure, light, temperature)
- Industry standard protocols, such as I2C and SPI (used by the Wii Nunchuk and analog-to-digital converters)

Unlike most other tutorials available at the time of writing, we'll teach you how to use digital input and output without having to invoke root privileges all the time. This provides security and stability benefits.

For digital input, you'll learn to use the Pi's internal pull-up resistor, so that your circuit uses a minimum number of components.

For measuring analog resistance, you'll use an external analog-digital converter chip.

Many components in everyday products communicate over industry standard protocols, such as I2C and SPI. You will see examples of both protocols later in the book.

But first, let's show you how to use the most basic form of digital output.

Hello GPIO, Blink an LED

In this "Hello GPIO World" example, you'll attach a new LED to Raspberry Pi and blink it.

We start all of our projects with a "Hello World" on any platform, on any language. So whenever you are about to build something more complicated, it's a good idea to build this "Hello GPIO World" first. This lets you confirm that the hardware and software are functioning at the most basic level. If your "Hello World" example doesn't work, you need to fix it before trying something more complicated.

Parts needed:

- Raspberry Pi
- Female-to-male jumper wires, black and green or yellow color
- Solderless breadboard
- 470 Ohm resistor (yellow-violet-brown stripes)
- An LED

If you don't have all the parts at hand, see "Troubleshooting" on page 17 for suggestions. For wires, a female connector is one that has a receptacle and a male connector is one that has an extending pin. You will often see female abbreviated as F and male abbreviated as M, along with other characteristics of the connector, such as its length. The type of female-to-male jumper wires you need are typically sold as "male to female breadboard jumper wires."

GPIO pins are not protected against overcurrent (Figure 1-6). Unlike Arduino, Raspberry Pi is not forgiving on user mistakes. Data pins can take only 3.3 V. Connecting a +5 V pin to a data pin can easily break your Raspberry Pi, or at a minimum, render that pin unusable. Double-check anything that you build on a breadboard, and be very careful where you place any test probes if you use a multimeter with the GPIO pins.

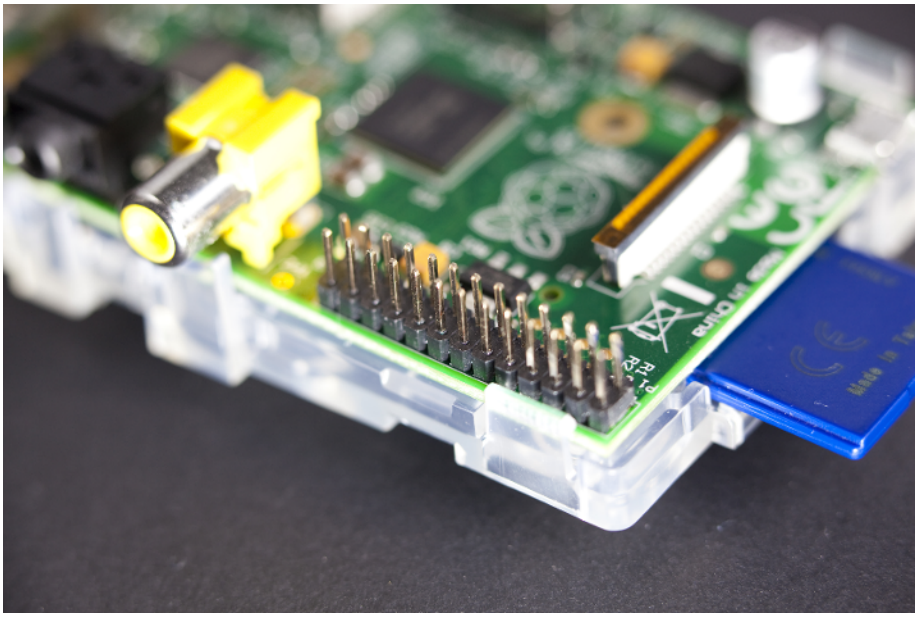

Figure 1-6. *GPIO header*

Building the Circuit

The circuit is simply an LED with a current limiting resistor, connected in series between GPIO pin 27 and ground (Figure 1-7). Connect the short (negative) lead of the LED to the black wire, and the long (positive) lead to the resistor. You should make these connections while the Raspberry Pi is shut down and unplugged.

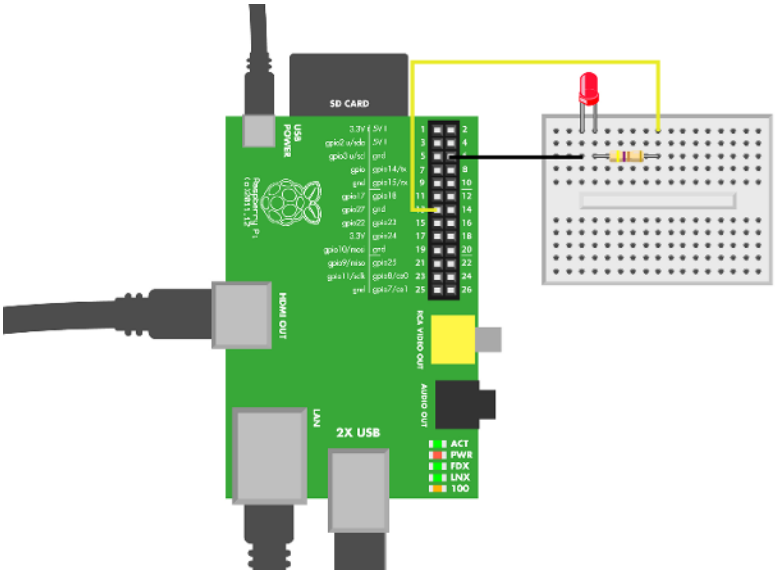

Figure 1-7. *LED hello breadboard*

Build the circuit on the breadboard (Figure 1-8). Double-check all connections to avoid breaking your Raspberry Pi. After you're sure you've connected the wires correctly, you can power up your Pi.

To learn about how the Raspberry Pi's pins are numbered, read on.

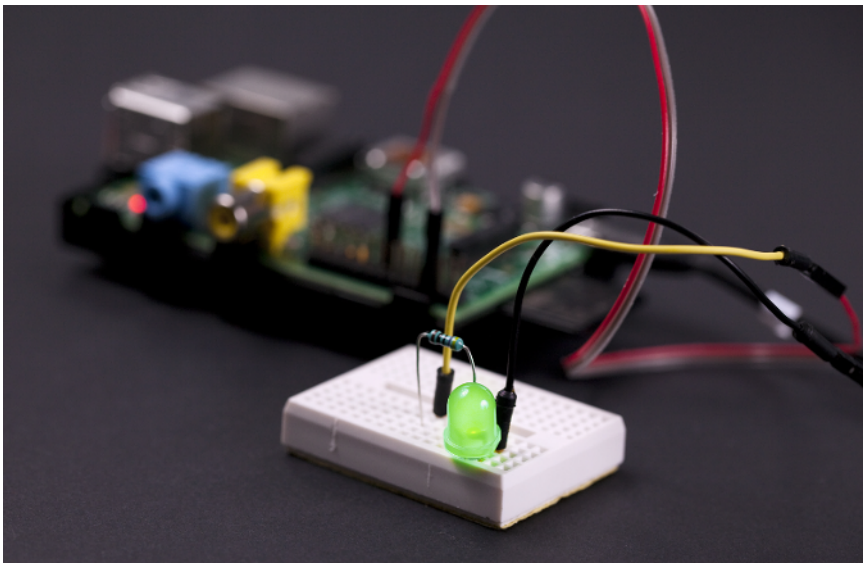

Figure 1-8. *Hello GPIO!*

Two Numbering Systems: Purpose and Location

Each GPIO pin has two numbers: purpose and physical location. To find the correct pin in the GPIO header, you should learn to convert between the two numbering systems.

Locate the GPIO pin header on Raspberry Pi (Figure 1-6). When converting between purpose and physical location numbers, use the numbering diagram (Figure 1-9).

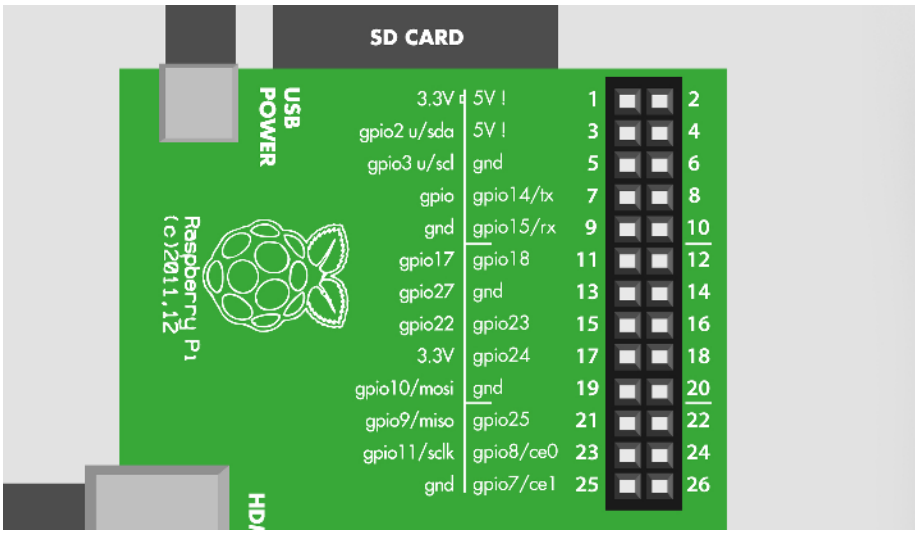

Figure 1-9. *GPIO numbering*

The left side of the numbering diagram shows the purpose of the pins (GND, GPIO 27). The right side of the numbering diagram shows the physical location (1 to 26).

The physical pin header has a running number, from 1 to 26. There is a tiny white box drawn near pin one, and it is also labeled P1. This physical pin header number tells you where to insert the jumper wire. These physical numbers are also known as *board numbers*.

There is also another, purpose-based numbering. The code you'll be writing uses GPIO numbers, such as GPIO 27, which corresponds to physical pin 13 (which is the pin you connected to the resistor with a wire). Circuit diagrams are likely to refer to ground (GND), +5 V, and +3.3 V, and Figure 1-9 will help you find them. Some pins can have multiple purposes. For example, GPIO 10 can be used for the SPI bus. These purpose-based numbers are also known as *BCM numbers*.

To help you get started, the two pins used here are listed in Table 1-1. For future projects, learn to find these numbers in the numbering diagram (Figure 1-9).

Table 1-1. Pins used in Hello GPIO

GPIO pin (BCM, used in code)	Physical location (Board)
GPIO 27	13
GND	6

Controlling GPIO Pins from the CLI

Let's see how to use the command-line interface to control the GPIO pins we just connected. First you'll try it out as root, then advance to using it without needing to invoke sudo privileges each time.

Text files control everything in Linux. The kernel GPIO driver (a piece of software that controls how Linux talks to GPIO pins) makes the GPIO pins available to you through the virtual */sys/* file system. To control GPIO, you simply edit or otherwise make changes to these text files.

You don't need a graphical user interface at this point, so we can do everything from the LXTerminal command-line interface. Double-click its icon to launch it.

If you configured your Raspberry Pi to not start in the graphical interface, or if you've used SSH to connect to it remotely from another computer, you don't need to start up the graphical desktop system to try this out.

To turn on the pin, you first *export* it, configure it for "out" mode, and write the number "1" to it. All this is done by editing text files.

Writing to Files Without an Editor

Earlier, you saw how to modify the contents of a file with the nano text editor. Here's how you can modify files without needing the editor.

First, let's look at how to display text. You can print text to the terminal using this command (don't type the $; it indicates the shell prompt):

```
$ echo "Hello BotBook"
```

Any text you display this way can also be redirected to overwrite a file (be careful using this, and don't overwrite anything important):

```
$ echo "Hello BotBook" > foo.txt
```

If the file *foo.txt*, did not exist, it will be created. If it existed, it was overwritten without warning. You can use the > (redirection) operator to send the output of almost any command to a text file. You can see what's inside that file with the `cat` command:

```
$ cat foo.txt

Hello BotBook
```

Couldn't you just use *nano foo.txt*? Yes, of course you could. But it requires more typing, and it's not as easy to automate.

Light Up the LED

We'll use sudo the first time through lighting an LED. Does it feel wrong to use root for non-administrative tasks? If not, it should; if you mistype a command with sudo, you can potentially render your operating system unusable and will need to reinstall it.

Don't worry, you'll use sudo just to initially try it. Later, you'll fix Linux's file permissions and interact with the files that control GPIOs as a normal user.

Type `sudo -i` to get a *root shell*. Use root shell only as long as required by this task, and type `exit` when you're done. You'll notice that your prompt changes to a hash mark, #. Be careful what you type as root, as mistakes can break your operating system.

After you start the root shell, export GPIO pin 27 to be able to manipulate it (remember, just type the text to the right of the # shell prompt):

```
# echo "27">/sys/class/gpio/export
```

This creates the new virtual file you'll use to blink the LED. Next, set pin 27 to *out* mode, so that you can turn it on and off.

```
# echo "out" > /sys/class/gpio/gpio27/direction
```

Now turn on the pin:

```
# echo "1" > /sys/class/gpio/gpio27/value
```

Your LED should light up now. Once you have enjoyed the light for a while, turn it off:

```
# echo "0" > /sys/class/gpio/gpio27/value
```

Did your LED light up? Hello, GPIO world!

Once you are done with playing root, remember to type `exit`. Your prompt should return to the dollar sign symbol $, indicating that you are working with the shell as a normal user.

Troubleshooting

If you had trouble setting things up, check out the following:

Lacking a 470 Ohm resistor?

Use any resistor in the hundreds of ohms range to take only a minor risk with your board. The resistor is just used for limiting current through the LED, to avoid frying the LED or the pins on the Raspberry Pi. If you don't mind the LED being a bit too dim or slightly overworked, you can use any resistor value between 100 Ohm and just under 1 kOhm. (Such resistors will have a brown third stripe; see "The Third Stripe Rule" on page 19.)

Can't find female-male jumper wires?

You can use an old 40-wire IDE hard drive cable (not 80-wire). Put the female side of the cable into GPIO header. Cut the other end. Pay attention to wire numbers. If you don't need all the wires, don't strip them. If you strip them all, at least use tape to cover +5 V wire ends. See *http://bit.ly/1f0GWfV* for a tutorial on using an old IDE cable. There are also many ready-

made connectors available such as this one from Adafruit. MAKE's Raspberry Pi Starter Kit, which includes a Raspberry Pi, includes the cable and a breadboard-ready breakout board.

My LED does not light

Did you insert it the right way? LEDs have polarity, and if you insert them backward, they won't light up. The long positive leg of the LED goes to GPIO 27 (through the resistor). The negative side of the LED has a flattened area in the plastic body of the lamp, and a shorter lead than the positive lead. The negative side of the LED goes to ground. The arrow in Figure 1-10 is pointing to the cathode. You may need to look closely to see the flattened part at the base of the LED package. If you have trouble seeing it, grab an LED and check it out yourself.

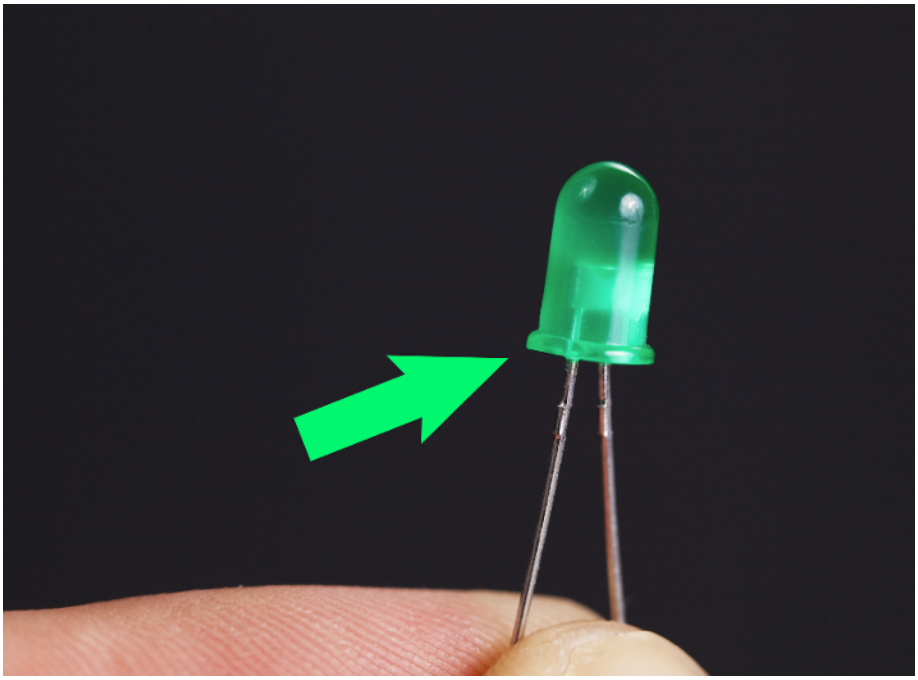

Figure 1-10. *LED polarity*

Nothing happens when I type root commands

Don't type the prompt (#); that is shown in the previous examples to show you what you'll see on your Raspberry Pi screen. The shell shows that prompt when it is waiting for you to type a command while in a root shell. You wouldn't type the normal user prompt "$" either, would you? The hash symbol has special meaning to the shell: it means that the rest of the line is a comment (a human-readable remark you leave in a program), and is ignored by the shell.

I can't access the shell

Review the instructions in "Looking Around" on page 9.

I get an error message

Copy and paste the exact error message into a search engine like Google. Also try putting quotes around your error message. You can try "Raspbian" or "Raspberry Pi" as additional search terms. When you solve your problem, it's a good idea to blog about it and include the exact error message in your blog post so other people searching can find it.

The Third Stripe Rule

The third stripe trick identifies a resistor easily. You can quickly find a resistor with approximately the value you are looking for by looking at the multiplier ring. The multiplier ring is usually the third one (with five band resistors, the multiplier is the fourth ring). In many connections, the exact value of resistor is not important as long as it's near enough.

For example, 10 kOhm resistor has kilo (thousand, 10^3) multiplier, 3: orange. A resistor from 10 MOhm to 50 MOhm has mega (million, 10^6) multiplier, so the third stripe is 6: blue.

For calculating all the color bands, search the Web for "resistor color code calculator." To double-check your resistor identification, measure the resistance with a multimeter.

GPIO Without Root

Avoiding root privileges will make the system more secure and more stable. For example, think about a program that serves sensor data to web. Would you run a program that strangers can connect to as root?

Before you begin, make sure that you can turn the LED on and off ("Light Up the LED" on page 17).

In modern versions of Linux, devices attached to your system are controlled by *udev*. Udev is a rule-based system that can run scripts when devices are plugged in. If you have developed apps for Android under Linux, you may have created a udev rule to modify permissions when your cell phone is plugged into the computer. If you have developed with Arduino on Linux, you have probably added yourself to the *dialout* group to get access to serial over USB.

Normally, the GPIO files in */sys/class/gpio/* are owned by the user "root" and the group "root." You'll see how to write a udev rule to change the group to "dialout." You'll then allow that group to read and write the files under */sys/class/gpio/*. Finally, you'll make the folders' group *sticky*, so that any newly created files and folders under it will also be owned by the "dialout" group.

All system-wide configuration in Linux is under */etc/*. Not surprisingly, udev configuration is in */etc/udev/*. First, open an editor with sudoedit so you can create a new rule file:

```
$ sudoedit /etc/udev/rules.d/88-gpio-without-root.rules
```