

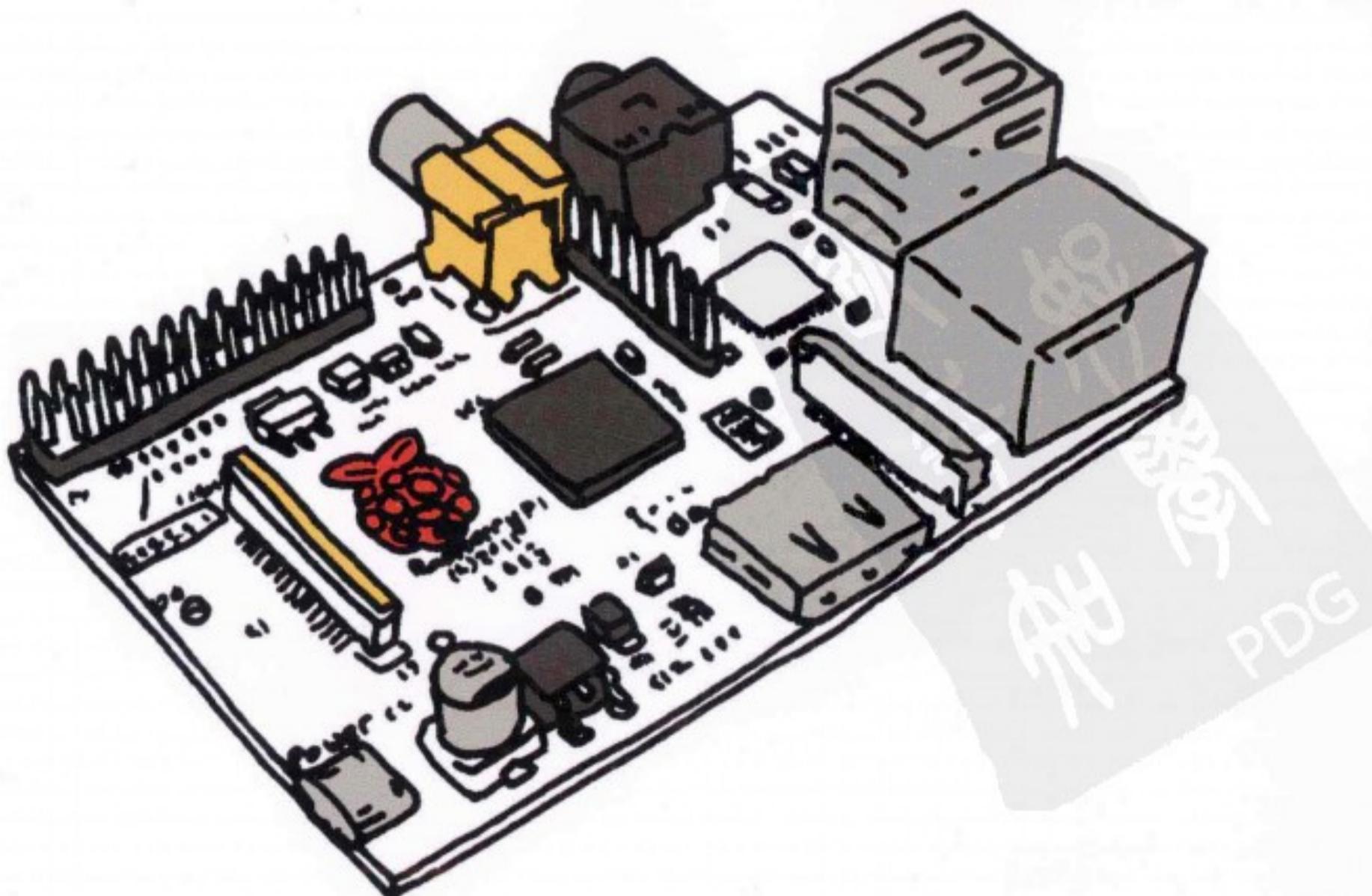
35美元的
ARM架构
Linux
微型电脑

爱上

Raspberry Pi

Getting Started with Raspberry Pi

〔美〕 Matt Richardson 著
Shawn Wallace
李凡希 译



O'REILLY



科学出版社

Make:
makezine.com

爱上Raspberry Pi

用一台信用卡大小且仅售35美元的电脑能做些什么？可以做任何你所能想到的事！你可以用它来编写程序，也可以把它当成电子积木。本书将手把手地教会你如何去发掘这个体积虽小却功能强大的平台的内在价值。

- » 熟悉Raspberry Pi的主板结构与硬件功能
- » 学习Linux的基本知识，玩转操作系统
- » 学习Python与Scratch的基础知识——编写出你的第一个计算机程序
- » 通过Pygame框架绘制图形、播放声音、处理鼠标事件、编写多媒体应用
- » 使用Raspberry Pi输入输出接口完成电子实验
- » 结合Arduino与Raspberry Pi，各取所长
- » 在你的程序中使用USB摄像头等各种硬件
- » 用Python在Raspberry Pi上搭建你自己的Web服务器

O'REILLY
oreilly.com.cn

Make:
makezine.com

O'Reilly Media, Inc.授权科学出版社出版

此简体中文版仅限于中国大陆（不包含中国香港、澳门特别行政区和中国台湾地区）销售发行 www.sciencep.com
This Authorized Edition for sale only in the territory of People's Republic of
China (excluding Hong Kong, Macao and Taiwan)

科学出版社 东方科龙公司
联系电话：010-82840399
E-mail: boktp@mail.sciencep.com
有关网址：<http://www.okbook.com.cn>

销售分类建议：工业技术/电子技术

定 价：39.80元



爱上Raspberry Pi

〔美〕 Matt Richardson
Shawn Wallace 著
李凡希 译



科学出版社

内 容 简 介

Raspberry Pi 是一台价格不到300元的卡片式电脑，它的体积虽然只有信用卡般大小，但具备强大的功能。本书将从在Raspberry Pi上安装、配置和开机启动Linux操作系统开始，深入浅出地介绍Raspberry Pi的各种特性。不仅介绍Raspberry Pi自身的底层GPIO接口编程技术，还将指导你把Raspberry Pi与Arduino结合在一起，设计出可以通过网络远程控制的电子作品。此外，你还将学习Pygame多媒体编程，并用Scratch完成一个简单的小游戏。

本书适合Raspberry Pi爱好者阅读，也可作为高等院校电子信息、计算机等相关专业的师生参考用书。

图书在版编目（CIP）数据

爱上Raspberry Pi/（美）Matt Richardson, Shawn Wallace著；
李凡希译. —北京：科学出版社，2013.10

ISBN 978-7-03-038196-5

I. 爱… II. ①M… ②S… ③李… III. Linux 操作系统-程序设计 IV. TP316.89

中国版本图书馆CIP数据核字（2013）第169518号

责任编辑：喻永光 杨 凯 / 责任制作：魏 谦

责任印制：赵德静 / 封面设计：Randy Comer 张 健

北京东方科龙图文有限公司 制作

<http://www.okbook.com.cn>

科学出版社出版

北京东黄城根北街16号

邮政编码：100717

<http://www.sciencep.com>

北京佳艺恒彩印刷有限公司 印刷

科学出版社出版 各地新华书店经销

*

2013年10月第 一 版 开本：A5 (890×1240)

2013年10月第一次印刷 印张：7 1/4 插页 1

字数：170 000

定价：39.80元

（如有印装质量问题，我社负责调换）

© 2012 by O'Reilly Media, Inc.

Simplified Chinese Edition, jointly published by O'Reilly Media, Inc. and China Science Publishing & Media Ltd.(Science Press), 2013. Authorized translation of the English edition, 2012 O'Reilly Media, Inc., the owner of all rights to publish and sell the same.

All rights reserved including the rights of reproduction in whole or in part in any form.

英文原版由 O'Reilly Media, Inc. 于 2012 年出版。

本中文简体字版经 O'Reilly Media, Inc. 授权中国科技出版传媒股份有限公司（科学出版社）于 2013 年独家出版、发行。

版权所有，未经书面许可，任何人不得以任何形式复制本书的任何部分。



推荐序

刚看到“树莓派”这个词，你会不会认为它是一款好吃的蛋糕呢？想当然你会问“度娘”，结果有将近 30700 条相关信息。Raspberry Pi（中文名为“树莓派”，简写为 RPi）是一款信用卡大小的卡片式电脑，是为学生计算机编程教育而设计的。自 2012 年问世以来，受众多计算机发烧友和创客的追捧，曾经一“派”难求。别看其外表“娇小”，内“心”却很强大，视频、音频、网络等功能通通皆有，可谓是“麻雀虽小，五脏俱全”。这么棒的东西能用来做什么呢？《爱上 Raspberry Pi》将会告诉你答案。

2008 年我开始接触 Arduino，这位能征善战的“故友”大家都很熟悉。如今在这“嵌入式”的天下，8 位微控制器在功能应用方面略显单薄，Raspberry Pi 的问世不仅能解决此类问题，与 Arduino 的结合还会发挥非凡的创造力。目前关于 Raspberry Pi 的技术论坛和相关资料不少，但杂乱无章，初学者无法系统学习。《爱上 Raspberry Pi》的出版将会给广大电子爱好者带来莫大的帮助。同时也会促使这股“树莓”风席卷中国。

本书全面讲解了 Raspberry Pi 硬件组合和操作系统的使用，还有与 Arduino 如何交互，深入剖析当下热门应用案例，具有很强的实操性，还有大量珍贵代码。为与广大“派”友交流互动研习本书，本人也将开通“Raspberry Pi”大制作网站（<http://www.iraspberrypi.cn>），将书中教学案例拍成视频与大家分享。预祝每一位认真阅读本书的人早日成为有“派”一族，未来电子领域的佼佼者！

于欣龙
奥松机器人创始人、资深创客

致中国读者

当我们刚开始编写《爱上 Raspberry Pi》这本书时，我们并不只是希望通过这本书教会读者如何把 Raspberry Pi 运行起来，而是希望能引导读者用它做出更多好玩、实用的东西。所以在这本书中，我们设计了很多实例，通过一步步地讲解，让读者可以在实践中逐步了解整个系统中的每个部分。希望通过这些实例，在读者面前展现出 Raspberry Pi 的潜力并激发读者的创造力，在 Raspberry Pi 上开发出更多创新的作品。

本书的英文版出版后，我们很高兴收到了许多读者的反馈。有人告诉我们，他修改了书中介绍的“Web 台灯”实例，并以此为基础做出了一个可以通过网络控制的咖啡机！

我们被 Raspberry Pi 的巨大潜力深深打动，并把我们的激动之情倾注到了这本书中。我们希望能通过本书的中文版，与中国读者一起分享我们的激情。希望大家能从本书中学到一些知识，并获得一些新的启示。

Matt Richardson
2013 年 4 月 15 日于纽约

译者序

2005 年时，尼葛洛庞帝（Nicholas Negroponte）教授与麻省理工学院多媒体实验室所提出的 One Laptop per Child (OLPC) 100 美元笔记本电脑设想深深地吸引了我，虽然 100 美元的电脑在当时似乎只是一个美好的梦想。然而，仅仅过去了短短的 7 年时间，我们就看到了信用卡大小的 Raspberry Pi 以 35 美元的售价在很短时间内风靡全球，截至 2013 年 3 月，已经售出了超过 100 万台 Raspberry Pi。

Raspberry Pi 不仅仅是一台便宜的微型电脑，也仅仅是创客们手中的创新玩具——如果只是用这样的标准去要求它，市面上还能找到性价比更高的产品。Raspberry Pi 从设计之初就承载了“教育”这一重要的理念，通过这样一台便宜的设备，能鼓励学生在上面大胆尝试，提高动手能力；通过使用 GPIO 接口完成电子实验，也可以更好地帮助学生理解电子电路与计算机的一些底层知识，培养编程能力。在这样的一个设计理念的指引下，一个成熟的 Raspberry Pi 社区很快地在全球范围内成长起来。这也正是 Raspberry Pi 的成功之处。

这本《爱上 Raspberry Pi》只是一本薄薄的小册子，但是“麻雀虽小，五脏俱全”，作者把很多琐碎的知识点有机地组织在一起，让读者不需要太多背景知识也可以轻松地读完全书。它很好地贯彻了 Raspberry Pi 的“教育”理念，用非常精炼的内容带领读者全方位领略 Raspberry Pi 的魅力，并手把手指导读者完成很多有意思



实例，从实践中学习知识、积累经验。在每一章的最后，还列出了详尽的参考资料，指引读者自己进一步深入学习相关知识。

在全书的翻译过程中，好友云汉、张志博、贾征细致地审阅了全书的内容，并指出很多可以改进的地方。阿里巴巴集团的“倒立工坊”创客小组的同事也给出很多有用的建议。在此向他们表示诚挚的谢意。同时也要感谢我的家人对我的支持，能让我有足够的时间放在翻译工作上，保证了在最短的时间内把最高质量的作品奉献给大家。从事多年语文教学工作的父亲虽然没有办法完全理解本书中所介绍的技术细节，但还是帮忙审阅了全书的内容，在词汇和句子的组织上给出很多很好的建议。最后还要感谢科学出版社的喻永光编辑，他积极调动了各种资源并开展工作，让这本书能更快与广大读者见面。

在本书即将付梓之际，兴奋之余，我还是感受到一丝惶恐。虽然我以前也曾经翻译过很多软件界面和在线文档，但翻译一本完整的书对我来说还是第一次。虽然自己已经反复校对了多次、字斟句酌，但纰漏和欠缺之处在所难免，还请广大读者不吝赐教和批评。有关本书及 Raspberry Pi 相关的任何问题和建议都可以通过电子邮件发送到 rpi@freemindworld.com。我还为本书建立了一个专题网页 (<http://rpi.freemindworld.com>)，你可以在上面找到勘误表和一些相关的参考资料。

李凡希
2013年4月20日于杭州

前 言

很容易理解为什么当 Raspberry Pi 计划宣布时，很多人都对此表示怀疑，因为制造一种售价只需 35 美元信用卡大小的电脑，这听起来完全像是在做梦。不过，这也正是造成 Raspberry Pi 开始发售时人们疯狂追捧的原因。

Raspberry Pi 开始发售后，所有的库存很快就销售一空，很多人排队等待发货。除了低廉的价格以外，到底是什么原因让 Raspberry Pi 能吸引这么多人的目光呢？在我们开始探讨 Raspberry Pi 各种让人激动的特性前，先来了解一下这个产品的目标用户。

英国剑桥大学的 Eben Upton 和他的同事们发现，现在计算机专业的学生，不像 20 世纪 90 年代初的学生们那样具备扎实的基本技能。造成这个现象的原因是家用电脑和电脑游戏的出现，取代了 Amigas、BBC Micros、Spectrum ZX 和 Commodore 64 这些早期的电脑，人们更多的是用电脑来娱乐，而不是开发程序。另外，电脑在家庭中发挥着越来越重要的作用，这使得年轻人们也不能在家用电脑上进行各种实验和探索，因为这样会有把电脑搞坏的风险。如今，手机和平板电脑的处理器变得越来越便宜，而性能也越来越好，这就为 Raspberry



Pi 这种足够便宜且功能完整的电脑主板的诞生指明了方向。正如 Linux 之父 Linus Torvalds 在接受 BBC News 采访时所说的那样：Raspberry Pi 降低了人们试错的成本。

你能拿它来做什么

Raspberry Pi 一个很大的优点就是，它没有一个固定的使用模式。你可以用它来播放视频或上网浏览网页，也可以“折腾”它，学习如何用这块主板做一些东西，Raspberry Pi 是一个具有弹性的平台，它既可以作为玩具来玩，也可以作为工具使用或者作为实验平台。下面是一些 Raspberry Pi 的常见玩法。

普通电脑

你应该意识到，Raspberry Pi 就是一台电脑，所以你也可以把它“当成”电脑来使用，当你按第 1 章中所描述的内容让它正常运行起来后，就可以让它直接启动进入图形化界面，并用它内置的浏览器来上网——这就是我们通常使用电脑的方式。除了上网浏览网页，你也可以在上面安装各种自由软件，如 LibreOffice (<http://www.libreoffice.org/>) 工作套件，在无法接入网络时，可以用它来处理文档或表格。

学习编程

Raspberry Pi 的设计初衷是作为一个教学工具，鼓励青少年开展各种计算机实验，所以它预装了各种编程语言的解释器和编译器。对于初学者来说，可以使用 MIT 开发的 Scratch 图形化编程语言——我们会在第 5 章中讲述。如果你想直接开始编写代码，可以考虑学习使用 Python 语言——我们会在第 3 章中讲述它的一些基础知识。除了 Scratch 和 Python 以外，你还可以用 C、



Ruby、Java 和 Perl 等各种语言为 Raspberry Pi 开发程序。

电子项目平台

与普通电脑相比，Raspberry Pi 除了更小、更便宜外，还有一个重要的特点，那就是可以用它来做电子项目实验平台。从第 7 章开始，我们会介绍如何使用 Raspberry Pi 来控制 LED 和其他电器设备，以及读取按钮和开关的状态。

创客与 Raspberry Pi

作为创客，我们有很多技术开发平台可以选择。最近一段时间，类似于 Arduino 的单片机开发板非常流行，因为它们使用起来很方便。但是，像 Raspberry Pi 这样的片上系统与这些传统的单片机开发板还是不太一样。事实上，Raspberry Pi 跟 Arduino 相比，它更像是一台电脑，而不是一块开发板。

这并不是说 Raspberry Pi 就比传统的单片机要好。例如，你只是想做一个电子温度计，那使用 Arduino UNO 或类似的单片机会更为简单。但是，如果想通过 Web 改变这个温度计的设置或从这个温度计下载温度记录数据文件的话，你可以考虑用 Raspberry Pi。

你所要开发的项目的需求决定了你应该如何在这两种方案中进行选择。但实际上，你也不一定非要做出选择，在第 6 章中，我们就介绍了如何将 Raspberry Pi 与 Arduino 结合起来并在它们之间进行通信。

通过阅读本书，你可以更好地理解 Raspberry Pi 的优势，使它成为你工具箱中的又一件有用的工具。



别急……还有更多精彩的应用！

你可以用 Raspberry Pi 做很多事情，我们无法在一本书中列出它的所有用法。下面是其他的一些常见用法。

媒体中心

由于 Raspberry Pi 提供了 HDMI 和复合视频输出端子，所以可以很方便地与电视相连。并且，它还具备足够的处理能力来播放全屏的高清视频。为了能发挥 Raspberry Pi 的这些能力，XBMC (<http://xbmc.org/>) 这个免费开源项目的开发者们已经把 XBMC 移植到了 Raspberry Pi 上面。XBMC 可以播放各种多媒体文件，并且它在界面上采用较大的按钮和菜单，很适合坐在沙发上遥控它。XBMC 把 Raspberry Pi 打造成一个完全可定制的家庭娱乐中心。

裸机开发

大部分人写的程序都需要在操作系统环境（如 Windows 或 Mac OS）下运行，对于 Raspberry Pi 来说，程序则需要在 Linux 操作系统下运行。但是，你有没有想过要写一些程序直接在 Raspberry Pi 处理器上运行而不需要通过操作系统呢？如果你愿意的话，甚至可以在上面从头开发一个全新的操作系统。剑桥大学的计算机实验室发布了一份免费在线课程 (<http://www.cl.cam.ac.uk/freshers/raspberrypi/tutorials/os/>)，指导你如何使用汇编语言为 Raspberry Pi 开发操作系统。

Linux 与 Raspberry Pi

我们平时使用的电脑上通常都运行着一个操作系统，如



Windows、OS X 或 Linux。当你打开电脑时，操作系统会自动启动，它为应用程序提供了访问电脑硬件的能力。例如，当你编写一个访问 Internet 的程序时，就可以使用操作系统提供的功能来完成必要的操作，既不需要理解各种不同的有线或无线网卡的硬件原理，也不需要为不同的硬件编写不同的代码。

与其他电脑一样，Raspberry Pi 也需要一个操作系统，官方推荐的操作系统是 Raspbian 这个 Linux 发行版。自由与开源的 Linux 系统与 Raspberry Pi 是一个很好的组合：一方面，它使整个平台的价格保持在一个最低的水平；另一方面，也使这个平台更适合“折腾”。当然，Raspbian 也不是你的唯一选择，除了它以外还有很多不同的 Linux 发行版可以使用。甚至还有一些非 Linux 系统，也可以在 Raspberry Pi 上运行。在本书中，我们只使用 Raspberry Pi 的下载（Downloads）页面（<http://www.raspberrypi.org/downloads>）上提供的 Raspbian 系统。

如果你不熟悉 Linux 操作系统，也不用着急，我们将在第 2 章中讲述一些使用 Linux 的基础知识。

其他人都用它来做些什么

当你刚接触到一项新的科技时，常常很难想象可以用它来做些什么。如果你觉得疑惑，可以参考其他人的创意来获取灵感。作为 MAKE 的编辑，我们看到了 Raspberry Pi 的很多奇妙的应用，在这里与大家一起分享其中的一部分。

街机游戏咖啡桌

（<http://www.instructables.com/id/Coffee-Table-Pi/>）

Instructables 网站的用户 grahamgelding 上传了一个详细的教程，讲述了如何制造一个可以当成街机来使用的咖啡桌——通过在



Raspberry Pi 上运行的街机模拟器来实现。为了能在 Raspberry Pi 上运行游戏，他使用了 MAME (Multiple Arcade Machine Emulator) 这个开源、自由的模拟器，这个模拟器可以用来在现代电脑上运行一些经典的街机游戏。这个桌子里内置了一台 24 英寸的液晶显示器，显示器通过 HDMI 接口与 Raspberry Pi 相连，街机的按钮和操纵杆则作为输入设备通过 GPIO 接口与 Raspberry Pi 相连。

RasPod

(<https://github.com/lionaneesh/RasPod>)

印度少年 Aneesh Dogra 是 2012 年 Raspberry Pi 基金会夏季编程大赛的亚军，他用 Raspberry Pi 设计了一个名为 RasPod 的网络 MP3 播放器，这个播放器可以通过 Web 页面进行操作。这个程序用 Python 和一个名为 Tornado 的 Web 框架来实现，通过使用 RasPod，你可以远程登录到 Raspberry Pi 上并控制音乐播放、改变音量、选歌和创建播放列表。音乐通过 Raspberry Pi 的音频输出口输出，所以可以连接到电脑音箱或音响设备上。

Raspberry Pi 超级计算机

(http://www.southampton.ac.uk/mediacentre/features/raspberry_pi_supercomputer.shtml)

很多超级计算机都是通过把很多普通电脑连接起来组成集群，并把计算任务分发到多个处理器上计算来实现的。英国南安普顿大学的一群计算机工程师把 64 台 Raspberry Pi 连接在一起，组成了一台廉价的超级计算机。虽然这个集群的计算能力与现代顶级的超级计算机相比还有很大的差距，但它完整地展示了超级计算机集群背后的技术和原理。更有意思的是，放置这些 Raspberry Pi 的“机架”是由这个团队负责人的 6 岁的儿子用乐高积木搭成的。



如果你用 Raspberry Pi 实现了什么有意思项目，我们对此会非常感兴趣。你可以通过 Makezine.com 上的表单 (<http://blog.makezine.com/contribute/>) 向 *MAKE* 教育团队提交你的创意。

本书约定

在本书中，我们使用下列字体格式来表达特定的含义。

斜体英文 (*Italic*)

斜体英文用于表示作品名、网址、电子邮件地址、文件名和文件扩展名。

等宽字体 (Constant Width)

用于表示源代码，同时也出现在正文中，用于标识程序中的元素，如变量名、函数名、数据库、数据类型、环境变量、表达式和关键字。

粗体的等宽字体 (Constant with Bold)

用于在命令中或正文中表示用户手工输入的内容。



这个图标表示小技巧、建议或注释。



这个图标表示警告或注意事项。

使用示例代码

撰写本书的目的就是帮助你完成工作。所以，可以在你的程序



或文档中自由使用本书中的代码。如果不是大量引用本书中的代码，都无须联系我们申请授权。例如，你自己写了一个程序，使用了本书中的几段代码，这种情况是无须申请授权的。销售或分发 O'Reilly 图书配套示例文件需要申请授权。回答问题时引用本书的内容并摘录代码片段无须申请授权。把本书中的示例代码大量用于你的产品文档时需要申请授权。

我们希望但不强制要求你注明出处。当注明出处时，通常应当包含书名、作者、出版社和 ISBN。比如：“*Getting Started With Raspberry Pi* by Matt Richardson and Shawn Wallace (O'Reilly). Copyright 2013, 978-1-4493-4421-4.”

如果你不确定对示例代码的使用是否在我们允许的范围内，请与 permissions@oreilly.com 联系。

Safari[®] 在线图书



Safari 在线图书是一个数字图书馆，它允许你在超过 7500 种技术创新图书和视频中快速检索你所需要的信息。

通过订购我们的服务，你可以从我们的在线图书馆中阅读各种图书、观看各种视频，用手机或其他移动设备来阅读图书，预览即将出版的新书、获取开发指导手册和向作者反馈意见，获取示例代码，管理收藏夹，下载章节内容，收藏关键内容，记录笔记，打印图书，享用我们各种有助于提高效率的服务。

O'Reilly Media 已经把本书英文版上传到 Safari 在线图书服务中，可以免费在 <http://my.safaribooksonline.com> 上注册账号，访问 O'Reilly 和其他出版社的数字图书及相关资源。



如何联系我们

请把关于本书的建议和问题写信到以下地址：

美国：

O'Reilly Media, Inc
1005 Gravenstein Highway North
Sebastopol, CA 95472

中国：

北京市西城区西直门南大街 2 号成铭大厦 C 座 807 室 (100035)
奥莱技术咨询 (北京) 有限公司

MAKE 致力于团结和激励那些在自家院子、地下室或车库中创造出奇妙事物的人们。MAKE 鼓励你使用各种技术进行创新。MAKE 的社区和文化正在不断地成长，我们有信心让自己变得更强、让我们的环境和教育变得更好、让我们的世界变得更美好。这不仅仅是一个社区，更是一个全球化的运动——属于创客的运动。

了解有关 MAKE 的更多信息，请访问：

MAKE 杂志：<http://makezine.com/magazine/>

Maker Faire：<http://makerfaire.com>

Makezine.com：<http://makezine.com>

Maker Shed：<http://makershed.com>

我们为本书提供了一个支持页面，上面提供了勘误表、示例代码和其他相关的信息。你可以访问：

<http://shop.oreilly.com/product/0636920023371.do>

有任何建议或需要咨询技术问题，请发送电子邮件到下面的地址：



bookquestions@oreilly.com

其他书籍、课程、会议或新闻，请访问我们的网站：

<http://www.oreilly.com>

我们的 Facebook 页面：

<http://facebook.com/oreilly>

可以在 Twitter 上关注我们：

<http://twitter.com/oreillymedia>

我们在 YouTube 上的视频：

<http://www.youtube.com/oreillymedia>

致 谢

感谢以下在本书写作过程中提供了相关知识、支持、建议和反馈意见的朋友：

Brian Jepson

Marc de Vinck

Eben Upton

Tom Igoe

Clay Shirky

John Schimmel

Phillip Torrone

Limor Fried

Kevin Townsend

Ali Sajjadi

Andrew Rossi

O'Reilly Media, Inc. 介绍

O'Reilly Media 通过图书、杂志、在线服务、调查研究和会议等方式传播创新知识。自 1978 年开始，O'Reilly 一直都是前沿发展的见证者和推动者。超级极客们正在开创着未来，而我们关注真正重要的技术趋势——通过放大那些“细微的信号”来刺激社会对新科技的应用。作为技术社区中活跃的参与者，O'Reilly 的发展充满了对创新的倡导、创造和发扬光大。

O'Reilly 为软件开发人员带来革命性的“动物书”；创建第一个商业网站（GNN）；组织了影响深远的开放源代码峰会，以至于开源软件运动以此命名；创立了 *Make* 杂志，从而成为 DIY 革命的主要先锋；公司一如既往地通过多种形式缔结信息与人的纽带。O'Reilly 的会议和峰会集聚了众多超级极客和高瞻远瞩的商业领袖，共同描绘出开创新产业的革命性思想。作为技术人士获取信息的选择，O'Reilly 现在还将先锋专家的知识传递给普通的计算机用户。无论是通过书籍出版，在线服务或者面授课程，每一项 O'Reilly 的产品都反映了公司不可动摇的理念——信息是激发创新的力量。

业界评论

“O'Reilly Radar 博客有口皆碑。”

——Wired

“O'Reilly 凭借一系列（真希望当初我也想到了）非凡想法建立了数百万美元的业务。”

——Business 2.0

“O'Reilly Conference 是聚集关键思想领袖的绝对典范。”

——CRN

“一本 O'Reilly 的书就代表一个有用、有前途、需要学习的主题。”

——Irish Times

“Tim 是位特立独行的商人，他不光放眼于最长远、最广阔的视野并且切实地按照 Yogi Berra 的建议去做了：‘如果你在路上遇到岔路口，走小路（岔路）。’回顾过去 Tim 似乎每一次都选择了小路，而且有几次都是一闪即逝的机会，尽管大路也不错。”

——Linux Journal

目 录

第1章 安装与启动

主 板	3
必备的外设	8
外 壳	11
选择操作系统	12
烧录 SD 卡	14
启动系统	16
配置你的 Pi	18
关闭系统	21
故障排除	21
进一步学习	23

第2章 初识 Raspberry Pi 上的 Linux

使用命令行	29
文件与文件系统	30
更多 Linux 命令	35
进 程	38
sudo 与权限	39
网 络	41
/etc 目录	43
设置日期时间	43
安装新软件	44



进一步学习	45
-------------	----

第3章 Pi 上的 Python

初识 Python	49
进一步学习 Python	52
对象与模块	55
更多模块	59
错误调试	60
进一步学习	61

第4章 用 Python 实现动画与多媒体

初识 Pygame	64
Pygame 的 Surface	66
在 Surface 上绘图	68
处理事件与输入	69
Sprite	73
播放声音	75
播放视频	78
进一步学习	79

第5章 Pi 上的 Scratch

初识 Scratch	82
舞 台	87
有关角色的两点知识	88
更复杂的例子：星际入侵者游戏	90
Scratch 与现实世界	97
分享你的程序	98



进一步学习	99
-------------	----

第6章 Arduino 与 Pi

在 Raspbian 上安装 Arduino	103
定位串口	105
串口通信	106
进一步学习	111

第7章 基本输入输出

使用输入输出接口	116
数字信号输出：点亮 LED	118
数字信号输入：读取按钮状态	122
项目：定时台灯	126
脚本命令	126
连接台灯	128
用 cron 设置定时任务	129
更多有关 cron 的知识	131
进一步学习	132

第8章 用 Python 进行输入输出编程

在 Python 中安装并测试 GPIO	134
让 LED 闪烁	137
读取按钮状态	140
项目：简易发音板	142
进一步学习	147



第9章 使用摄像头

测试摄像头	151
安装并测试 SimpleCV	152
显示图片	154
修改图片	156
操作摄像头	159
人脸检测	160
项目：Raspberry Pi 照相馆	162
进一步学习	166

第10章 Python 与 Internet

从 Web 服务器下载数据	168
获取天气预报	170
用 Pi 提供服务（做 Web 服务器）	176
Flask 入门	176
把 Web 与现实世界相连	181
项目：Web 台灯	183
进一步学习	188

附录 A 烧录 SD 卡镜像

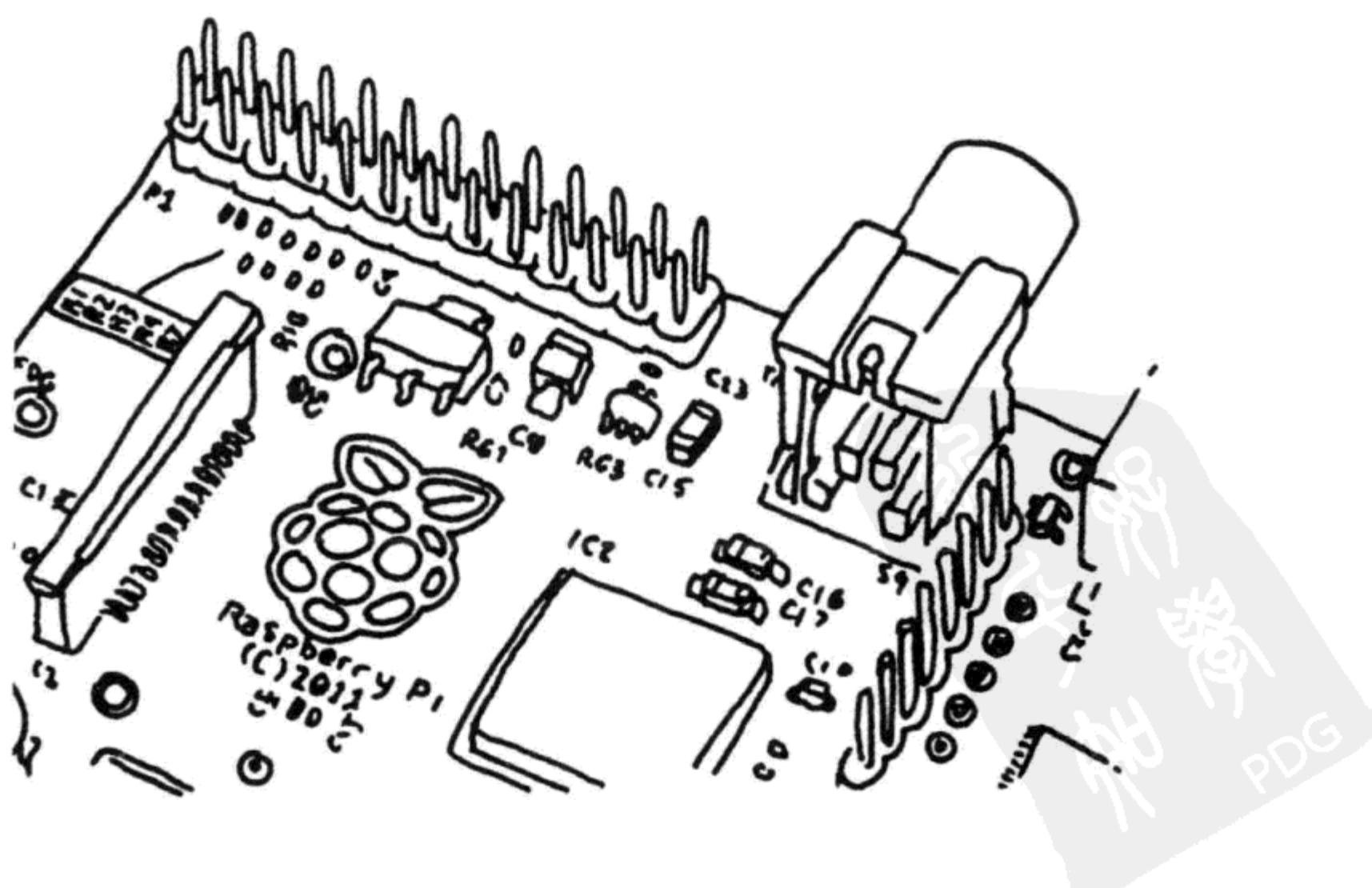
附录 B 星际入侵者游戏完整版

附录 C 模拟信号输入

第 1 章

安装与启动

Getting Up and Running





当谈起 Raspberry Pi 时，总会一次又一次地提到小巧、便宜、适合“折腾”和面向教育这几个特性。尽管它可以方便地连接到电视机并在屏幕上显示出一些内容，但它仍然不适合作为一个即插即用的设备。Raspberry Pi 并不是一个消费类电子产品，根据你所设想的不同用途，还需要添加一系列额外的硬件、软件才能让它正常运作起来。

首先，你需要购买一台 Raspberry Pi。如果你现在还没有的话，Raspberry Pi 基金会与多家制造商达成了协议：以 25~35 美元的价格向你发售一台 Raspberry Pi。

Premier Farnell/Element14

(<http://www.element14.com/community/groups/raspberry-pi/>)

一家英国的电子器件批发商，它在全球范围内有分销机构，如美国的 Newark 和 MCM。

RS Components

(<http://www.rs-components.com/raspberrypi>)

另一家总部位于英国的全球电子器件批发商，它也是美国 Allied Electronics 的母公司。

Raspberry Pi 的低廉价格得益于它的销售模式，大众可以直接去大批发商那里以批发价少量购买设备，甚至只购买一台设备，这是一种并不常见的销售模式。很多分销商在 Raspberry Pi 宣布



售价时感到很为难，因为他们销售这个设备几乎得不到利润，所以你会发现市场上有些零售商家会以略高于 35 美元的价格（如 40 美元）销售 Raspberry Pi。当然，大众可以直接从批发商那里以原价购买到 Raspberry Pi，只是，与从零售商那里购买相比，批发商的发货速度可能会稍微慢一些。MAKE 的 Make Shed (http://www.makershed.com/category_s/227.htm) 以及 Adafruit (<http://www.adafruit.com/category/105>) 都以微薄的利润销售 Raspberry Pi 和相关配件。

说了很多有关微观经济学的题外话，下面我们可以一起来看看 Raspberry Pi 到底是怎样的一台设备。

主 板

我们一起来看看从包装盒里拿出来的 Raspberry Pi 是什么样子的。

有些人可能理所当然地认为 Raspberry Pi 是像 Arduino 一样的单片机开发板，也有人认为 Raspberry Pi 是笔记本电脑的替代品，事实上，它更像是一个移动设备的内脏，并且加上了一些可以用于二次开发的接口。图 1.1 展示了这块板子的全貌。

A. 处理器。Raspberry Pi 使用与 iPhone 3G 或 Kindle 2 一样的处理器，所以你可以认为 Raspberry Pi 的处理能力与这两个设备类似。这是一块 32 位的片上系统（SoC，System on a Chip），主频 700MHz，ARM11 架构。ARM 芯片有很多种架构，并且每一种架构的特性都有所区别，当然价格也不一样。B 型的 Raspberry Pi 装备了 512M 内存，A 型的则装备了 256M 内存（但早期第一批 B 型的 Raspberry Pi 也只装备了 256M 内存）。

B. SD 卡插槽。Pi 上没有硬盘，所有的数据都保存在 SD 卡上。



建议你尽早为你的 Pi 配备一个保护外壳，因为插上 SD 卡后，不小心碰到 SD 卡可能会损坏 SD 卡插槽。

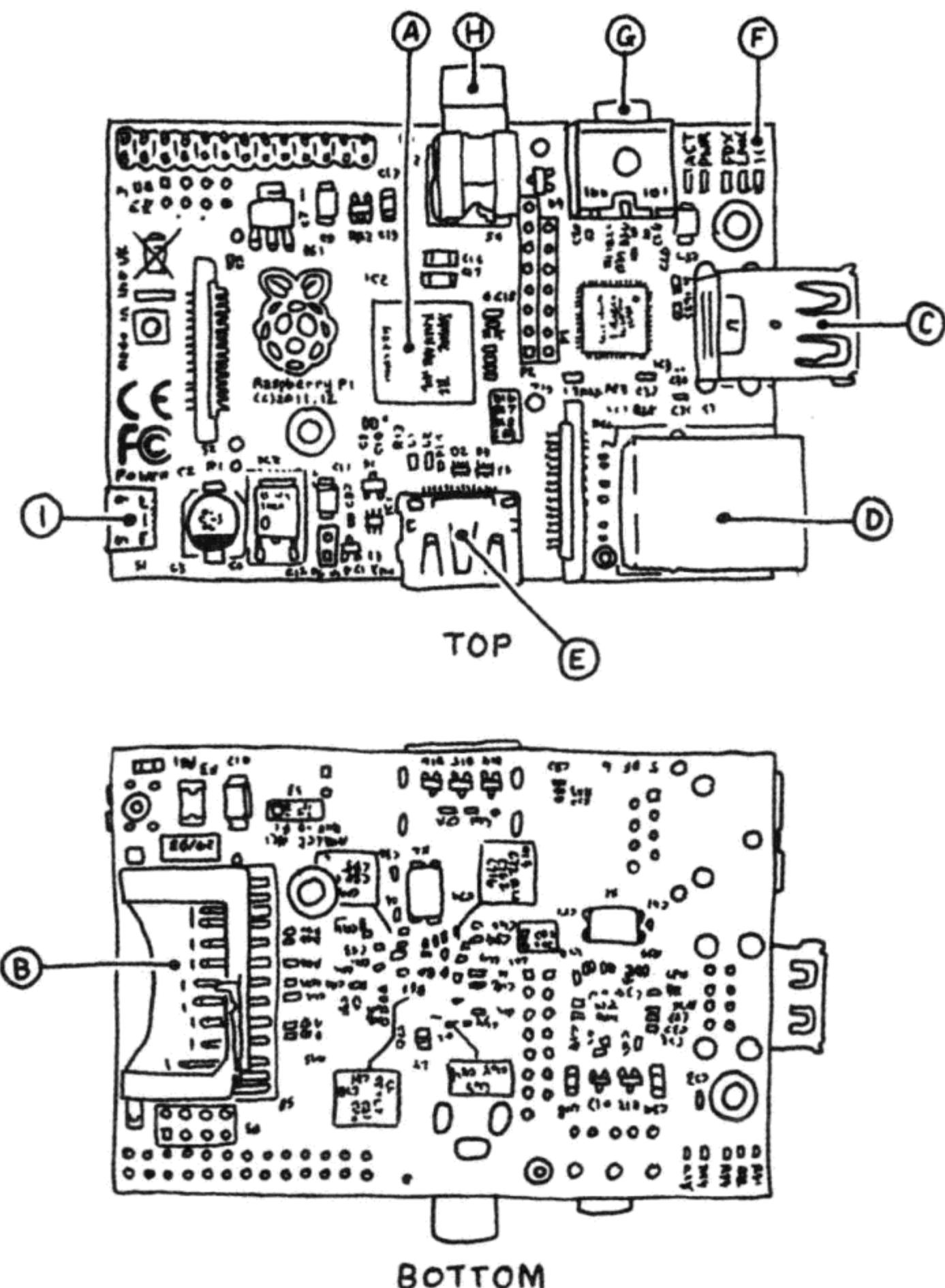


图1.1 Raspberry Pi的硬件接口图



C. USB 接口。B 型的 Raspberry Pi 提供两个 USB 接口，而 A 型的只有一个。早期版本的 Raspberry Pi 主板只能给 USB 接口提供非常有限的电流，但一些 USB 设备却可能会消耗 500mA 以上的电流。早期的 Pi 的 USB 接口只能输出 100mA 左右的电流，但新版本都已经可以提供更大的电流，满足完整的 USB 2.0 标准的要求。你可以检查你的 Pi 的 USB 接口旁是否有两个限制电流用的保险丝（参考图 1.2），用于确认自己用的是哪一个型号批次的 Pi。在任何情况下，都不要用 Pi 的 USB 接口给你的手机充电。如果你的 USB 设备需要较大的电流，可以考虑使用一个有源的 USB Hub。



图1.2 部分老版本的主板为USB口配备了保险丝（左图），部分主板把保险丝换成了导线（中图），最新的B型主板，去除了保险丝或导线，在这个位置布置了一个固定螺丝孔

D. 以太网接口。B 型的 Raspberry Pi 提供一个标准的 RJ45 以太网口，A 型没有，但它可以通过额外安装 USB 接口的以太网适配器连接网络（事实上 B 型板载的以太网口也是连接在 USB 总线上的）。也可以通过 USB 接口的无线网卡接入网络。

E. HDMI 接口。HDMI 接口提供数字视频与音频输出，它可以支持 14 种不同的分辨率。通过使用外置的转换器，HDMI 信号可以转换为 DVI（很多显示器用这种接口）信号、复合信号（通过黄色 RCA 端子连接的模拟视频信号）或 SCART（欧洲的一种连接音视频设备的标准）信号。



F. 状态指示灯。Pi 主板上有 5 个发光二极管 (LED)，可以用于显示系统的工作状态 (表 1.1)。

表1.1 5个状态指示灯

ACT	绿 色	当系统读写 SD 卡时点亮 (在早期的主板上被标示为 OK)
PWR	红 色	连接 3.3V 电源
FDX	绿 色	当网卡工作在全双工模式时点亮
LNK	绿 色	网络活动指示灯
100	黄 色	当网络工作在 100Mbps 时点亮 (在部分早期的主板上被错标成 10M)

G. 模拟音频输出。Raspberry Pi 提供一个标准的 3.5mm 模拟音频输出插孔，可以用于连接高阻抗的音频设备 (如有源音箱)。如果直接连接耳机或无源音箱，音质可能不会很好。截至本书写作时为止，这个音频插孔输出的声音音质比 HDMI 接口连接电视输出的音质要差很多，造成这个问题的原因与驱动程序有一定关系——这个驱动程序也还在不断改进之中。

H. 复合视频输出。复合视频输出孔是一个标准的 RCA 插孔，可以输出 PAL 或 NTSC 制式的视频信号。与 HDMI 相比，这种视频格式的分辨率会低很多。如果你的显示器或电视机可以支持 HDMI，就尽量使用 HDMI，而不要使用复合视频输出。

I. 电源接口。首先，Pi 上没有电源开关，在电源接口上插上 Micro USB 电源线就接通了 Pi 的电源 (这个 Micro USB 口只用于供电，不能作为普通的 USB 口传输数据)。我们选择用 Micro USB 接口作为供电接口的原因是，这种接口比较便宜，而且现在很容易找到这种 USB 接口的电源适配器。

图 1.3 展示了 Raspberry Pi 上的所有输入输出 (I/O) 接口，详见下面的描述。

A. 通用输入输出接口 (GPIO, General Purpose Input and

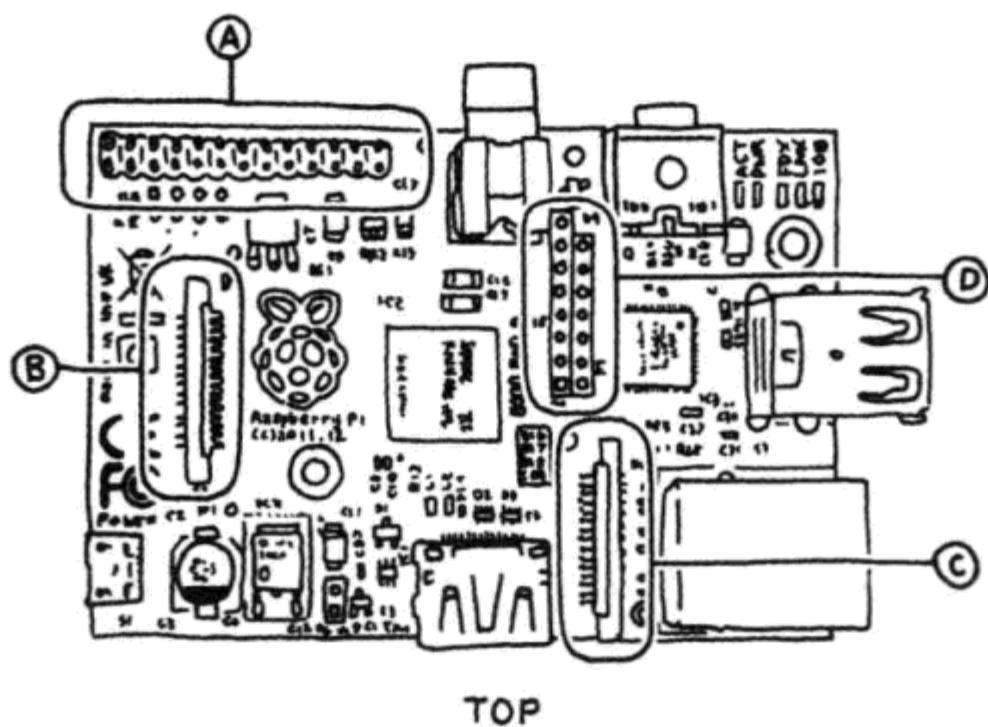


图 1.3 Raspberry Pi 上的接口与插座

Output) 等接口。我们会在第 7 章与第 8 章中介绍如何使用这些接口来获取按钮的状态或用它们来控制 LED、继电器或电动机。

B. DSI (Display Serial Interface) 接口。可以通过这个接口用 15 针扁平电缆连接液晶屏或 OLED 显示屏。

C. CSI (Camera Serial Interface) 接口。这个接口可以用于把摄像头直接接到主板上。

D. P2 与 P3 接口。这两个接口提供了 Broadcom 芯片 (P2) 与 LAN9512 网卡芯片 (P3) 的 JTAG 调试接口。不过, 由于 Broadcom 的芯片组标准并不公开, 这些接口对普通用户来说并没有太大的用处。



2012 年秋季, Raspberry Pi 基金会发布了新版本的 Raspberry Pi 主板, 新版本的主板上增加了两个 2.5mm 的固定螺丝孔和一个两针的重启接口。同时还提供了一个 2×4 的接口 (没有焊接插针), 预留给第三方的时钟或音频扩展卡 (可以固定在主板的下方)。



必备的外设

通过前面的介绍，你已经了解了 Raspberry Pi 主板上的所有部件，接下来需要了解一些让 Pi 能正常工作所必需的外部设备，如图 1.4 所示。市面上可以购买到各种事先搭配好的入门套件可供直接使用，如果你自己准备这些外设，则有一些细节需要注意。在 Raspberry Pi 的 Wiki 页面上有一个完整的外设支持列表 (http://elinux.org/RPi_VerifiedPeripherals) 可以参考。

A. 电源适配器。这是最重要的一个外设，你需要选购一个至少可以提供 700mA 电流的 5V Micro USB 接口的电源（如果是 A 型 Pi，只需提供 500mA 电流）。很多手机充电器即使接口合适也可能并不能满足这个电流的需求，因为一般的手机充电器可能只能提供 400mA 甚至更小的电流，不过实际情况以充电器铭牌上的标注为准。如果电源不能给 Pi 提供足够的电流，也许 Pi 看上去也能工作，但有可能会工作得不太正常，出现各种不可预知的问题。



按照当前版本的 Raspberry Pi 的电路设计，你可以使用有源 USB Hub 给它进行反向供电。但是，如果用 USB Hub 进行反向供电，则所有的电源保护电路都不会起作用，所以最好不要这样做。尤其是，如果拿 Raspberry Pi 进行一些电路实验，你很有可能会不小心造成短路，短路所形成的巨大电流很可能会烧坏你的 Pi 和电源。

B. SD 卡。你需要一张最小 4GB 容量、速度为 Class 4 的 SD 卡，Class 4 的 SD 卡每秒至少可以读写 4M 数据。Class 6 的 SD 卡读写速度更快，但相对不稳定，早期的 Raspberry Pi 与 Class 6 的 SD 卡

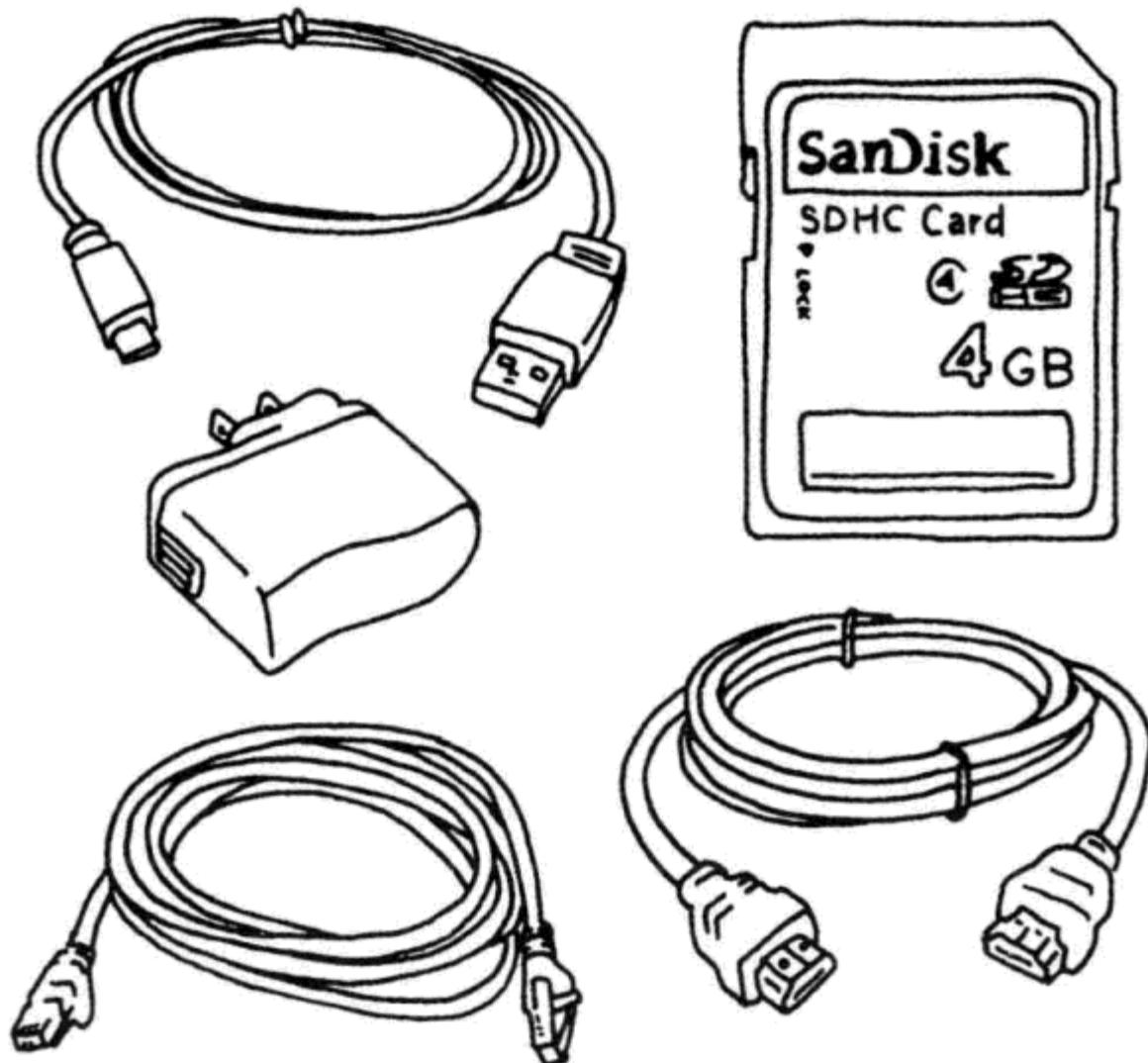


图1.4 基本外设：Micro USB电源适配器、各种电缆和SD卡。你需要一张最小4GB容量、速度为Class 4的SD卡（通过一个SD卡适配器使用Micro SD卡也可以）。不同的SD卡质量差别很大，最好选择质量有保证的型号（参考http://elinux.org/RPi_VerifiedPeripherals#SD_cards）

可能会不兼容。如果你通过一个SD卡适配器使用Micro SD卡，也是可以的。

C. HDMI 电缆。你需要 HDMI 电缆来连接显示器，如果显示器是 DVI 接口，则需要 HDMI 转 DVI 的连接线。你也可以无显示器（Headless）使用 Pi，这在后面的章节中会介绍。不同的 HDMI 电缆的价格差别很大，如果你只是需要一根 3~6ft(1ft=0.3048m) 的电缆来连接显示器的话，你只需购买一根 3 美元以下的电缆就可以了。但如果你需要用一根很长的电缆来连接显示器，则需要考虑购买质量更好的电缆，不要买低价、低质的产品。



D. 网线。现在无线网络越来越普及，也许你的家中不像 5 年前那样有很多的有线网络接口，要找到一个有线网口可能比较困难。这种情况下，你可以参考“无显示器运行”中的内容来找到除了把网线插到墙上有线网口或集线器上以外的解决方案。

如果你会大量使用 Raspberry Pi，那么还有一些外设也值得关注，我们会在第 5 章中提到它们。你也许会对这个已知可用的外设列表 (http://elinux.org/RPi_VerifiedPeripherals) 上的设备感兴趣。

有源 USB Hub

推荐使用 USB 2.0 的 Hub。

散热片

散热片是一小块金属片，上面通常有很多散热鳍来增加散热面积，改善散热效果。散热片可以贴到需要散热的芯片上。Pi 的芯片是为移动设备设计的，所以大多数情况下并不需要添加散热片。但是，在后面的章节中我们会看到，如果希望你的 Pi 以更快的速度运行或者长时间地高负荷运行，芯片还是会有些发烫。另外，网卡芯片有时也会发热。

实时时钟 (RTC)

你可以为 Raspberry Pi 装备一块 RTC 芯片（如 DS1307），用于在断电时继续保持时钟运行。

摄像头

Raspberry Pi 官方的 500 万像素的摄像头将在 2013 年的年初发布^①。在此之前，你可以用一个 USB 接口的网络摄像头来代替它，

^① 该摄像头已于 2013 年 5 月 14 日正式发布。——译者注



在第 9 章有一个完整的例子。

液晶屏

很多简易的液晶屏可以通过少量的接线连接到 GPIO 接口上使用。适用于 Pi 的 DSI 接口的液晶屏在 2013 年上市。

USB 无线网卡

很多 USB 接口的无线网卡都可以在 Pi 上正常使用，选购时请尽量选择对电流要求较低的型号。

扩展坞

有些人改装了原本用于手机的扩展坞（如 Atrix），可以用作 Raspberry Pi 的底座和显示器。

外 壳

开始使用 Raspberry Pi 后，你很快会发现需要给它配备一个外壳。插满了电线的 Raspberry Pi 很难被安放稳妥，SD 卡槽等组件即使是正常使用，也可能会被不小心碰坏。

Pi 的 PCB 板是 6 层板，不像很多其他开发板那样只有上下两层是信号层，它的板子中间还夹了 4 层信号层。所以，如果你把它的主板过度弯曲变形，很可能会破坏板子中间的导线，造成一些难以排查的故障。解决的方案就是，给它加上一个外壳。

市面上有很多现成的外壳可以购买，网上也能找到很多适合自己用激光切割机或 3D 打印机制作的外壳设计方案。请尽量避免选购亚克力板立体拼装的盒子，因为亚克力板的材质很脆，不过可以考虑用多层亚克力板水平层叠的方案，如图 1.5 所示的五彩 Pibow 外壳 (<http://pibow.com/>) 就是个不错的选择。

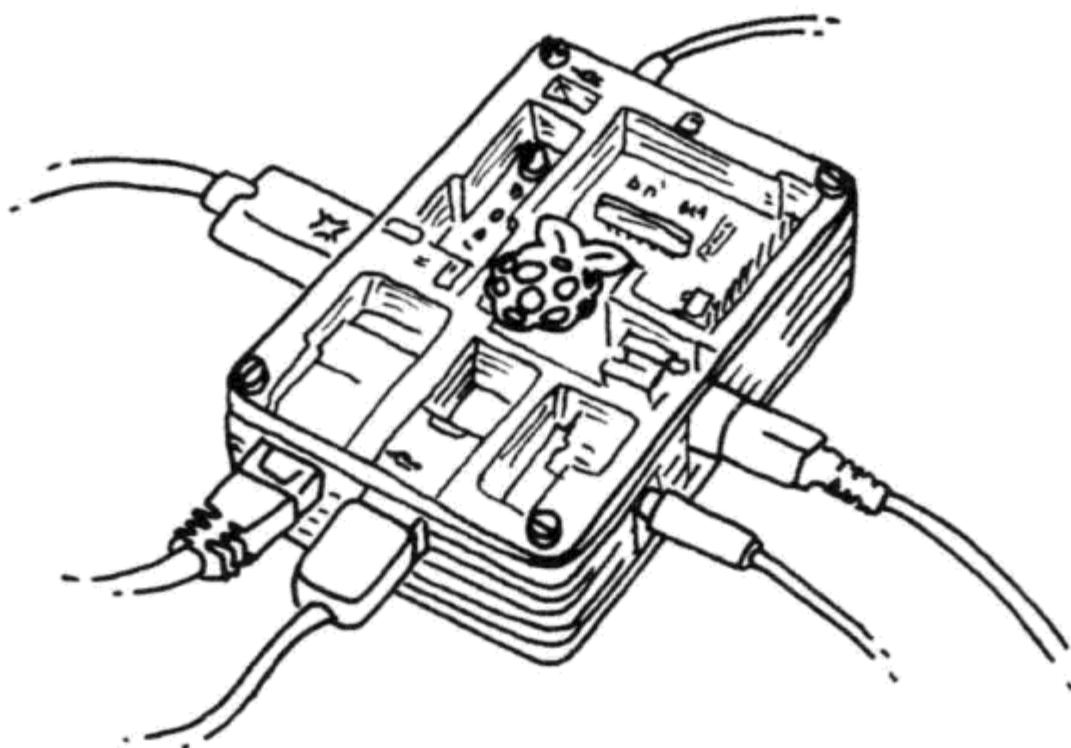


图1.5 五彩的Pibow外壳

虽然是显而易见的道理，但还是要提醒大家，因为错误总是在不经意间发生——千万不要把 Raspberry Pi 放在一个导体的表面上。把主板翻过来，你可以在主板的背面看到很多元器件的焊接引脚，这些引脚很容易被短路，这也是为什么需要为你的 Pi 配备一个外壳的重要原因。

选择操作系统

Raspberry Pi 使用 Linux 作为其操作系统，从严格的技术层面来说，Linux 只是一个内核，而一个完整的操作系统还需要包含驱动程序、服务和应用程序等很多组件。Linux 诞生至今已经演化出很多不同的发行版，如在台式电脑上常用的发行版有 Ubuntu、Debian、Fedora 和 Arch 等，每个发行版都在不同的方面为特定的应用程序进行了优化，并有属于它自己的用户社区。

由于 Pi 是基于移动设备芯片设计的，它与台式电脑的软件需求不太一样。Broadcom 处理器有一些私有的功能需要一些特殊的



二进制设备驱动程序来驱动，这些代码不能被包含在任何标准的 Linux 发行版中。另外一方面，台式电脑通常配有几吉（G）的内存和几百吉的硬盘，但 Pi 的硬件资源非常有限，所以需要开发一些 Pi 专用的 Linux 发行版，常见的如下。

Raspbian

(<http://raspbian.org>)

Raspberry Pi 基金会“官方推荐”的发行版，基于 Debian 实现（图 1.6）。不过，raspbian.org 网站（<http://raspbian.org>）是一个社区独立运营的网站，与 Raspberry Pi 基金会无关。如果要下载官方的发行版，请访问 raspberrypi.org 的下载页面。

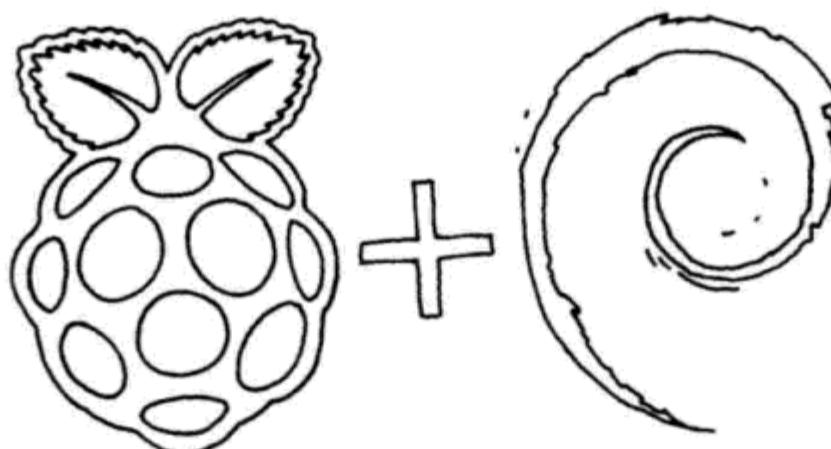


图1.6 Raspberry Pi + Debian = Raspbian

Adafruit Raspberry Pi Educational Linux (Occidentalis)

(<http://learn.adafruit.com/adafruit-raspberry-pi-educational-linux-distro>)

这是由 Adafruit 开发的基于 Raspbian 的发行版，里面包含了一些适用于电子开发教学的工具和驱动程序。

Arch Linux

(<http://www.archlinux.org/>)



Arch Linux 一直很重视对 ARM 架构的电脑的支持，所以它们从很早的时候就开始支持 Pi。

XBian

(<http://xbian.org/>)

这是一个基于 Raspbian 的发行版，主要用于把 Raspberry Pi 当成媒体中心来使用。与之类似的还有 OpenELEC (<http://openelec.tv/>) 和 Raspbmc (<http://www.raspbmc.com>) 。

QtOnPi

(<http://qt-project.org/wiki/qt-raspberrypi>)

一个基于 Qt 5 框架的发行版。

在本书中，我们只关注官方版本的 Raspbian 发行版。

烧录 SD 卡

不少商家出售事先安装好操作系统的 SD 卡，对于一些初学者来说，这是个不错的选择。虽然这些 SD 上不能提供最新版本的操作系统，但你只要先用它们把 Pi 启动起来，就可以通过 Internet 对系统进行更新。

Raspbian 也提供了一个网络安装工具 (<http://www.raspbian.org/RaspbianInstaller>)，如果想要使用它，只需把相关的安装文件写到 SD 卡上 (SD 卡需要格式化为 FAT32 格式，通常 SD 卡默认就是这种格式的)，并用这张 SD 卡启动 Pi 即可。不过，你需要确认你的 Pi 正确接入了网络，这种安装方式才会奏效。

如果不打算通过网络安装，那就需要从 [raspberrypi.org](http://www.raspberrypi.org/downloads) 的下载页面 (<http://www.raspberrypi.org/downloads>) 上下载



Raspbian 的镜像文件。注意，你不能直接把镜像文件复制到 SD 卡上，而是需要把它按位写到 SD 卡上。你需要一个普通的读卡器和一个磁盘镜像工具来完成这项任务。具体的操作方法取决于你的电脑上所安装的操作系统：首先把下载的镜像文件解压缩，得到一个 *.img* 文件，然后按照附录 A 中描述的步骤进行操作。

使用 BitTorrent 进行高速下载

在 Raspbian 下载页面上，你会看到一个提示——“可以通过下载种子文件更快速地完成下载”。通过种子文件下载是一种分布式的文件下载模式，使用这种模式，你可以同时从多个 BitTorrent 客户端那里下载文件的不同部分，而不是只从一个中心服务器下载。如果想用这种方式完成下载，你需要安装一个 BitTorrent 客户端程序。

常见的 BitTorrent 程序如下。

- Vuze (<http://www.vuze.com/>)：整合了种子搜索和下载的功能。
- Miro (<http://www.getmiro.com/>)：开源的音乐和视频播放器，同时也可进行 BitTorrent 下载。
- MLDonkey (http://mldonkey.sourceforge.net/Main_Page)：Windows 和 Linux 下的文件分享工具。
- Transmission (<http://www.transmissionbt.com/>)：轻量级的 Mac 和 Linux Torrent 客户端程序，也常常用在嵌入式系统中。



启动系统

第一次启动 Raspberry Pi 时，可以按下面的步骤操作。

1. 把 SD 卡插入卡槽。
2. 接好 USB 键盘和鼠标。对于 A 型的 Raspberry Pi，你需要先把它们接到一个有源的 USB Hub 上，然后把 USB Hub 接到 Pi 上。
3. 把 HDMI 输出接到你的显示器或电视上，并确认打开了显示器。
4. 接上电源线。原则上，请在接上电源线前确认其他接线都已经接好。

接入网络

有若干种方式可以把你的 Raspberry Pi 接入网络。如果你手头有一个路由器或交换机（或者一个接到路由器上的网络接口），直接用一根标准的以太网线把 Pi 与之相连即可。如果你有一个 **USB** 无线网卡，则可以通过系统桌面上的无线设备图标来设置无线网络连接。不过，不是所有的 **USB** 无线网卡都可以在 Pi 上正常工作，你可以参考已知可用的外设列表 (http://elinux.org/RPi_VerifiedPeripherals) 上的内容来选择一种可以使用的型号。

如果手头有一台笔记本电脑，或者采用无显示器运行模式，你可以把电脑上的 WiFi 连接共享给 Pi 使用（图 1.7）。在 Mac 上实现网络共享非常容易，只需在“共享”设置中启用“Internet 共享”，然后用网线把 Mac 和 Pi 连接起来即可。在 Windows 中，可以启用“允许其他网络用户通过此计算机的 Internet 连接来连接”选项。Pi 连接上后，可以自动获取到 IP 地址并接入 Internet。



如果使用 Windows 电脑，你也许需要通过交叉网线来连接 Pi 和电脑。但如果是苹果电脑，它会自动识别网线的类型。

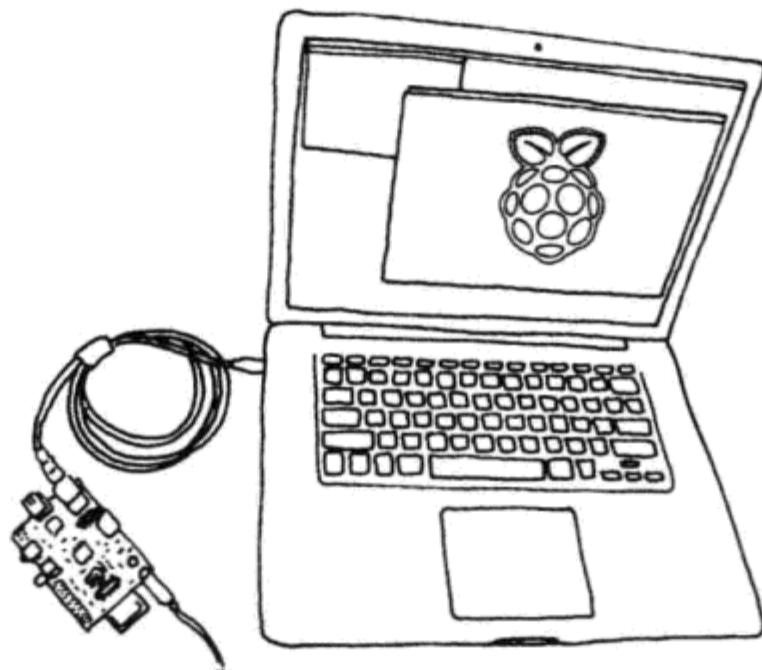


图1.7 Pi共享笔记本WiFi连接的方案，你还可以无显示器运行Pi（参考“无显示器运行”），这样用起来非常方便

如果一切正常，你可以在屏幕上看到很多启动日志。如果 Pi 没有正常启动，请参考本章最后的“故障排除”一节的内容尝试解决问题。这些启动日志显示了 Pi 启动时所运行的所有进程，从中你可以看到网卡的初始化过程以及系统所识别出来的 USB 设备。系统启动完成后，你也可以通过在命令行上输入 `dmesg` 命令来查看这些启动日志。

第一次启动成功后，屏幕上会显示出 `raspi-config` 工具程序的界面（图 1.8），你需要在这里配置一些重要的运行参数，因为你的 Raspberry Pi 很可能不能在默认的配置下正常运行。如果今后你需要再次运行这个配置工具，可以在命令行上输入：

```
sudo raspi-config
```

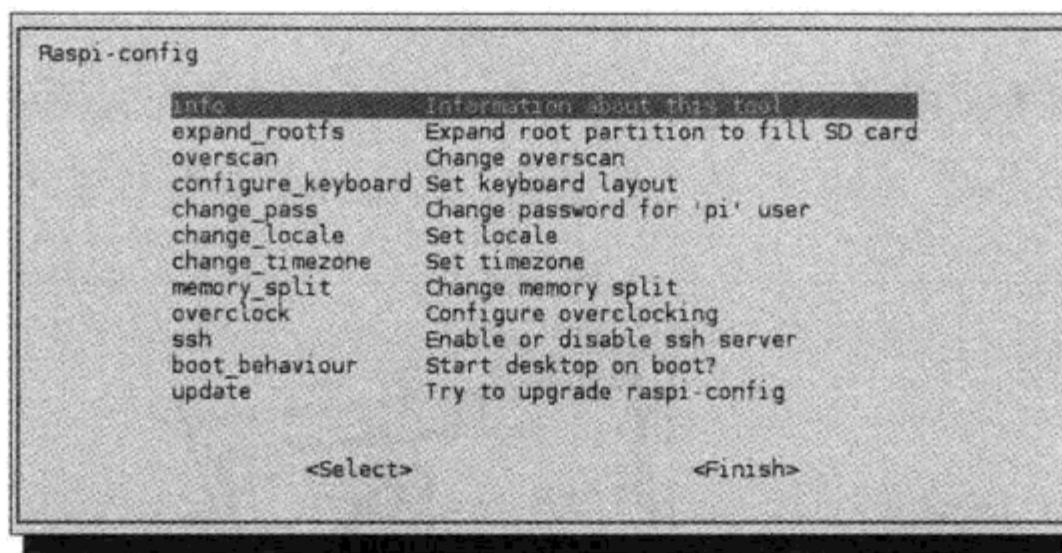


图1.8 raspi-config工具菜单

配置你的 Pi

下面我们会列出你可能需要修改的重要配置，需要时可以参考。在配置界面上，你可以用光标上下移动键来选择菜单中的不同项，按空格键确认选择，用 Tab 键在不同输入框中进行切换光标，用光标左右移动键选择窗口底部的不同按钮。下面，我们按菜单中的顺序依次介绍这些配置项^①。

Expand Rootfs（扩展根分区）

当你第一次启动 Pi 时，总是需要执行一下这个选项。这个选项执行后，可以把你的文件系统扩展到整个 SD 卡容量的大小。

Overscan（过扫描）

你可以暂时先把这个选项禁用。如果你正在使用一个高分辨率的显示器，可能会发现有些文字跑到了显示区域的外面。要解决这个问题，可以启用 Overscan 选项并修改配置值使显示的画面撑满整

^① 最新版本的 Raspbian 系统中的配置工具界面产生了一些变化，但基本功能和操作方式仍可参考这里的描述。——译者注



个屏幕，这个配置值决定了显示驱动需要进行过扫描的量。如果屏幕上无法显示完整的画面，把这个值设置为正数；如果画面边上留有黑边，把这个值设为负数。

Keyboard (键盘)

默认的键盘设置是普通英式键盘布局。为了让键盘上按下的键与显示在屏幕上的字符一致，你需要把键盘布局设置成与所使用的键盘实际布局一致。幸运的是这个键盘布局列表非常完整，不必担心在里面找不到与你键盘布局一致的选项。注意，系统的区域设置也可能会影响你的键盘设置。

Password (密码)

最好把默认密码 `raspberry` 改为其他更复杂的密码，确保安全。

Change Locale (修改区域设置)

如果你在英国以外的地区，可以修改区域设置让它与你的语言和字符编码相匹配。默认的设置是标准 UTF-8 编码的英式英语 (`en_GB.UTF-8`)，如果你在美国，可以设为 `en_US.UTF-8`^①。

Change Timezone (修改时区)

通常你可能都需要设置一下正确的时区^②。

Memory Split (内存分配)

这个选项用于修改分配给 CPU 和 GPU 的内存比例。暂时你可以先保持默认的设定。

^① 在中国可以把区域设置设为 `zh_CN.UTF-8`，并把它设为默认区域，重启 Raspberry Pi 后再进入图形桌面环境就可以看到中文界面了。——译者注

^② 中国所在的时区是 `UTC+0800`，在时区设置中可以设为 `Asia/Shanghai`。——译者注



Overclock (超频)

你可以把处理器的运行速度设定为高于正常 700MHz 的值。如果你是初次使用，建议保持默认值或尝试 Medium (中等) 或 Modest (适中)。以后可以再考虑改成其他设置 (Turbo 模式可以让处理器运行在 1000MHz 的速度)。

SSH

这个选项用于启动 SSH (Secure Shell) 服务器，用于通过网络远程登录到 Raspberry Pi 上。这是一个非常有用的功能，建议你打开它。

Desktop Behavior (桌面行为)

这个选项用于设置是否默认启动图形化桌面环境，默认值是启动。如果你选择不启动，系统启动后就会停留在命令行模式下，你可以自己登录系统并手工启动图形化桌面环境：

```
raspberrypi login: pi
Password: raspberry
pi@raspberrypi ~ $ startx
```

当你进入图形界面后，你的命令行界面就会消失，你可以通过打开一个终端程序来使用命令行。点击屏幕左下角的桌面菜单，然后选择附件→LX 终端 (Accessories → LXTerminal)。

Update (更新)

如果你接入了 Internet，可以通过这个选项来更新配置工具。刚开始尝试使用 Pi 时，请暂时不要更新系统^①。

^① 可以用第 2 章中介绍的 apt-get 工具来手工更新系统：依次执行 sudo apt-get update 和 sudo apt-get upgrade，就可以把系统中的各软件包都更新到最新版本。——译者注



当你完成所需的设置后，选择“Finish”（完成），然后就会重新回到命令行模式下。输入：

```
pi@raspberrypi ~ $ sudo reboot
```

重启系统，使你的设置生效。如果一切顺利（并且你设置了自动启动到图形桌面环境），重启后你会看到运行在轻量化 X11 桌面环境（Lightweight X11 Desktop Environment, LXDE）下的 Openbox 窗口管理器。这时，你就完成了设置，并且 Pi 已经正常运行了！

关闭系统

Raspberry Pi 上没有电源按钮（虽然新版本的主板上有重启引脚）。正确的关机方式是在桌面环境中选择 Logout 菜单，然后选择 Shutdown 关闭系统。

如果在命令行环境下，也可以输入下面的命令来关机：

```
pi@raspberrypi ~ $ sudo shutdown -h now
```

请确保每次都正确地关闭系统（不要直接断开电源线）。在某些极端情况下，直接断开电源线非正常关闭系统可能会损坏 SD 卡上的文件。

故障排除

如果系统没有像你所设想的那样工作，也许是因为你犯了一些常见错误或是遗漏了某些步骤。请按下面列出的建议一一检查。

- SD 卡有没有稳妥地插入卡槽？是否使用了正确型号的



SD 卡？

- 磁盘镜像有没有正确地写入 SD 卡？可以尝试用另一个读卡器重写一次。
- 你是否打开了 SD 卡上的写保护？SD 卡的写保护开关是卡侧面的一个小滑块，很容易被误拨到写保护的状态上。
- 检查你下载的磁盘镜像文件是否完整。可以通过运行 SHA 校验工具计算磁盘镜像的校验值，并与公布在下载页面上的 40 个字符的校验值比较，检查它们是否一致。
- Pi 是否出现自动重启或间断性的问题？检查你的电源是否合格。如果电源不能提供足够的电流，系统就会看上去可以工作，但实际上很不稳定。
- 启动时是否出现了内核崩溃（Kernel Panic）？内核崩溃与 Windows 中的蓝屏死机很相似，通常由 USB Hub 上所连接的外部设备引起，可以尝试断开一些外部设备并重启。

如果所有这些都不能帮助你解决问题，可以上网参考 Raspberry Hub 的 Wiki 上的故障排除页面 (http://elinux.org/R-Pi_Troubleshooting)，看看别人都遇到了哪些问题，以及是如何解决的。

你的主板是哪个版本的？



如果你在网络上通过论坛或电子邮件寻求帮助，明确你所使用的主板版本和操作系统版本，会让帮助你的人更容易向你提供有价值的信息。要查看操作系统版本，可以打开 LX 终端（LXTerminal）运行：

```
cat /proc/version
```

要查看主板版本，运行：

```
cat /proc/cpuinfo
```



进一步学习

The Raspberry Pi Hub

(http://elinux.org/RPi_Hub)

由 elinux.org 主办，提供有关 Pi 的硬件和配置信息的 Wiki。

已知可用的外设列表

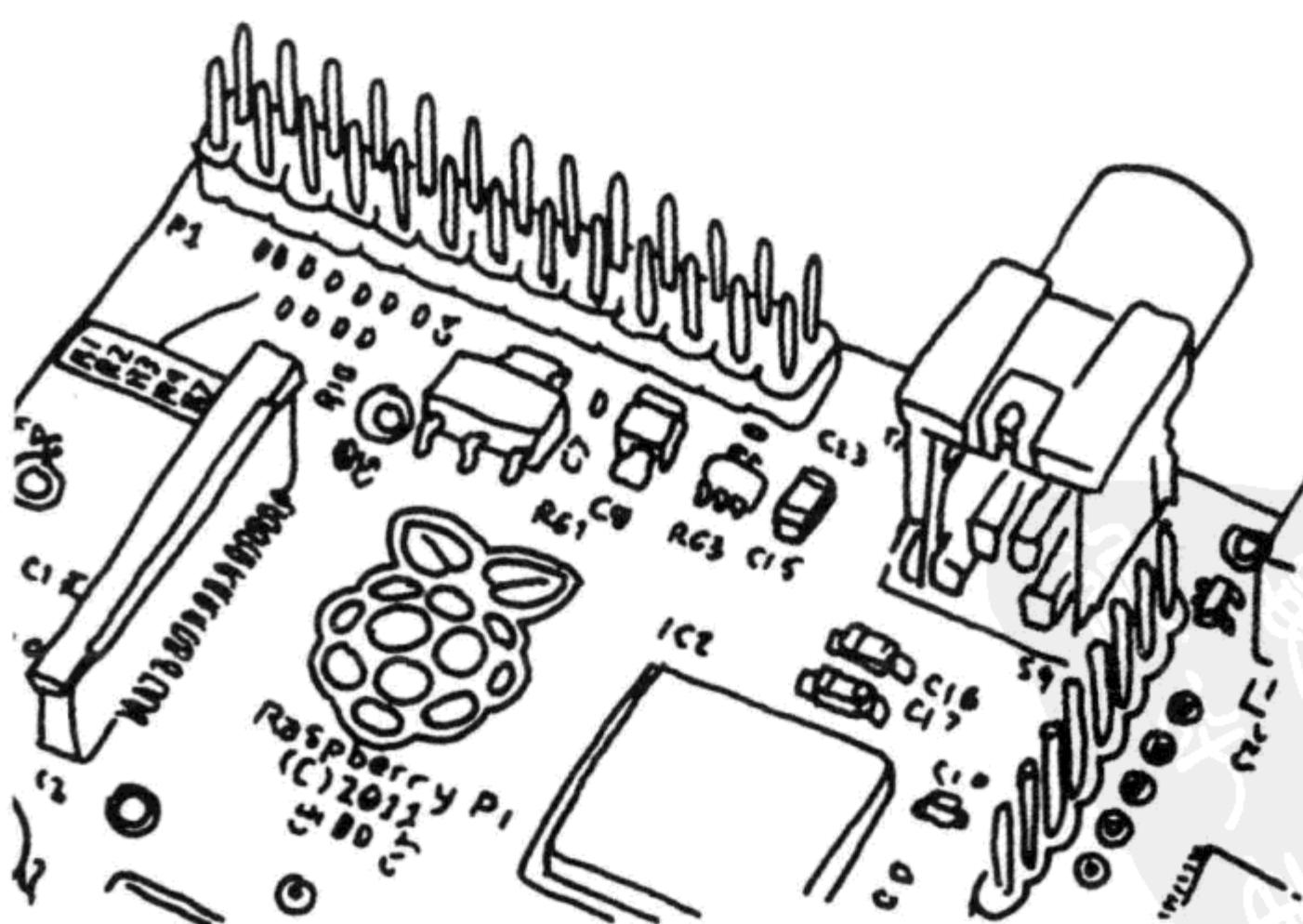
(http://elinux.org/RPi_VerifiedPeripherals)

一个已知可以在 Raspberry Pi 正常工作的外部设备列表。

第 2 章

初识 Raspberry Pi 上的 Linux

Getting Around Linux on the
Raspberry Pi





学习一点 Linux 的基本知识，可以帮助你从 Raspberry Pi 中获得更多的收获。本章的目标是带你快速领略 Linux 操作系统的基本知识，让你学会用基本的命令来操作文件系统、用命令或图形界面安装软件包，以及使用一些最常用的工具。

Raspbian 预装了名为“轻量级 X11 桌面环境”（Lightweight X11 Desktop Environment, LXDE）的图形化桌面环境（图 2.1）。这是一个基于 X Window 系统精简过的桌面系统，X Window 系统从 20 世纪 80 年代以来就是 UNIX 和 Linux 系统上最常用的图形界面。你在桌面或菜单中看到的部分软件是与 LXDE 绑定在一起的（如 Leafpad 文本编程器和 LX 终端）。

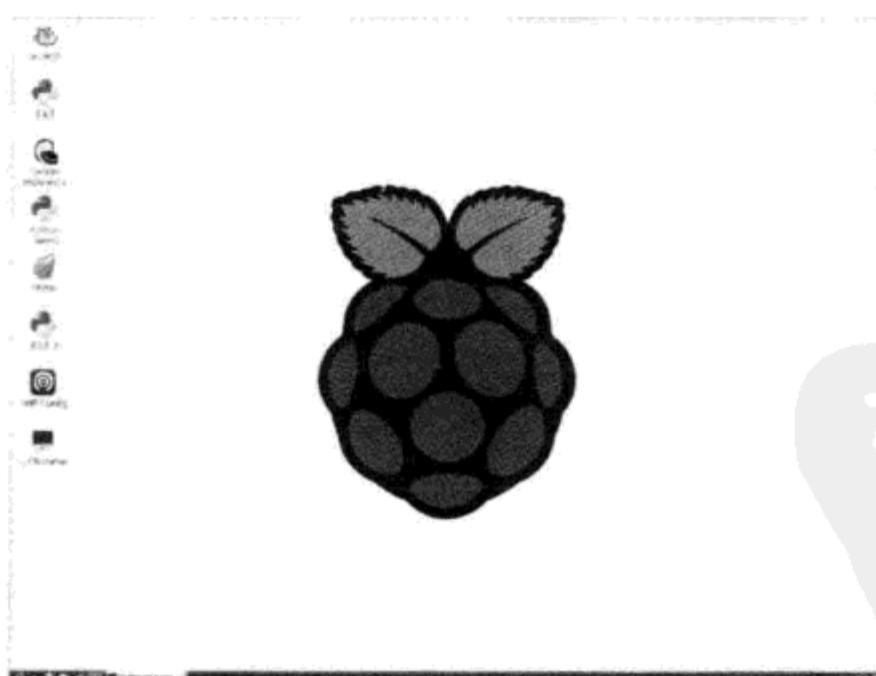


图2.1 图形化桌面

Openbox 窗口管理器在 LXDE 的基础上运行，它决定了窗口



和菜单的外观。如果你想调整桌面的外观，可以打开 Openbox 配置工具：点击屏幕左下角的桌面菜单，然后选择其他（Other）→ Openbox Configuration Manager。与 OS X 或 Windows 相比，在 Linux 下可以更容易自定义桌面环境或安装其他的窗口管理器。Raspberry Pi 的其他一些 Linux 发行版提供了不同的图形界面，有的适用于媒体中心，有的适用于电话系统，还有一些适用于网络防火墙。请参考 http://elinux.org/RPi_Distributions 上的列表。

文件管理器（File Manager）

如果不想用命令行来移动文件（稍后会介绍命令行的操作），从附件（Accessories）菜单中选择文件管理器（File Manager），然后就可以通过你所熟悉的文件图标和文件夹的形式来浏览文件系统了。

网络浏览器

默认的网络浏览器是 Midori (<http://twotoasts.de/index.php/midori/>)，这个浏览器只需占用很少的系统资源就可以运行。如今大家都已经熟悉了浏览器，便忽视了浏览器界面背后所发生的复杂的处理过程。由于 Raspbian 被设计成一个非常轻量级的操作系统，所以它自带的浏览器也缺少一些常见的功能：不支持 Flash 和 Java 插件（所以就看不了 YouTube），也不支持基于 HTML5 的视频。后面我们会学习如何安装一些新的软件（如 Java）。可以在 Midori 窗口右上角的下拉菜单中找到常用的工具和菜单项（图 2.2）。你也可以安装一些其他的浏览器软件，如 NetSurf (<http://www.netsurf-browser.org/>) 或 Dillo (<http://www.dillo.org/>)。

视频与音频

截至本书写作时为止，多媒体播放器主要是 Omxplayer，它还



处在一个实验阶段。Omxplayer 是针对 Raspberry Pi 的处理器中所集成的图形处理单元（Graphics Processing Unit, GPU）而设计的，其他的自由软件（如 VLC 或 mPlayer）并不能与 Raspberry Pi 的 GPU 良好地配合工作。

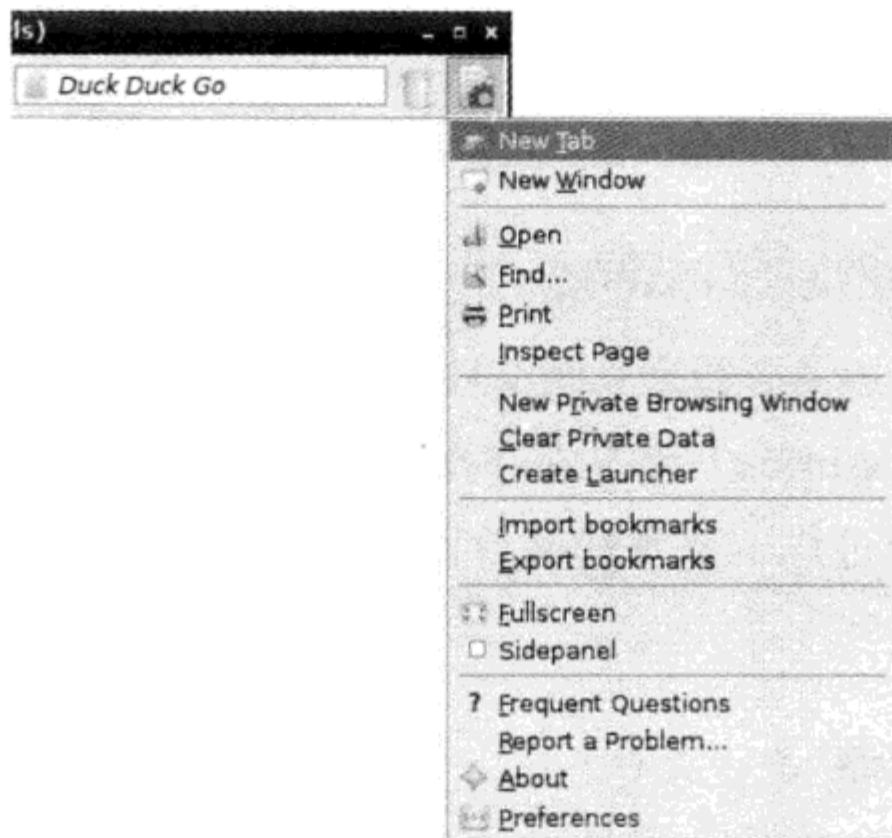


图2.2 Web浏览器的下拉菜单



为了降低成本，Raspberry Pi 没有包含部分视频解码程序的许可证。如果你想观看 MPEG-2 格式（或微软的 VC-1 格式）录制的电视或 DVD，则需要从 Raspberry Pi 基金会的在线商店 (<http://www.raspberrypi.com/>) 购买一个许可证密钥。Raspberry Pi 自带了 H.264 (MPEG-4) 编解码器的许可证。

文本编辑器

Leafpad (<http://wiki.lxde.org/en/Leafpad>) 是系统默认的编辑器，



可以在主菜单下找到。你也可以使用 Nano (<http://www.nano-editor.org/>)，这是一个文本界面的编辑器，非常易学易用。系统默认没有安装传统 UNIX 系统下的文本编辑器 Vim 和 Emacs，但它们都可以很方便地安装上（参考“安装新软件”）。

复制与粘贴

在应用程序之间复制、粘贴数据非常容易，不过也有一小部分古怪的软件不能完美地支持这个功能。如果鼠标上有中键，你可以像你所习惯的那样通过按住鼠标左键拖动来高亮选中一块文字实现复制，然后到目标窗口上点击鼠标中键把选中的文字粘贴过来。

Shell 程序

很多任务需要你切换到命令行模式下，通过运行命令去实现。LX 终端（LXTerminal）程序提供了在图形界面下访问命令行终端的能力。Raspbian 上默认的 Shell 程序是 Linux 上最常见的 Bourne Again Shell（bash，<http://www.gnu.org/software/bash/manual/bashref.html>），也可以使用另一个名叫 Dash（http://en.wikipedia.org/wiki/Debian_Almquist_shell）的 Shell 程序。你可以通过程序菜单来切换 Shell 程序，也可以通过运行 `chsh` 命令来切换。

使用命令行

为了让学习使用命令行看上去更有意思一些，你可以把操作命令行想象成玩文字冒险游戏，游戏场景发生在文件与文件系统的迷宫中。如果你觉得这个想象一点也不吸引你，也没关系，本节中所介绍的命令和概念都是标准 Linux 系统中所使用的，非常值得去学习。



在开始前，先打开一个 LX 终端（LXTerminal）（图 2.3）。在操作终端时，有两个基本技巧非常有用：自动补全与命令历史。通常情况下，你只需输入命令或文件名的前几个字符，然后按 Tab 键，终端就会尝试以当前目录中的文件或常用目录中的程序（Shell 通常会在 `/bin` 和 `/usr/bin` 目录搜索可执行程序）来补全你输入的字符串。如果你按键盘上的上下箭头键，可以调出以前执行过的命令。如果你不小心在命令中打错了字，可以用这个方法把错误的命令找出来修改后再重新执行。

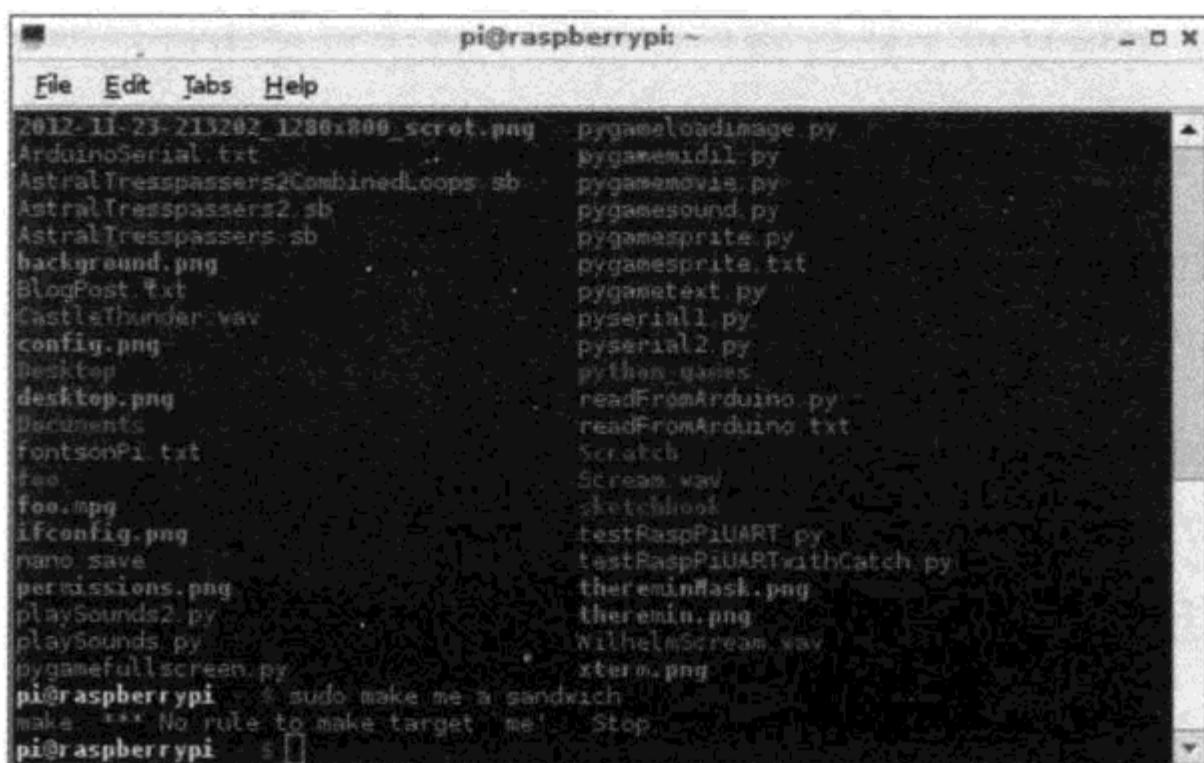


图2.3 通过LX终端（LXTerminal）访问命令行（Shell）

文件与文件系统

表 2.1 列出了文件系统中的部分重要目录，它们中的大部分都是按照 Linux 标准来设置的，标准决定了一些文件应该放在哪个目录中；其中有一部分则是 Raspberry Pi 所特有的。通过访问 `/sys` 目录中的文件，你可以操作 Raspberry Pi 上的所有硬件。



表2.1 Raspbian文件系统中部分重要的目录

目录	描述
/	
/bin	所有用户都可以运行的程序和命令
/boot	系统启动时所需要的文件
/dev	表示你系统上的各种设备的特殊文件
/etc	配置文件
/etc/init.d	启动各种服务的脚本
/etc/X11	X11 配置文件
/home	各用户的主目录
/home/pi	pi 用户的主目录
/lib	内核模块与驱动程序
/media	可移动存储的挂载点
/proc	包含操作系统和各个运行中的进程相关信息的虚拟目录
/sbin	用于系统管理与维护的程序
/sys	Raspberry Pi 上的一个特殊的目录, 可用于操作设备
/tmp	用于让各个程序创建临时文件
/usr	所有用户都可以使用的程序和数据
/usr/bin	包含了操作系统中大部分的程序
/usr/games	游戏都在这里
/usr/lib	用于支持常用程序的库文件
/usr/local	存放一些这台机器专用的软件
/usr/sbin	更多系统管理程序
/usr/share	一些在多个程序间共享的数据, 如图标与字体
/usr/src	Linux 是开放源代码的, 这里就包含了它的源代码
/var	系统日志或脱机缓存文件
/var/backups	一些系统关键文件的备份
/var/cache	存放应用程序 (如 apt-get 和网页浏览器) 缓存文件的目录
/var/log	所有的系统及服务程序的日志
/var/mail	如果配置了电子邮件功能, 所有用户的邮件都保存在这里
/var/spool	待处理的数据 (如接收到的电子邮件、打印任务)



在命令提示符前面，可以看到你当前所处的工作目录。在 Linux 中，你的主目录会被缩写为波浪线 (~)。刚刚打开 LX 终端 (LXTerminal) 时，它会把当前的工作目录切换到你的主目录下，命令提示符会显示为

```
pi@raspberrypi ~ $
```

有关命令提示符的详细解释：

```
pi@① raspberrypi ② ~③ $④
```

- ① 你的用户名：pi，后面加上 @ 符号。
- ② 你的主机名（默认的主机名是 raspberrypi）。
- ③ Shell 的当前工作目录。默认是你的主目录 (~)。
- ④ 这是命令提示符。你输入的一切内容都会出现在它的右侧，按回车键执行你所输入的命令。

使用 cd (change directory，改变目录) 命令可以在文件系统的各个目录间切换，下面的两条命令对于 pi 用户来说具有相同的效果（把当前工作目录切换到主目录）：

```
cd /home/pi/  
cd ~
```

如果传给 cd 命令的路径以斜线开头，则表示这是一个绝对路径；否则这个路径被认为是相对于当前工作目录的位置的相对路径。可以用 . 和 .. 分别表示当前目录和上一层目录。例如，把当前目录切换到系统根目录：

```
pi@raspberrypi ~ $ cd ..  
pi@raspberrypi /home $ cd ..
```



你也可以直接用 / 切换到根目录：

```
pi@raspberrypi ~ $ cd /
```

切换完目录后，可以用 ls 命令列出当前目录下的文件：

```
pi@raspberrypi / $ ls
bin  dev  home  lost+found  mnt  proc  run  selinux  sys  usr
boot  etc  lib  media  opt  root  sbin  srv  tmp  var
```

大部分命令都允许添加一些参数或开关来改变默认的行为。例如，ls 命令的 -l 参数可以让显示的文件列表更为详尽，显示出文件的大小、修改时间和权限：

```
pi@raspberrypi ~ $ ls -l
total 8
drwxr-xr-x 2 pi pi 4096 Oct 12 14:26 Desktop
drwxrwxr-x 2 pi pi 4096 Jul 20 14:07 python_games
```

用 -a 参数可以列出所有文件，包含隐藏文件：

```
pi@raspberrypi ~ $ ls -la
total 80
drwxr-xr-x 11 pi pi 4096 Oct 12 14:26 .
drwxr-xr-x  3 root root 4096 Sep 18 07:48 ..
-rw-------  1 pi pi 25 Sep 18 09:22 .bash_history
-rw-r--r--  1 pi pi 220 Sep 18 07:48 .bash_logout
-rw-r--r--  1 pi pi 3243 Sep 18 07:48 .bashrc
drwxr-xr-x  6 pi pi 4096 Sep 19 01:19 .cache
drwxr-xr-x  9 pi pi 4096 Oct 12 12:57 .config
drwx-----  3 pi pi 4096 Sep 18 09:24 .dbus
drwxr-xr-x  2 pi pi 4096 Oct 12 14:26 Desktop
-rw-r--r--  1 pi pi 36 Sep 18 09:35 .dmrc
drwx-----  2 pi pi 4096 Sep 18 09:24 .gvfs
drwxr-xr-x  2 pi pi 4096 Oct 12 12:53 .idlerc
-rw-------  1 pi pi 35 Sep 18 12:11 .lessshst
drwx-----  3 pi pi 4096 Sep 19 01:19 .local
```



```
-rw-r--r-- 1 pi pi 675 Sep 18 07:48 .profile
drwxrwxr-x 2 pi pi 4096 Jul 20 14:07 python_games
drwx----- 4 pi pi 4096 Oct 12 12:57 .thumbnails
-rw----- 1 pi pi 56 Sep 18 09:35 .Xauthority
-rw----- 1 pi pi 300 Oct 12 12:57 .xsession-errors
-rw----- 1 pi pi 1391 Sep 18 09:35 .xsession-errors.old
```

使用 `mv` 命令可以改变文件的名字。使用 `touch` 命令可以用于创建一个新的空文件。

```
pi@raspberrypi ~ $ touch foo
pi@raspberrypi ~ $ ls
foo Desktop python_games
pi@raspberrypi ~ $ mv foo baz
pi@raspberrypi ~ $ ls
baz Desktop python_games
```

删除文件用 `rm` 命令。要删除空目录，可以使用 `rmdir` 命令。如果要删除非空目录，使用 `rm -r` 命令。传给 `rm` 命令的 `-r` 参数意味着要求 `rm` 程序递归进入所有的子目录删除所有的文件。

如果想了解一个命令所包含的所有参数，可以使用 `man` 命令（或在命令后面添加 `--help` 参数）：

```
pi@raspberrypi ~ $ man curl
pi@raspberrypi ~ $ rm --help
```

要创建新的目录，使用 `mkdir` 命令。要把一个目录下的所有文件打包在一起，用 `tar` 命令——这个名字来源于“磁带存档”（tape archives）的缩写。你常常能看到很多文件与源代码以 `tar` 包的形式发布，并且它们常常还会用 `gzip` 进行一次压缩。练习一下下面的操作：

```
pi@raspberrypi ~ $ mkdir myDir
pi@raspberrypi ~ $ cd myDir
```



```
pi@raspberrypi ~ $ touch foo bar baz
pi@raspberrypi ~ $ cd ..
pi@raspberrypi ~ $ tar -cf myDir.tar myDir
pi@raspberrypi ~ $ gzip myDir.tar
```

这时,你就把*myDir*目录打包成了一个名为*myDir.tar.gz*的文件,这个文件可以很方便地通过网络或电子邮件传递。

更多 Linux 命令

Linux (与 UNIX) 如此成功的原因之一在于它的设计哲学:通过一系列可以组装在一起的简单小模块来构建一个非常复杂的系统。所以,你需要学习一些有关管道和重定向的基础知识,它们是实现这个设计哲学的基础。

管道是把两个程序联系起来的一种方式,通过管道,可以把其中一个程序的输出作为另一个程序的输入。所有的 Linux 程序都可以从标准输入 (通常表示为 `stdin`) 读入数据,向标准输出 (表示为 `stdout`) 输出数据并把错误信息抛向标准错误输出 (表示为 `stderr`)。通过管道可以把一个程序的 `stdout` 与另一个程序的 `stdin` 相连 (图 2.4)。管道的操作号是 `|`,如:

```
pi@raspberrypi ~ $ ls -la | less
```

(按 Q 键退出 `less` 程序)

还可以尝试一下通过管道传输更多的数据:

```
pi@raspberrypi ~ $ sudo cat /boot/kernel.img | aplay
```

运行这个命令前,建议先把音量调小一些。这个命令把系统内



核对应的 0 和 1 的数据传递给音乐播放器，当成音乐来播放。这就是“内核之声”。

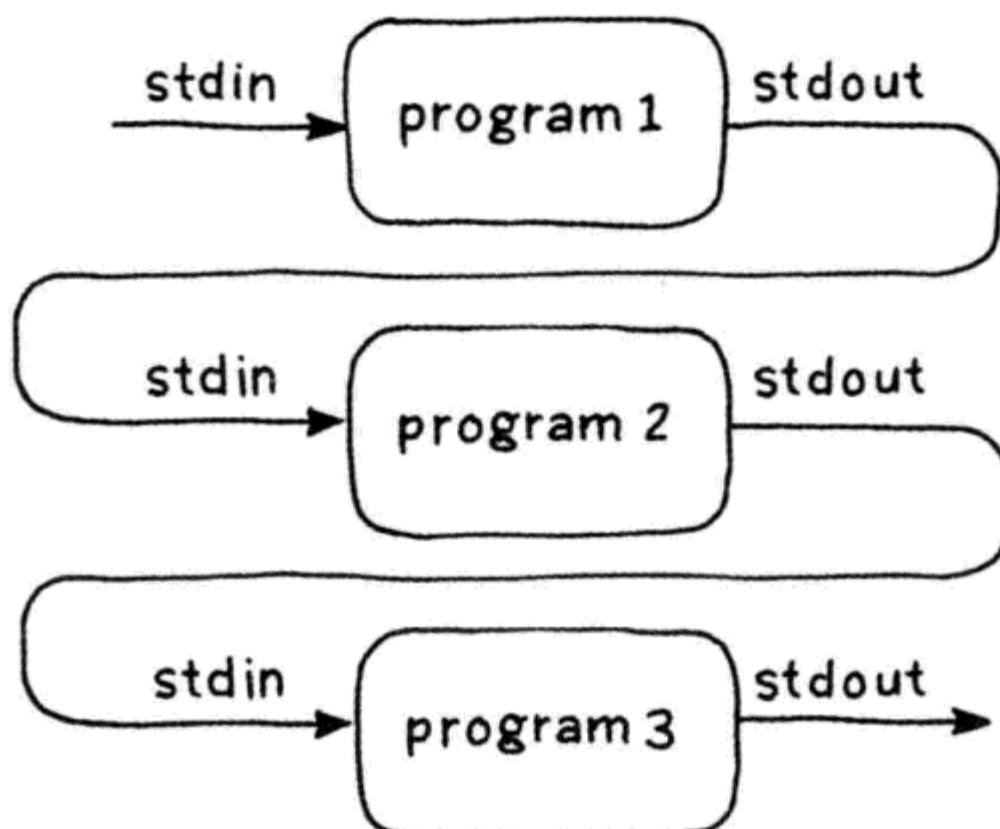


图2.4 管道可以用于把小程序连接在一起形成一个更大的任务

本书后面章节中的命令还会用到重定向，也就是指把程序的标准输出发送到一个文件中去。在后面你会看到，Linux 中很多的东西都可以被当成文件来使用（如 Pi 的通用输入输出口），所以重定向是一个非常有用的功能。使用 > 操作符来实现重定向：

```
pi@raspberrypi ~ $ ls > directoryListing.txt
```

特殊的控制键

除了先前提到的自动补全 (Tab 键) 和调用历史命令 (光标向上键) 的控制键，Shell 中还提供了其他一些有用的控制键。



Control-C

中断当前正在运行的程序。在有些交互式应用程序（如文本编辑器）中可能无效。

Control-D

退出 Shell。必须在命令提示符后紧跟着输入才有效（在 \$ 提示符后不要输入 Control-D 以外的其他任何字符）。

Control-A

把光标移动到行首位置。

Control-E

把光标移动到行末位置。

除此之外，还有很多可以使用的快捷键，不过上面列出的是日常操作中最为常用的。

需要在屏幕上显示一个文件的内容，并且要分屏显示时，可以使用 `less` 命令：

```
pi@raspberrypi ~ $ ls > flob.txt
pi@raspberrypi ~ $ less flob.txt
```

如果想把一个文件的内容一次性全部输出到标准输出上，可以用 `cat` 命令（`concatenate` 的缩写）。在需要把一个文件的内容传到另一个程序中或者把它重定向到其他地方时，这个命令非常有用。

例如，用 `cat` 命令可以非常方便地实现把一个文件复制为另一个文件的功能（第二个 `cat` 命令中，把两个文件连接成一个新的文件）。

```
pi@raspberrypi ~ $ ls > wibble.txt
pi@raspberrypi ~ $ cat wibble.txt > wobble.txt
pi@raspberrypi ~ $ cat wibble.txt wobble.txt > wubble.txt
```



用 **tail** 命令可以查看一个文件的最后几行内容（如一个日志文件中的最后几条），用 **head** 命令则可以查看前几行。如果要在一个或多个文件中搜索一个特定的字符串，可以用强大的 **grep** 命令：

```
pi@raspberrypi ~ $ grep Puzzle */*
```

由于 **grep** 命令能支持正则表达式丰富的语意，这帮助它成了一个非常强大的工具。正则表达式不太容易让人理解，这也是很多人认为 Linux 对初学者来说很难用的一个重要原因。

进 程

Pi 上运行的每一个程序都是一个独立的进程，所以任何时候总是有很多个进程同时在运行。当你刚把 Pi 启动完成时，系统会运行大约 75 个进程，每个进程负责一项任务或提供一个服务。可以用 **top** 命令查看正在运行的进程，与此同时也可以看到 CPU 和内存的使用情况。**top** 命令能让你看到占用资源最多的进程，**ps** 命令可以用来列出所有的进程和它们对应的进程 ID (PID)：

```
pi@raspberrypi ~ $ ps -aux | less
```

如果要中断一个出错或失去响应的进程，可以用 **ps** 命令找出这个进程的 ID，然后用 **kill** 命令来中断它的运行：

```
pi@raspberrypi ~ $ kill 5689
```

对于一些系统进程，你是没有权限去中断它的运行的，但可以通过稍后介绍的 **sudo** 命令来获得需要的权限。



sudo 与权限

Linux 是一个多用户操作系统，一条最基本的原则就是每个用户可以在他们自己的目录下创建、删除或修改属于他们自己的文件。**root** 用户（管理员）可以修改文件系统中的任何文件，为了规避错误地对文件进行操作，日常使用时应该尽量避免使用 **root** 用户登录。



如果以 **pi** 用户登录，你很难对系统造成什么致命性的破坏。但以管理员身份操作则可能对系统造成严重的破坏，这也许是不小心的错误操作引起的，也有可能是故意的。所以，在使用 **sudo** 时应该特别小心，尤其是移动或删除文件时，更应慎之又慎。当然，如果你确实已经把系统破坏了，也还是可以通过参考附录 A 中的方法重新在 SD 卡上安装一个新的系统。

通过使用 **sudo** 或类似的工具可以使普通用户临时获得管理员权限，用来执行安装软件包之类的操作，这样就可以规避直接以 **root** 用户登录所带来的安全风险。在修改系统配置或与硬件打交道时，你常常会使用到 **sudo**。

系统中的每一个文件都属于某一个用户和某一个用户组，用 **chown** 和 **chgrp** 命令可以修改文件所属的用户或用户组。要执行这两个命令，你必须拥有 **root** 权限。

```
pi@raspberrypi ~ $ sudo chown pi garply.txt
pi@raspberrypi ~ $ sudo chgrp staff plugh.txt
```

每个文件都被设置了相关的权限，决定了一个文件是否可以被



读、写或执行。这些权限可以按文件的拥有者、文件所属的用户组或全部用户分别设置（图 2.5）。

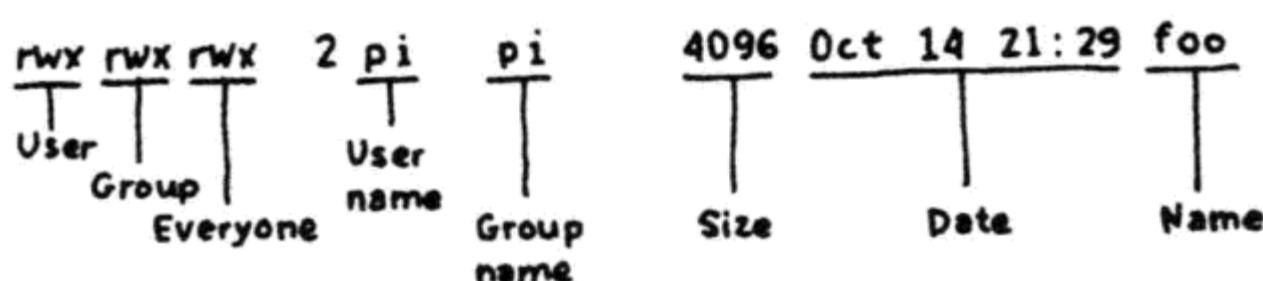


图2.5 对文件所有者、文件所属组和全部用户的权限设置

用 `chmod` 命令可以设置文件的权限。相关的选项如表 2.2 所示。

表2.2 chmod命令中使用的选项

u	用户
g	用户组
o	所属用户组以外的用户
a	所有用户
r	读权限
w	写权限
x	执行权限
+	添加权限
-	去除权限

下面是把这些选项组合在一起使用的一些例子：

```
chmod u+rwx,g-rwx,o-rwx wibble.txt ①
chmod g+wx wobble.txt ②
chmod -rw,+r wubble.txt ③
```

① 只允许文件所有者读、写和执行。



② 允许文件所属用户组的用户写和执行。

③ 该文件对所有用户都是只读。

密码是保护你的个人空间与文件的唯一安全措施，所以你应该设置较为复杂的密码，尤其是把 Pi 接入网络的情况下。可以用 `passwd` 命令来修改密码。

网 络

如果你的 Pi 已经接入网络，就会经常用到一些网络相关的 Linux 工具。如果你需要诊断网络连接的故障，可以用 `ifconfig` 命令，这个命令可以用于显示你所有的网络接口以及它们上面所绑定的 IP 地址（图 2.6）。

```
pi@raspberrypi ~ $ ifconfig
eth0      Link encap:Ethernet HWaddr b8:27:eb:de:7a:b6
          inet addr:192.168.2.2  Bcast:192.168.2.255  Mask:255.255.255.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:7632 errors:0 dropped:0 overruns:0 frame:0
          TX packets:7050 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:558086 (545.0 KiB)  TX bytes:1680110 (1.6 MiB)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:14 errors:0 dropped:0 overruns:0 frame:0
          TX packets:14 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:1432 (1.3 KiB)  TX bytes:1432 (1.3 KiB)
```

图2.6 用ifconfig命令可以显示出网卡的所有信息

在诊断网络连接问题时，`ping` 命令是最基本的工具。你可以使用 `ping` 命令（像声呐那样）来检测 Internet 或局域网中两个 IP 地址之间是否可以相互连通。不过要注意，有些网站会屏蔽 `ping` 的检测流量，所以你也许需要多 `ping` 几个网站才能确定网络是否确实存在问题。



```
ping yahoo.com
ping altavista.com
ping lycos.com
ping netscape.com ①
```

① 这个 ping 命令会失败。

Secure Shell (SSH) 可以用于安全地（会被加密）远程登录另一台电脑，前提是远程电脑上启动了 SSH 服务。Raspbian 中已经预装了 SSH 客户端程序。通过 SSH 远程访问 Raspberry Pi 就不需要在 Pi 上连接显示器和键盘了，这就是我们稍后会介绍的“无显示器运行”方式。

与 SSH 相关的还有 `sftp` 程序，可以用于两台电脑之间安全地传输文件。除此之外还有 `scp` 命令，可以通过局域网或 Internet 在两台电脑之间复制文件。这些工具都是基于安全套接字层（Secure Sockets Layer, SSL）协议来开发的，可以保证密码和传输过程的安全。这些都是标准的 Linux 工具。

无显示器运行

如果你想在不连接显示器、键盘和鼠标的情况下使用 Raspberry Pi：即所谓“无显示器运行”，是完全可行的。如果只是需要使用命令行终端，你只需把 Raspberry Pi 接上网线，就可以用 SSH 客户端程序去连接它了（默认的用户名为 `pi`，密码为 `raspberry`）。在 Mac 上，可以使用 Terminal 工具，Windows 和 Linux 上可以使用 PuTTY 工具 (<http://www.chiark.greenend.org.uk/~sgtatham/putty/>)，Linux 上也可以使用 `ssh` 命令。Raspberry Pi 上的 SSH 服务默认就是启用的（需要的话可以通过运行 `raspi-config` 命令把它设置为开机



默认不启动）。

另外一种通过网络使用 Raspberry Pi 的方式是使用 VNC（虚拟网络计算，Virtual Network Computing）服务器。使用 VNC 的好处在于，你可以在你的笔记本或台式电脑上操作 Raspberry Pi 上的图形界面。这种方式可以很好地作为便携开发环境的解决方案。参考 Raspberry Pi Hub 网页 (http://elinux.org/RPi_VNC_Server) 的介绍设置 TightVNC (<http://www.tightvnc.com/>)，TightVNC 是一个轻量级的 VNC 服务器。

/etc 目录

/etc 目录中存放了所有的系统全局配置文件和启动脚本，在你第一次启动 Raspberry Pi 并通过配置工具修改系统配置时，实际上就是修改了 /etc 目录下的文件。你需要通过 sudo 获取管理员权限才能修改 /etc 目录下的文件，所以如果看到某些教程上提到要修改配置文件，就需要通过 sudo 启动一个文本编辑器来修改这些文件：

```
pi@raspberrypi ~ $ sudo nano /etc/hosts
```

设置日期时间

一般的笔记本电脑或台式电脑会有额外的硬件和备用电池（通常是一个纽扣电池）用于记录当前的日期时间，Raspberry Pi 没有。但是 Raspbian 系统被配置为一旦接入网络就自动通过网络时间协议（Network Time Protocol，NTP）来同步系统日期和时间。

把系统时间设置为正确的值对于一些软件来说非常重要（如第 7 章中提到的用 cron 服务来做一个定时台灯）。如果要手工设置日期时间，可以用 date 程序：



```
$ sudo date --set="Sun Nov 18 1:55:16 EDT 2012" ①
```

安装新软件

强大的软件包管理是 Linux 完胜于其他操作系统的原因之一。包管理程序负责下载和安装新的软件，并自动解决软件之间的依赖关系。为了让系统做到模块化，Linux 下的很多软件包在安装时都会依赖其他的软件包。包管理程序让这种依赖关系很容易得到自动解决，并且包管理程序自身也非常可靠。

Raspbian 默认只安装了很少的一些软件，所以你很快就会遇到需要下载和安装软件的问题。在本书的示例中，我们使用命令行来安装软件，因为这是最灵活也是最便捷的方法。

`apt-get` 程序带上 `install` 参数是用于下载软件包的，`apt-get` 会自动下载安装你想要的软件所必需依赖的其他软件包，这样你就无须人工去到处寻找那些依赖包了。安装软件需要系统管理员权限，所以运行 `apt-get` 时总是需要带上 `sudo` 前缀（下面的命令用于安装 Emacs 编辑器）：

```
pi@raspberrypi ~ $ sudo apt-get install emacs
```

屏幕截图



在我们写这本书时，非常需要一个 Pi 上的屏幕截图工具。我们找到了一个名叫 `scrot`（是 SCReenshOT 的缩写）的工具。当然，你也可以安装 GNU Image Manipulation Program（GIMP）或 ImageMagick 来获取屏幕截图，不过

① 这条命令中的 EDT 是 Eastern Daylight Time 的缩写，在美国等国家表示使用东部夏令时。当系统时区设置为中国所对应的时区时，可以使用中国标准时间（China Standard Time）的缩写 CST 来替换它。——译者注



我们用 scrot 已经足够。安装 scrot 的命令：

```
sudo apt-get install scrot
```

进一步学习

通过很多渠道可以学习到更多的 Linux 相关知识，你可以尝试从以下地方开始。

Linux Pocket Guide

(<http://shop.oreilly.com/product/0636920023029.do>)

非常有用的快速参考书。

Linux in a Nutshell

(<http://shop.oreilly.com/product/9780596154493.do>)

比上一本要详细一些，但仍然是一本快速参考手册。

The Debian Wiki

(<http://wiki.debian.org/FrontPage>)

Raspbian 是基于 Debian 开发的，所以 Debian Wiki 上的很多信息对 Raspbian 同样适用。

The Jargon File

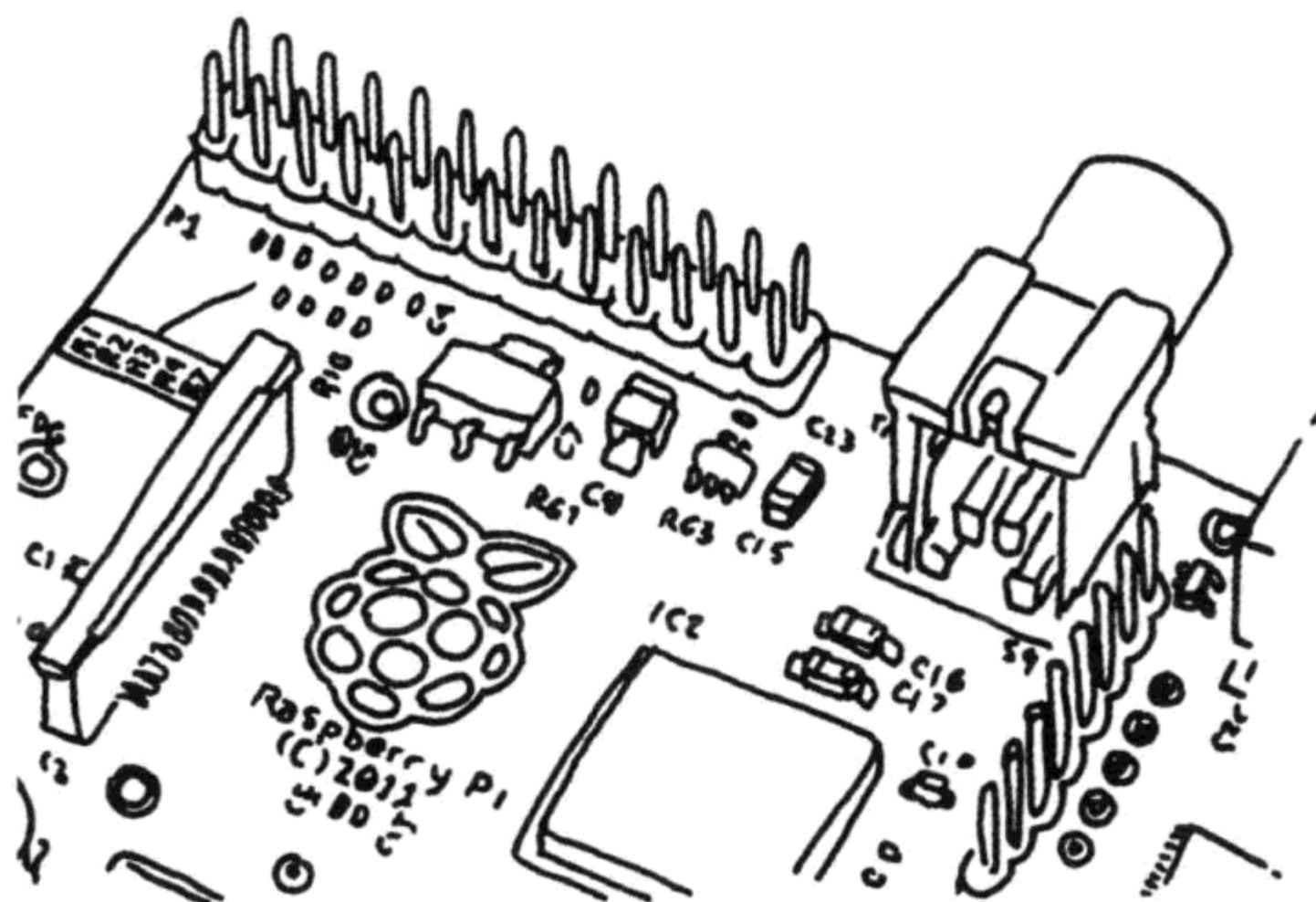
(<http://catb.org/jargon/>)

曾经以 *New Hacker's Dictionary* 的书名出版，里面收录了值得了解的一些定义和故事，属于 UNIX/Linux 的亚文化。

第3章

Pi 上的 Python

Python on the Pi





把 Python 作为入门语言非常合适，它的代码非常清晰，安装和设置运行环境也很容易。更重要的是，它有一个庞大的用户群，大家可以在一起分享代码或共同分析、解决问题。

Guido van Rossum 创造了 Python 语言，并在很早的时候就把它设计成一门适合入门的计算机语言。1999 年，van Rossum 提出一项叫作“人人能编程”（Computer Programming for Everybody，<http://www.python.org/doc/essays/cp4e.html>）的宏伟计划，计划用 Python 语言在中小学校开展编程教学。十多年时间过去，Raspberry Pi 的出现让这个计划得以实现。

Python 是一种解释性语言，你所编写的程序或脚本可以直接被执行，而不需要把它们先整体编译成机器码。解释性语言用起来很便捷，并且有一些额外的优点。例如，在 Python 中，你不需要显式地指定某一个变量是数字、列表还是字符串，解释器在执行你的脚本时会自动推断出变量的类型。

Python 的解释器有两种运行模式：既可以当成一个交互式的终端执行单条命令，也可以作为一个命令行工具运行独立的脚本。Raspberry Pi 上也提供了与 Python 绑定在一起的集成开发环境 IDLE（图 3.1）。

Python 的版本之谜

Pi 上预装了两个版本的 Python，所以在 Pi 上能看到两



套 IDLE 开发环境。这看起来有点让人糊涂，但却是一种常见的做法。原因在于，目前 Python 3 是最新的 Python 版本，但 Python 2 到 Python 3 之间发生了很多变化，并且并不向下兼容。虽然 Python 3 已经出现很久，但仍有很多软件包没有升级到 Python 3。当你查找 Python 的文档时尤其要注意区分不同的版本，避免造成混淆。

没有特别说明的话，本书中的代码既可以在 Python 2.7 上运行，也可以在 Python 3.x 上运行。

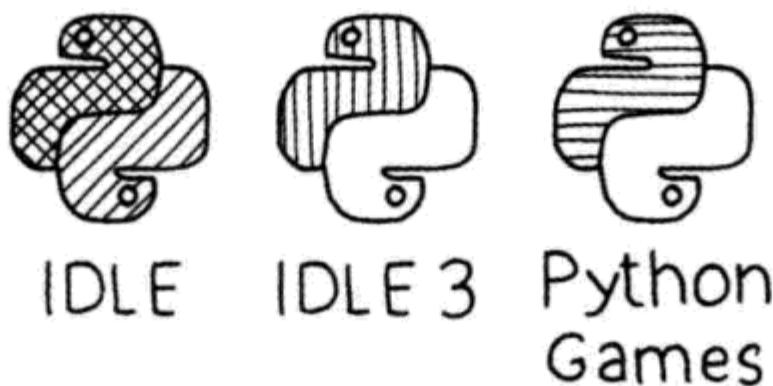


图3.1 Raspbian桌面上的Python图标：左侧是Python 2.0的IDLE，中间是Python 3.0的IDLE，右侧是集合了inventwithpython.com (<http://inventwithpython.com>) 上很多游戏的示例程序，这些游戏是通过Pygame开发的

初识 Python

学习 Python 最好的方式就是通过实践，虽然你可以用任何文本编辑器来编写脚本，但我们仍会从使用 IDE(集成开发环境)开始。可以双击桌面上的 IDLE 3 图标打开 IDLE 3 开发环境，也可以点击左下角的桌面菜单，选择编程 (Programming) → IDLE 3。

IDLE 启动需要花费几秒钟时间，当它启动后，你可以看到它的界面，里面包含了一个交互式的终端。行首显示的 3 个大于号



(>>>) 是命令提示符，当你看到这个提示符时，就表明解释器正在等待你输入命令。下面，尝试在提示符后输入：

```
>>> print("Saluton Mondo!")
```

按回车键，Python 就会执行你输入的命令，并在窗口中显示运行的结果。注意，`print()` 命令是在 Python 3.0 中引入的重要改变之一，如果你运行命令时出错了，请检查一下你运行的是不是 3.0 版本的 IDLE。

你可以把命令行当成计算器来计算表达式的值：

```
>>> 3+4+5  
12
```

你可以把命令提示符后面输入的每一条命令看成一个程序，只不过每次只运行了这个程序中的一行而已。你可以在命令行中创建变量或导入模块：

```
>>> import math  
>>> (1 + math.sqrt(5)) / 2  
1.618033988749895
```

`import` 命令把 Python 数学函数库的功能都导入你的程序供你使用（参考“对象与模块”了解更多具体的信息）。如果要创建变量，可以使用赋值运算符（=）：

```
>>> import math  
>>> radius = 20  
>>> radius * 2 * math.pi  
125.66370614359173
```

如果你想清除所有的变量并恢复到初始状态，选择 Shell → Restart Shell 即可。你可以通过 `help` 命令获取某个表达式、模块或



其他 Python 功能的描述：

```
help("print")
```

如果要显示主题的列表，可以运行：

```
help("topics")
help("keywords")
help("modules")
```

Python 解释器很适合用于测试表达式或进行简单的操作，但通常更希望把 Python 脚本当作一个程序来运行。要新建一个 Python 程序，选择菜单中的 File → New Window，IDLE 会帮你打开一个脚本编辑窗口（图 3.2）。

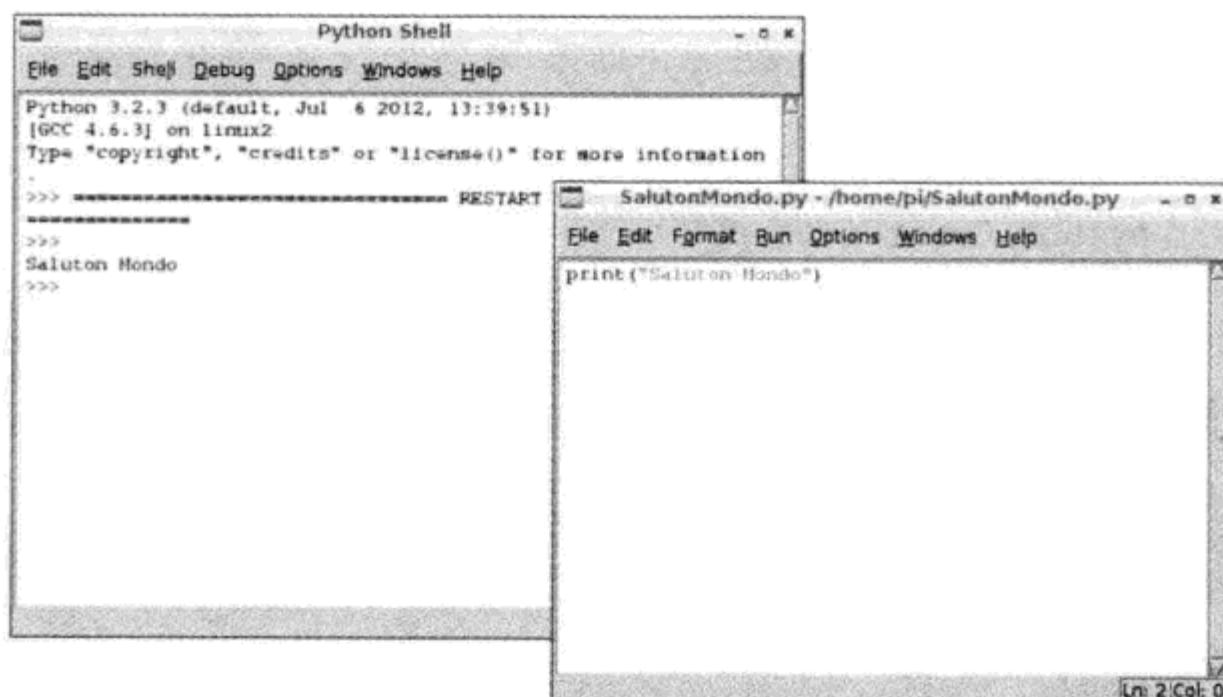


图3.2 IDLE的交互式命令行（左）与脚本编辑窗口（右）

你可以尝试输入一些命令，然后选择 Run → Run Module，系统会弹出一个警告框“Source Must Be Saved OK To Save?”。把这个脚本保存到你的主目录中，命名为 *SalutonMondo.py*，你就可以在终端中执行它了。



如果不想通过 IDLE 环境来运行程序，你也可以打开 LX 终端（LXTerminal），在命令行上输入：

```
python SalutonMondo.py
```

来运行这个程序。

前面介绍的都是最基本的环境使用知识，下面我们可以真正开始学习这门语言了。

命令行与 IDLE

你可能会发现，在 IDLE 中运行示例代码向终端输出内容时速度非常缓慢。要想知道有多慢，你可以同时打开一个 IDLE 窗口和一个 LX 终端（LXTerminal）窗口，在 IDLE 中把下一节中的 *CountEvens.py* 脚本保存为文件并在命令行上执行它：

```
python CountEvens.py
```

在此之前，也可以用 IDLE 中的 Run Module 菜单项运行同一个脚本，你就能马上发现在 Pi 这种资源有限的设备上使用 IDE 所带来的额外性能开销。本书后续的例子都将在命令行中直接运行，不过你仍然可以用 IDLE 作为你的代码编辑器。

进一步学习 Python

如果以前用过 Arduino，你会习惯把程序（在 Arduino 术语中称为 Sketch，但在 Python 中称为脚本）写成 **setup/loop** 结构，**setup** 只在程序启动时运行一次，而 **loop** 函数会一直被循环执行。下面的例子展示了如何在 Python 中实现这样的结构。在 IDLE 3 中



选择 New Windows 创建一个新窗口，输入下面的代码：

```
# Setup
n = 0
# Loop
while True:
    n = n + 1
    # The % is the modulo operator
    if ((n % 2) == 0):
        print(n)
```

执行 Run Module 并给你的脚本起一个名字（如 *EvenIntegers.py*）。当程序运行起来后，你可以看到屏幕上依次打印所有的偶数（按 Control-C 键中断程序，否则它会一直运行下去）。



在上面的代码示例中，每一层的缩进都用了 4 个空格，而不是 Tab 键（不过在 IDLE 中你也可以用 Tab 键缩进，它会自动帮你转换成多个空格）。在 Python 中，缩进是程序语意的一部分。这对于初学者来说，可能是一个学习的障碍；当复制 / 粘贴代码时，也可能会带来一些麻烦。但是，我们仍然认为严格的缩进要求使 Python 成为一种非常易读的语言。参考 Python Style Guidelines (<http://www.python.org/dev/peps/pep-0008/#indentation>) 学习如何编写可读性高的代码。

在 Python 程序中，空格非常重要，空格和缩进决定了程序的逻辑结构。在下面的例子中，`loop()` 函数下面同一缩进层次的代码被定义为这个函数的函数体。当代码的缩进层次提高一层（或到达文件结尾）时，就意味着循环的结束。这与 C 语言等以花括号或其他符号作为代码块分隔符的语言很不一样。



通过定义函数，可以把一个代码块组合成一个整体，在脚本中其他地方进行调用。我们可以用函数把上一个例子进行重写，如下（运行时，请把它保存为 *CountEvens.py*）：

```
# Declare global variables
n = 0❶

# Setup function
def setup():❷
    global n
    n = 100
def loop():❸
    global n
    n = n + 1
    if ((n % 2) == 0):
        print(n)

# Main❹
setup()
while True:
    loop()
```

在这个例子中，输出的值是从 102 开始的所有偶数。解释如下。

❶ 首先，变量 `n` 被定义成全局变量，所以它可在整个脚本的任意位置使用。

❷ 这里定义了 `setup()` 函数（但还没有执行它）。

❸ 类似地，这里定义了 `loop()` 函数。

❹ 在主程序段中，`setup()` 被调用了一次，然后循环调用 `loop()`。

每个函数第一行的 `global` 关键字非常重要，它告诉解释器，这个函数中将要使用全局变量 `n`，而不是在这个函数中创建一个只能在本函数中使用的新的局部变量（`local`，只对本函数有效）`n`。

受篇幅所限，我们的教程不可能写成一份完整的 Python 参考



手册, 如果要深入学习这门语言, 可以参考 *Think Python* (<http://shop.oreilly.com/product/0636920025696.do>) 和 *Python Pocket Reference* (<http://shop.oreilly.com/product/9780596158095.do>)。本章的后续内容会向你介绍编写与运行后续章节的示例程序中所需要用到的 Python 基础知识和一些有用的模块。第 4 章会介绍 Pygame 框架, 它是在 Pi 上编写多媒体应用程序的好帮手。

对象与模块

要理解本书后续的示例代码, 你需要理解有关对象和模块的基本语法。Python 是一种很纯净的语言, 只有 34 个保留的关键字(表 3.1)。这些关键字是语言的核心部件, 在编写脚本时, 通过使用它们来构造程序结构与流程。在 Python 中, 除了这些关键字以外的所有东西都可被看成是一个对象。所谓对象, 是指数据和与其相关操作的结合体, 并拥有一个名字。你可以修改一个对象的数据、获取它的相关信息或通过它操作其他对象。

表3.1 Python只有34个保留关键字

条件	循环	内置函数	类、模块与函数	错误处理
if	for	print	class	try
else	in	pass	def	except
elif	while	del	global	finally
not	break		lambda	raise
or	as		nonlocal	assert
and	continue		yield	with
is			import	
True			return	
False			from	
None				



在 Python 中，字符串、列表、函数、模块甚至数字都是对象。一个 Python 的对象可以被理解为是对一组属性和方法的封装，你可以通过点（.）语法来访问这些属性和方法。例如，在交互命令行中输入下面的代码创建一个 String（字符串）对象，然后调用它自身的一个方法把这个字符串转换为首字母大写的形式：

```
>>> myString = "quux"  
>>> myString.capitalize()  
'Quux'
```

也可以用 List（列表）对象的 `reverse()` 方法对列表中元素的顺序进行反转：

```
>>> myList = ['a', 'man', 'a', 'plan', 'a', 'canal']  
>>> myList.reverse()  
>>> print(myList)  
['canal', 'a', 'plan', 'a', 'man', 'a']
```



String 和 List 都是 Python 标准库中的内置模块，任何 Python 程序中都可以使用。String 和 List 模块各自定义了一系列的函数分别用于处理字符串和列表，包括我们上面演示过的 `capitalize()` 和 `reverse()`。

有一些标准库模块不是内置的，如果要使用它们，必须显式地使用 `import` 命令。例如，要使用标准库中的 `time`（时间）模块来进行时间相关的处理，需要：

```
import time
```

你也可以用 `import as` 在你的程序中加载一个模块并改变它的名字：



```
import time as myTime
```

还可以用 `from import` 来指定加载某个模块中的部分功能：

```
from time import clock
```

下面是一个简单的程序，使用 Python 脚本调用标准库中的 `time` 和 `datetime` 模块，实现每秒钟显示一次当前的时间：

```
from datetime import datetime
from time import sleep

while True:
    now = str(datetime.now())
    print(now)
    sleep(1)
```

`sleep` 函数可以让程序的运行暂停 1s。在运行这段程序时，你可能会发现每次打印的时间与实际的时间稍稍有些偏移。造成这个问题的原因如下。

- 这段代码没有考虑程序计算当前时间所花费的时间（所以暂停 `.9s`^①也许是一个更好的选择）。
- 其他的进程也在分享 CPU 资源，在你的程序执行时，CPU 的计算周期也可能被其他进程占用。这一点非常重要：在 Raspberry Pi 上编写程序时，你并没有工作在一个实时环境中。

在 Pi 上，`sleep()` 函数的精度在 5ms 以内。

下面，我们修改一下这个示例程序，引入一个文本文件并定期往里面写入一些日志数据。在处理文本文件时，所有的处理对象都是字符串。可以用 `str()` 函数把数字转换成字符串，也可以用

^① 在计算机语言中，常常可以省略掉整数位的 0，所以这里的 `.9` 就是 0.9。——译者注



`int()` 函数把字符串转换回整型数。

```
from datetime import datetime
from time import sleep
import random

log = open("log.txt", "w")

for i in range(5):
    now = str(datetime.now())
    # Generate some random data in the range 0-1024
    data = random.randint(0, 1024)

    log.write(now + " " + str(data) + "\n")
    print(".")
    sleep(.9)
log.flush()
log.close()
```



在一个实际的数据日志记录程序中，你需要保证你的 Raspberry Pi 已经按第 2 章所讲述的方法设置了正确的日期时间。

下面是另外一个例子 (*ReadFile.py*)，它从命令行参数读入一个文件名（用 `python3 ReadFile.py filename` 的形式在命令行中运行这个程序），然后程序打开这个文件，读出文件中的每一行并显示在屏幕上。在 Python 中，`print()` 函数与其他语言中的 `println()` 函数很相似，会在每次输出后自动添加一个换行。用 `print()` 函数的 `end` 参数可以改变这个添加换行的行为。

```
# Open and read a file from command line argument
import sys
```



```
if (len(sys.argv) != 2):  
    print("Usage: python ReadFile.py filename")  
    sys.exit()  
scriptname = sys.argv[0]  
filename = sys.argv[1]  
  
file = open(filename, "r")  
lines = file.readlines()  
file.close()  
  
for line in lines:  
    print(line, end='')
```

更多模块

Python之所以广受欢迎是因为它有大量由用户基于标准库开发的模块。Python包索引（Python Package Index, PyPI, <http://pypi.python.org/pypi>）是一个完整的包和模块的列表，其中的一部分在Raspberry Pi上尤其有用，如表3.2所示。后续你会陆续使用到这些模块，尤其是用于操作Raspberry Pi通用输入输出接口的GPIO模块。

表3.2 一些对Pi用户尤其有用的Python包

模块名	描述	URL	软件包名
RPi.GPIO	访问GPIO接口	http://code.google.com/p/raspberry-gpio-python/	<i>python-rpi.gpio</i>
Pygame	游戏开发框架	http://pygame.org	<i>python- pygame</i>
SimpleCV	简易的计算机视觉库	http://simplecv.org/	没有打包
Scipy	科学计算	http://www.scipy.org/	<i>python-scipy</i>
Numpy	Scipy库的数值基础库	http://numpy.scipy.org/	<i>python-numpy</i>
Flask	微型Web开发框架	http://flask.pocoo.org/	<i>python-flask</i>
Requests	友好的HTTP协议库	http://docs.python-requests.org	<i>python-requests</i>
PIL	图像处理	http://www.pythonware.com/products/pil/	<i>python-imaging</i>



续表3.2

模块名	描 述	URL	软件包名
wxPython	图形用户界面 (GUI) 框架	http://wxpython.org	<i>python-wxgtk2.8</i>
PySerial	串口通信	http://pyserial.sourceforge.net/	<i>python-serial</i>
pyUSB	FTDI-USB 接口	http://bleyer.org/pyusb	没有打包

要使用这些模块，你需要下载代码、配置包并安装。例如，PySerial 模块可以这样安装：

```
sudo apt-get install python-serial
```

如果一个包的开发者使用了标准的打包方式（通过使用 Python 的 distutils 工具），那么安装这个包只需下载代码、解压缩并执行下面的命令即可：

```
python setup.py install
```

你还可以学习一下 Pip package installer (<http://www.pip-installer.org>)，它使从 PyPI 上安装包的过程变得非常容易。

错误调试

你可能会遇到代码运行出错、需要调试的情况，这时 IDLE 的交互模式就很有用，它的 Debug 菜单中提供了一些有用的工具，帮助你了解程序的实际运行情况。你可以在程序运行过程中查看各变量的值或单步执行程序。

语法错误是最容易解决的问题，这类问题常常是输入错误或对语言语法的理解有偏差造成的。语义错误就会难发现得多，这种错



误表现为程序语法正确但运行结果不是我们所想要的。调试器可以在这种情况下大显身手。熟练地掌握各种调试技巧可能需要数年的积累，但以下内容可以帮助你调试在 Pi 上运行的 Python 程序中一些最常见的问题：

- 用 `print()` 打印信息表明程序曾经执行到这个位置。
- 用 `print()` 显示程序运行过程中变量的值。
- 仔细确认代码块缩进是否与你设想的逻辑一致。
- 如果是调试语法错误，要意识到真正出错的地方也许在解释器给你报错的位置的前面。
- 仔细检查所有的局部变量与全局变量。
- 检查开闭括号是否都有正确的匹配。
- 确认表达式中运算符的优先级是否正确，如果不太确定，就使用括号来明确优先级。例如 `3+4*2` 与 `(3+4)*2` 得到的是不同的结果。

进一步学习

有关 Python，还有很多的东西可以学习，以下资源也许会对你有所帮助。

Allen Downey 所著的 *Think Python*

(<http://shop.oreilly.com/product/0636920025696.do>)

这本书简明而又精炼地介绍了有关编程的入门知识，并正好使用了 Python 作为教学语言。

The Python Pocket Reference

(<http://shop.oreilly.com/product/9780596158095.do>)

有时候，完整地读完一本书会比在 Stack Overflow 网站上查阅一堆帖子更有价值。



Stack Overflow

(<http://stackoverflow.com/>)

Stack Overflow 是一个分享群体知识的网站，尤其当你在寻找一个特定的解决方案或分析某个特定的出错提示时，你遇到的问题常常也是别人曾经遇到过的问题。

Learn Python the Hard Way

(<http://learnpythonthehardway.org/>)

一本出色的书及在线资源。至少你应该先读一读它的序言 *The Hard Way Is Easier*。

Jason R. Briggs 所著的 *Python For Kids*

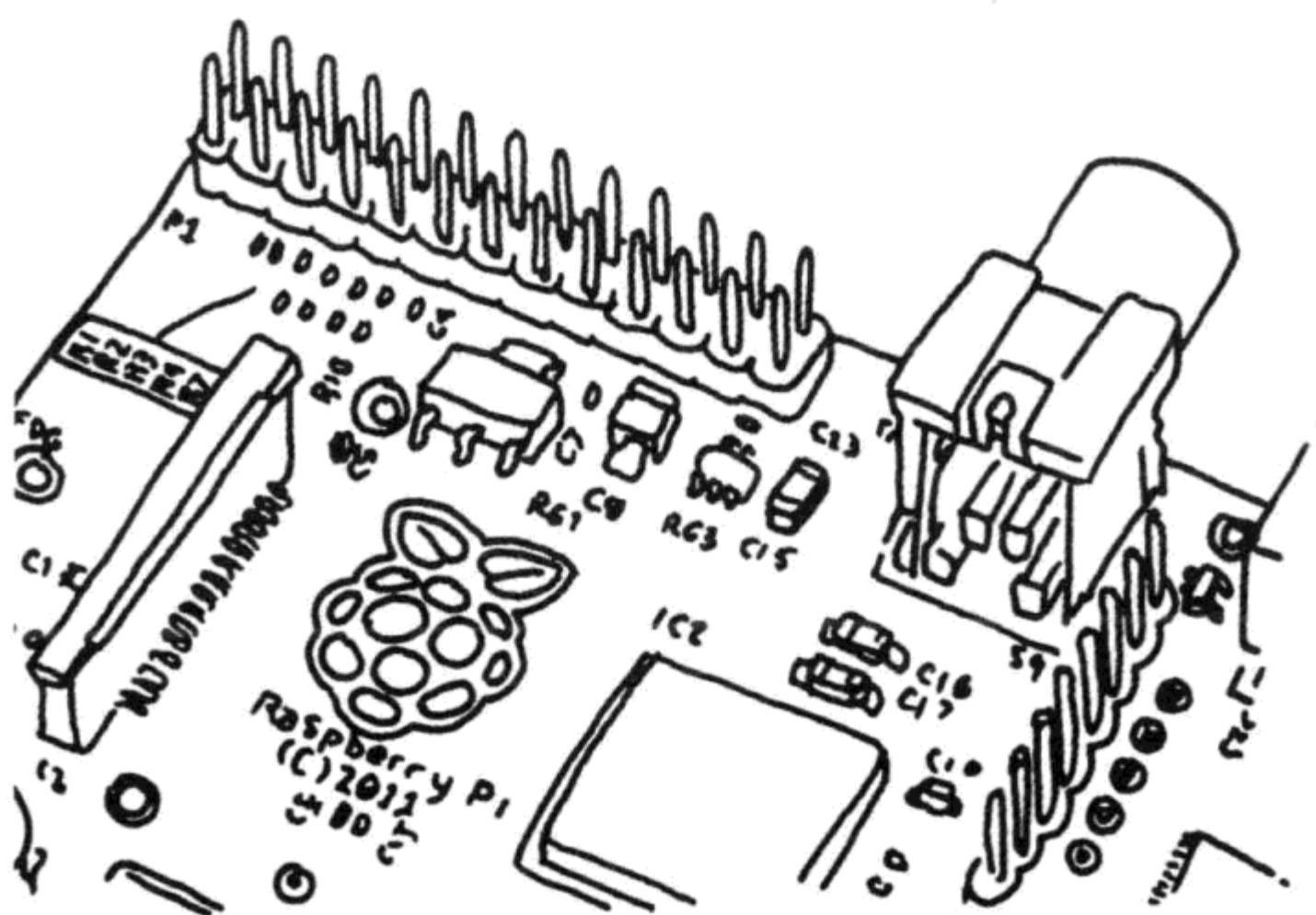
(<http://shop.oreilly.com/product/9781593274078.do>)

又一本介绍编程的入门书，正好用了 Python 作为教学语言，这本书适合年龄较小的读者。

第 4 章

用 Python 实现 动画与多媒体

Animation and Multimedia in Python





Pygame 是一个用于通过 Python 创建小游戏的轻量级框架。你也可以把它看作是一个用于多媒体程序设计的工具，通过它可以很方便地在屏幕上绘图、播放声音或处理键盘和鼠标事件。

Pygame 是对另外一个名为 Simple DirectMedia Library (SDL) 的库的封装，通过 SDL 库可以对 Pi 底层键盘、鼠标、音频驱动和视频驱动进行各种操作。Pygame 可以让 SDL 用起来更简便。

本章的重点是展示 Pygame 在多媒体编程方面的一些能力，而并非是游戏编程的一个教程。本章末尾列出了一些有关游戏编程的参考资料。

初识 Pygame

Raspberry Pi 上预装了 Pygame 库。虽然 Pygame 有支持 Python 3.0 的版本，但 Raspbian 中所提供的版本只能在 Python 2.7 中使用。因此，你可以使用非 3.0 版本的 IDLE (双击 IDLE 图标，不要双击 IDLE 3 图标)，也可以直接在命令行中用 2.7 版本的 Python 解释器来运行你的脚本。

下面的例子演示了一个最简单的 Pygame 程序，它在程序窗口中画了一个圆：

```
import pygame ❶  
  
width = 640 ❷
```



```
height = 480
radius = 100
fill = 1

pygame.init()③

window = pygame.display.set_mode((width, height))④
window.fill(pygame.Color(255, 255, 255))⑤
while True:⑥
    pygame.draw.circle(window,⑦
                       pygame.Color(255, 0, 0),
                       (width/2, height/2),
                       radius, fill)
    pygame.display.update()⑧
```

① 导入 Pygame 模块，使 Pygame 中的对象和函数在这个脚本中可用。

② 初始化一些全局变量。

③ 先调用一个 `init()` 函数，执行 Pygame 的一些初始化动作，使你可以正常使用 Pygame 模块。另外，在这个函数中也调用了 Pygame 中所有子模块的 `init()` 函数。

④ 创建一个窗口，准备用于在上面画图。这个窗口是一个 Pygame 的 `Surface` 类的对象。

⑤ 把窗口填充为白色。

⑥ 不断循环，你可以把每次循环所执行的代码看成是绘制了一个动画中的一帧。

⑦ 在窗口的中央画一个红色的圈。

⑧ 循环中的所有绘画命令其实都是在一个不可见的画布上执行的，当你画完了一帧后，调用 `display.update()` 把画好的图像显示在屏幕上。



在这些例子中，你需要按 Control-C 来中断程序的运行。如果你希望能够通过 Pygame 窗口上的关闭按钮来结束程序运行的话，可以在 `while` 循环的尾部添加下面的代码：



```
while True:  
    if pygame.QUIT in [e.type for e in  
        pygame.event.get()]:  
        break
```

这样，按下关闭按钮的产生事件就可以被正确捕捉并处理。有关事件的详细介绍，请参考“处理事件与输入”。

Pygame 中包含了很多子模块和对象，本章的后续章节中将陆续介绍与之相关的基础知识。

Pygame 的 Surface

Pygame 中的 Surface 可以被想象成一个矩形的图像，Surface 可以由多个图像帧组成，用于实现游戏或动画中一个场景。Surface 上的每一个像素由一组 8 位的 RGB 数字来表示，如 `(0, 255, 0)` 表示绿色。如果加上第 4 个数字，则可以用于表示透明度，如 `(0, 255, 0, 127)` 表示具有 50% 透明度的绿色。

图像显示窗口是一个作为背景的 Surface，可以在上面绘制其他的 Surface。通过 `pygame.display` 模块可以控制显示窗口或获取显示窗口的各种参数。`set_mode()` 函数用于创建一个新的图像显示窗口，`update()` 函数用于在绘制每一帧图像后刷新图像显示窗口。

使用 `pygame.image` 模块中的 `load()` 函数可以把一幅图像加载到 Surface 中并用于显示，使用 `blit()` 函数把加载后的图像与



创建好的 Surface 进行合并：

```
import pygame

pygame.init()
screen = pygame.display.set_mode((450, 450))
background = pygame.image.load("background.png") ❶
background.convert_alpha() ❷
screen.blit(background, (0, 0)) ❸
while True:
    pygame.display.update()
```

- ❶ 加载当前目录下的背景图片文件。
- ❷ `convert_alpha()` 函数用于把 Surface 的格式转换为与当前显示模式相匹配的格式。这不是一个必须进行的操作，但建议通过这样做来提高图像绘制的速度。
- ❸ 默认情况下 Surface 是纯黑色的，使用 `blit()` 函数把背景图片与黑色的 Surface 合并。

下面是一个把两张图片合并显示的例子（图 4.1）：

```
import pygame

pygame.init()
screen = pygame.display.set_mode((450, 450))
background = pygame.image.load("background.png").convert_alpha()
theremin = pygame.image.load("theremin.png").convert_alpha()
screen.blit(background, (0, 0))
screen.blit(theremin, (135, 50))
while True:
    pygame.display.update()
```

`pygame.transform` 模块提供了用于缩放和旋转 Surface 的函数。如果需要获取 Surface 上单个像素的颜色值，可以使用



pygame.surfarray 模块中的相关函数。

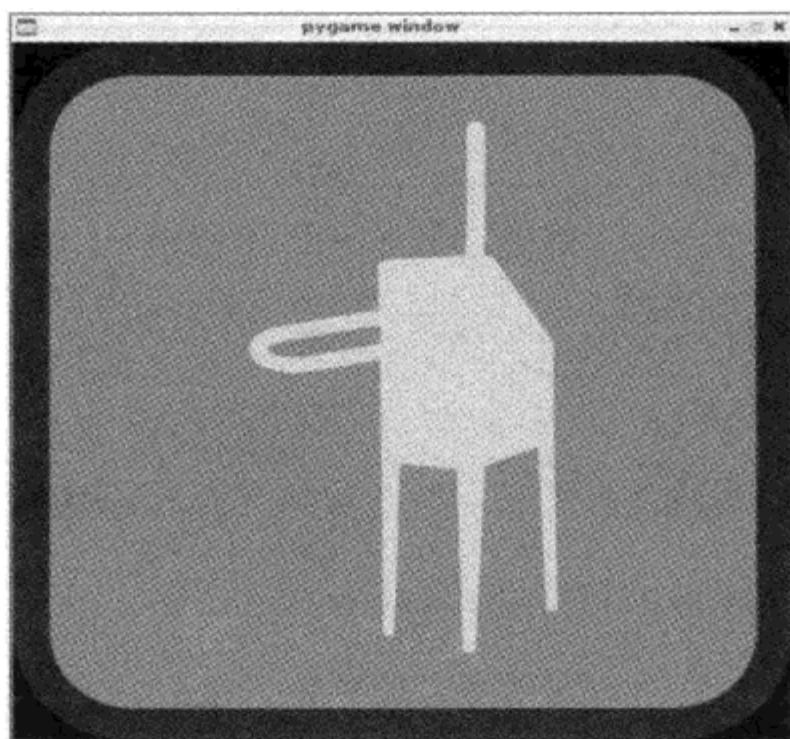


图4.1 把两张图片合并显示

新创建的 Surface 总是一个矩形的图片——尽管在上一个例子中通过使用图片的透明区域使 Surface 看上去是非矩形的，但 Surface 对象本身仍然是矩形的。如果想让 Surface 具有一个非矩形的边框（如用于进行像素级的碰撞检测），你可以使用 pygame.mask 模块通过创建另一个 Surface 对象设置一个蒙版。本章后面的有关游戏编程的参考资料中介绍了蒙版的具体使用方法。

在 Surface 上绘图

前面我们演示了用 Pygame 来绘制一个圆形，除此之外，pygame.draw 模块中还提供了绘制矩形、直线、弧线和椭圆形的功能。pygame.gfxdraw 模块则提供了另外一种绘制图形的方法，它提供了更多的绘制选项，但这个模块仍处于试验期，很多 API 以后还会发生变化。

如果要绘制一段文字，你需要先创建一个 Font 对象（由



pygame.font 模块提供），然后用它去加载字体文件并渲染文本。通过 pygame.font.get_fonts() 函数可以获取 Raspberry Pi 上可用的字体列表：

```
import pygame
pygame.init()
for fontname in pygame.font.get_fonts():
    print fontname
```

可以看到，Raspberry Pi 上默认预装了少量的字体。下面的代码通过使用 SysFont 对象加载 freeserif 字体渲染了一段文本：

```
import pygame
pygame.init()
screen = pygame.display.set_mode((725, 92))
font = pygame.font.SysFont("freeserif", 72, bold = 1)
textSurface = font.render("1 Theremin Per Child!", 1,
                          pygame.Color(255, 255, 255))
screen.blit(textSurface, (10, 10))
while True:
    pygame.display.update()
```



如果你需要使用更多的字体，可以用下面的命令安装：

```
sudo apt-get install ttf-mscorefonts-installer
sudo apt-get install ttf-liberation
```

处理事件与输入

在 Pygame 中，用户所触发的事件（如按下键盘上的某个键、移动或点击鼠标）会被捕获并生成 Event 对象放入消息队列，等待程序去做相应的处理。pygame.event 模块提供了从消息队列中获取未处理的消息并对之进行处理的功能。你还可以通过创建自己的消



息类型来实现一个消息系统。下面的例子中，我们使用消息队列来扩展前面的画圆的程序：在每一帧中，根据鼠标箭头所在的位置重新绘制一个圆形，鼠标箭头离窗口边框越近则绘制的圆的半径越大（图 4.2）。

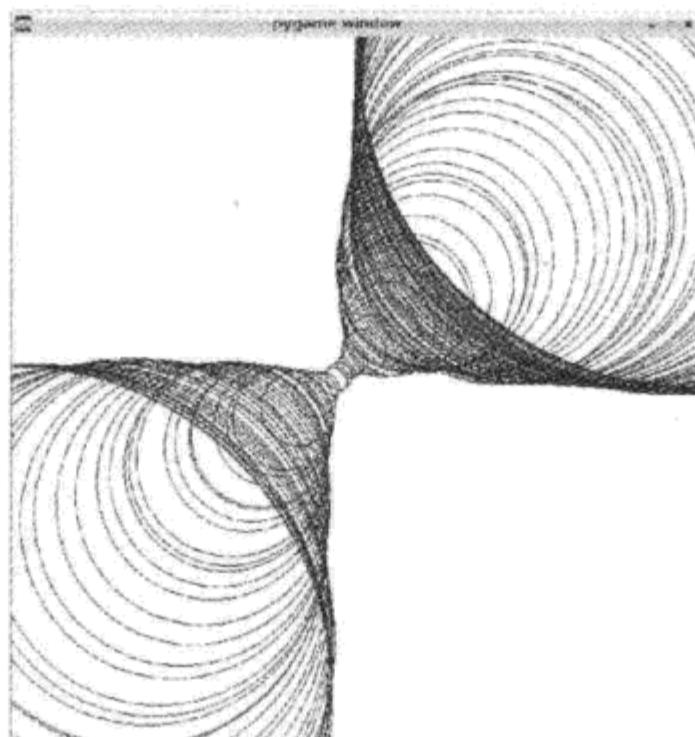


图4.2 Pygame消息处理例子的输出

```
import pygame
from pygame.locals import *

width, height = 640, 640
radius = 0
mouseX, mouseY = 0, 0

pygame.init()
window = pygame.display.set_mode((width, height))
window.fill(pygame.Color(255, 255, 255))

fps = pygame.time.Clock()

while True:
    for event in pygame.event.get():
        if event.type == MOUSEMOTION:
            mouseX, mouseY = event.pos
```



```
if event.type == MOUSEBUTTONDOWN: ⑦
    window.fill(pygame.Color(255, 255, 255))
    radius = (abs(width/2 - mouseX)+abs(height/2 - mouseY))/2 + 1 ⑧
    pygame.draw.circle(window, ⑨
                        pygame.Color(255, 0, 0),
                        (mouseX, mouseY),
                        radius, 1)
    pygame.display.update()
    fps.tick(30)⑩
```

① `pygame.locals` 模块中定义了包括 `MOUSEMOTION` 在内的一些常量。以这样的形式导入这个模块后，后续我们需要使用里面定义的常量时就不需要再添加 `pygame.` 前缀。

② 存放鼠标坐标的变量。

③ 这个函数初始化并返回我们用于作为帧计数器的对象。通过使用 `fps` 变量（每秒的帧数，Frames per Second）你可以在每一帧图像显示完成后暂停一段时间，以便获取一个稳定的帧速。

④ 一直循环，每循环一次就生成新的一帧。

⑤ 循环处理消息队列，每循环一次，`event` 变量中会获得消息队列中的下一个消息。

⑥ 如果获得的消息是一个鼠标移动消息，更新变量值，记录下鼠标的位置。

⑦ 如果获得的消息是一个鼠标点击消息，清除窗口中的显示。

⑧ 根据鼠标距离窗口中心的位置，计算出要绘制的圆形的半径。

⑨ 画圆。

⑩ 暂停一段时间，保证帧速为 30fps。

以下模块在处理消息和用户输入时也很有用。

`pygame.time`

监测时间的模块。



pygame.mouse

获取鼠标信息的模块。

pygame.key

获取键盘信息的模块，里面提供了很多常量用于表示每一个键。

pygame.joystick

操作游戏操纵杆的模块。

全屏模式

如果想要让你的程序以全屏模式运行，可以在设置显示模式时加上 `pygame.FULLSCREEN` 选项。在全屏模式下，不能使用 **Control-C** 键中断脚本的运行，所以你需要保证在程序中提供了某种退出程序的方式。

```
import pygame
import random
from time import sleep

running = True
pygame.init()
screen=pygame.display.set_mode((0,0), pygame.FULLSCREEN)
while running:
    pygame.draw.circle(
        screen,
        pygame.Color(int(random.random()*255),
                     int(random.random()*255),
                     int(random.random()*255)),
        (int(random.random()*1500),
         int(random.random()*1500)),
        int(random.random()*500), 0)
    pygame.display.update()
    sleep(.1)
```



```
for event in pygame.event.get():
    if event.type == pygame.KEYDOWN:
        running = False
pygame.quit()
```

在这个脚本运行时，按键盘上的任意键就可以退出程序。

Sprite

在一个游戏中，可以把各种可移动、可操控的图形元素都当作 `Sprite` 对象来处理。`pygame.sprite` 模块提供了在屏幕上绘制 `Sprite` 和在多个 `Sprite` 之间进行碰撞检测的功能。多个 `Sprite` 可以被组合在一起，同时控制或更新。通过使用 `Sprite` 编写一个完整的游戏的方法已经超出本书的内容范围，可以参考“进一步学习”中的参考资料了解更多的细节。

当需要创建多个屏幕元素并且这些元素共享很多相似的代码时，你就可以考虑使用 `Sprite`。下面的代码演示了如何创建和更新多个 `Sprite`，程序的运行结果是在屏幕上绘制两个小球，这两个小球在运动到屏幕边缘时会反弹回来。如果要增加小球的数量，可以创建更多的 `Sprite` 对象并对其起始坐标、方向和速度进行不同的初始化。

```
import pygame

class Ball(pygame.sprite.Sprite):❶
    def __init__(self, x, y, xdir, ydir, speed):❷
        pygame.sprite.Sprite.__init__(self)
        self.image = pygame.Surface([20, 20])
        self.image.fill(pygame.Color(255, 255, 255))
        pygame.draw.circle(self.image,
                           pygame.Color(255, 0, 0),
                           (10, 10), 10, 0)
        self.rect = self.image.get_rect()
```



```
    self.x, self.y = x, y③
    self.xdir, self.ydir = xdir, ydir
    self.speed = speed

    def update(self):④
        self.x = self.x + (self.xdir * self.speed)
        self.y = self.y + (self.ydir * self.speed)
        if (self.x < 10) | (self.x > 490):
            self.xdir = self.xdir * -1
        if (self.y < 10) | (self.y > 490):
            self.ydir = self.ydir * -1
        self.rect.center = (self.x, self.y)

    pygame.init()
    fps = pygame.time.Clock()
    window = pygame.display.set_mode((500, 500))
    ball = Ball(100, 250, 1, 1, 5)⑤
    ball2 = Ball(400, 10, -1, -1, 8)

    while True:
        ball.update()⑥
        ball2.update()
        window.fill(pygame.Color(255, 255, 255))
        window.blit(ball.image, ball.rect)⑦
        window.blit(ball2.image, ball2.rect)
        pygame.display.update()
        fps.tick(30)
```

① `class` 关键字用于基于 `Sprite` 对象^① 创建一个新的 `Ball` 对象，你可以为这个新的对象定义绘制这个对象的方法以及更新这个对象时所需要的操作。

② 当用 `Ball()` 创建一个新的对象时，这个函数会被调用。
③ 这些变量会作为 `Ball` 实例的一部分，每一个 `Ball` 对象会保存一份属于它自己的变量值。

^① 严格意义上来说，这里的“对象”应该是“类”。在面向对象程序设计中，“类”表示对一类相似概念的抽象定义，“对象”是基于“类”生成的实体。——译者注



④ `update()` 函数会在绘制每一帧画面时被调用，对象所在的位置会根据对象中记录的方向和速度做相应的变化。同时，还会检测是否遇到了窗口的边缘：如果遇到边缘，则把对应坐标轴上的方向进行颠倒。

⑤ 以不同的起始位置、运动方向和运动速度创建两个 `Ball` 对象。

⑥ 根据当前的位置、运动方向和速度，更新小球的位置。

⑦ 在当前位置绘制小球。

播放声音

在 Pygame 中，你可以使用 `pygame.mixer` 模块加载声音文件并进行播放，也可以用 `pygame.midi` 模块向 Pi 上运行的 MIDI 程序或 USB 接口上连接的 MIDI 设备发送 MIDI 消息。下面的例子演示了如何播放一个 WAV 格式的音频文件，这个文件你可以从网上下载 (<http://archive.org/details/WilhelmScreamSample>)。

```
import pygame.mixer
from time import sleep

pygame.mixer.init(48000, -16, 1, 1024)

sound = pygame.mixer.Sound("WilhelmScream.wav")
channelA = pygame.mixer.Channel(1)
channelA.play(sound)
sleep(2.0)
```

这个程序加载了音频文件 (WAV 格式) 并与一个通道关联，然后通过混音器在独立的进程^①中播放每一个通道中的音频，因此，在同一时间可以同时播放多个声音。程序最后的 `sleep()` 函数调用是为了保证主程序在声音播放完毕后才退出。

① 确切地说应该是线程。——译者注



在第 8 章的发音板程序中，我们还会进一步介绍通过混音器播放音频文件的知识。Raspberry Pi 的 MIDI 功能同样让人着迷，你可以很容易地创建一个 MIDI 控制器程序。截至本书写作时，Raspberry Pi 上的软音源还比较粗糙，模拟音频输出效果也远不如 HDMI 输出，不过你还是可以通过使用一个 USB 接口的 MIDI 设备来获得较好的效果。下面的代码可以列出 Raspberry Pi 上连接的所有 MIDI 设备：

```
import pygame
import pygame.midi ❶

pygame.init()
pygame.midi.init() ❷
for id in range(pygame.midi.get_count()): ❸
    print pygame.midi.get_device_info(id) ❹
```

- ❶ midi 模块在导入 pygame 模块时不会自动导入。
- ❷ 同样地，midi 模块还需要单独进行初始化。
- ❸ get_count() 可以用于获取 Raspberry Pi 上所有 MIDI 设备的数目，包括 USB 设备、软音源或其他虚拟 MIDI 设备。
- ❹ 显示该设备相关的信息。

如果你在 Raspberry Pi 上连接了一个 MIDI 键盘，你会从这个程序中得到以下输出：

```
('ALSA', 'Midi Through Port-0', 0, 1, 0)
('ALSA', 'Midi Through Port-0', 1, 0, 0)
('ALSA', 'USB Uno MIDI Interface MIDI 1', 0, 1, 0)
('ALSA', 'USB Uno MIDI Interface MIDI 1', 1, 0, 0)
```

第 1 列的信息告诉你当前系统使用的是 ALSA (Advanced Linux Sound Architecture，高级 Linux 音频架构) 音频系统，第 2 列



是对每一个 MIDI 设备的描述。最后 3 列数字表示这个端口是输入设备还是输出设备，以及这个设备当前是打开还是关闭的。在这个例子中，你可以认为是有 4 个不同的 MIDI 设备，端口号分别为 0~3。

MIDI Through Port-0 (0, 1, 0)

0 号端口，输出端口，用于向 Pi 上所运行的某一个软音源设备输出 MIDI 消息。

MIDI Through Port-1 (1, 0, 0)

1 号端口，输入端口，用于获取 Pi 上所运行的某一个 MIDI 控制程序发出的控制信息。

USB Uno MIDI Interface MIDI 1 (0, 1, 0)

2 号端口，输出端口，连接到 USB 口上的一个 MIDI 设备，如 MIDI 键盘。

USB Uno MIDI Interface MIDI 1 (1, 0, 0)

3 号端口，输入端口，连接到 USB 口上的一个 MIDI 设备，如 MIDI 键盘。

有了这些基础知识，你就可以在 USB 接口连接一个 MIDI 键盘并使用 Pygame 来控制它了，示例代码如下：

```
import pygame
import pygame.midi
from time import sleep

instrument = 0❶
note = 74
volume = 127

pygame.init()
pygame.midi.init()
```



```
port = 2❶
midiOutput = pygame.midi.Output(port, 0)
midiOutput.set_instrument(instrument)
for note in range(0, 127):
    midiOutput.note_on(note, volume)❷
    sleep(.25)
    midiOutput.note_off(note, volume)
del midiOutput
pygame.midi.quit()
```

- ❶ 这些是 MIDI 消息中使用的编号，通常在 0~127 取值。
 - ❷ 打开 2 号端口，这个端口即 USB 接口上所接的 MIDI 键盘的输出端口。
 - ❸ 发送一个音符开始消息，暂停，发送一个音符结束消息。
- 由此可见，Raspberry Pi 具有很大的潜力可以成为一个音乐创作平台。

播放视频

使用 `pygame.movie` 模块可以用于播放视频，它所播放的视频文件必须使用 MPEG1 格式编码。如果你的视频文件是以其他格式编码的，可以使用 `ffmpeg` 工具对它进行格式转换（先用 `sudo apt-get install ffmpeg` 安装这个工具）。要回放视频，只需创建一个新的 `Movie` 对象并调用 `play()` 函数：

```
import pygame
from time import sleep
pygame.init()

screen = pygame.display.set_mode((320,240))
movie = pygame.movie.Movie("foo.mpg")
movie.play()
while True:
```



```
if not(movie.get_busy()):  
    print("rewind")  
    movie.rewind()  
movie.play()  
if pygame.QUIT in [e.type for e in pygame.event.get()):  
    break
```

如果所播放的视频包含声音，则需要在播放之前关闭 Pygame 的混音器。方法是在开始播放视频前调用：

```
pygame.mixer.quit()
```

更多示例



`pygame.examples` 模块中提供了很多完整的示例程序。可以在 `/usr/share/pyshared/pygame/examples` 目录中找到这些程序的源代码。

进一步学习

Pygame 官方文档

(<http://www.pygame.org/docs/>)

Pygame 的官方文档比较散乱，不过希望你在学习完本章内容后，能从该文档中找到你所需的信息。

Al Sweigart 所著的 *Making Games with Python & Pygame* 和 *Invent Your Own Computer Games with Python*

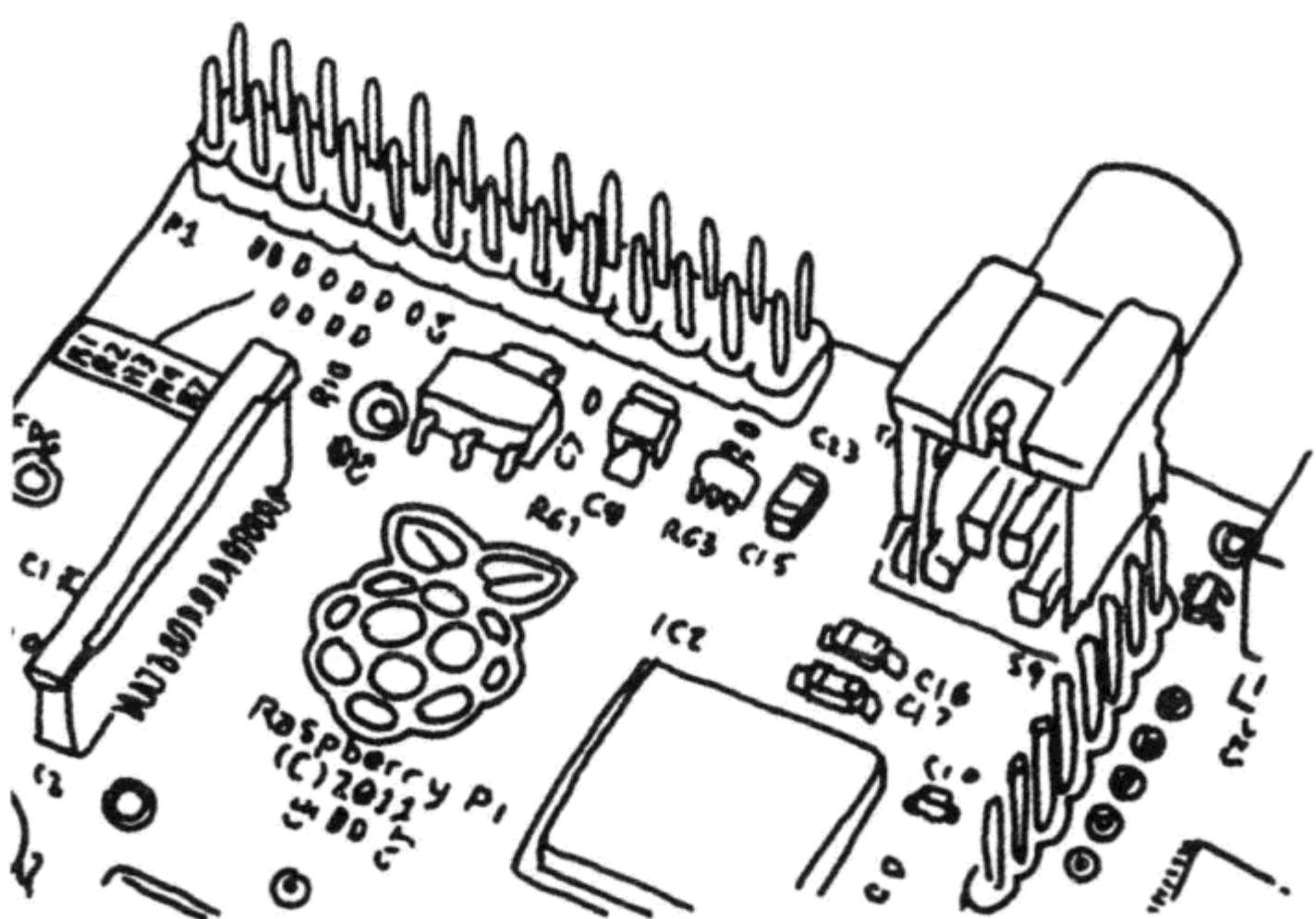
(<http://inventwithpython.com/>)

两本都是以知识共享许可协议 (Creative Commons) 授权的书，这两本书中开发的游戏都已经预装在了 Raspberry Pi 中。

第 5 章

Pi上的Scratch

Scratch on the Pi





Scratch 由 MIT (麻省理工学院) 媒体实验室的 Lifelong Kindergarten 小组开发，它采用一种创新的方式在青少年中开展程序设计教学。Scratch 程序由一系列彩色的代码积木组成，每块代码积木可以用于完成一项操作。用积木这种可视化的方式来编程，可以让一些初学者避免在初次接触到文本编程语言时产生困惑。

如果只是把 Scratch 看成一种简易的彩色编程语言难免有失偏颇，它其实提供了一个友好的开发环境，让开发者能更为便捷地完成一项任务。在程序运行过程中，当前运行到的代码积木会变得高亮，并且所有的代码积木都可以随时更换并实时看到更换后的效果，这对于初学者来说很有帮助。

编写 Scratch 程序的基本形式是操作角色在一个舞台上完成一系列动作。Scratch 有一个很大的用户社区，并且 Scratch 平台本身就提供了在社区中分享角色和代码的功能。

初识 Scratch

用 Scratch 编程非常容易，我们以一个名为“盒子里的猫”的程序作为开始。Scratch^①的主界面中包含了多个操作区域，图 5.1 中标明了每个区域的功能。你可以双击桌面上的 Scratch 图标，也

① 点击 Scratch 主界面左上角的“设定语言” (Select Language) 地球图标可以选择 Scratch 界面上显示所用的语言。先点击地球图标，在下拉菜单中点击 More，再选择“简体中文”就可以把界面语言改成中文。如果先前已经把系统区域设置设为 zh_CN.UTF-8，则启动 Scratch 后会自动切换到中文界面。——译者注



可以点击屏幕左下角的桌面菜单并选择 Education → Scratch 来启动 Scratch。

Scratch 的程序由舞台上的角色组成，角色的行为由脚本控制，而脚本是由积木盒中的一块块积木组成的。在新建一个 Scratch 程序时，你会默认得到一只小猫作为你的角色。如果你不喜欢这只猫，可以在角色列表区域中的小猫角色上点击鼠标右键，并在菜单中选择“删除”（Delete）来删除它，然后再自己画一个喜欢的角色或点击角色列表区域上部的“来个令人惊喜的角色吧”（Get Surprise Sprite）按钮从 Scratch 社区设计的其他角色形象中随机选择一个角色。

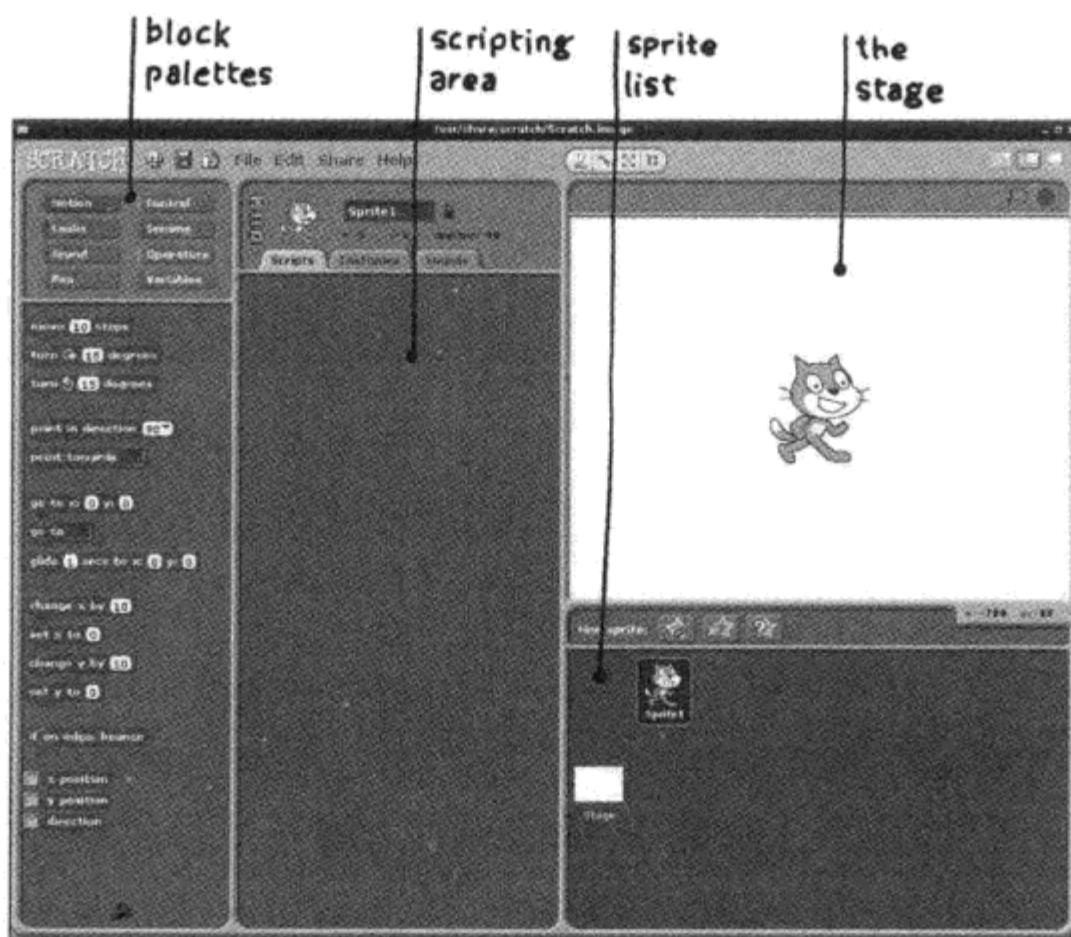


图5.1 Scratch的界面，程序的窗口被划分为多个区域，分别是积木盒、脚本与造型区、角色列表和舞台

脚本区中显示的是当前选中角色的控制脚本。在角色列表中点击猫的图标，由于这时还没有为这个角色编写脚本，所以你会看到一个空白的脚本区，可以在脚本区中的任意位置开始构建脚本。如



果你新添加了一个角色，也可以通过点击这个角色在脚本区中显示出与它相关的脚本。

通过把积木盒中的积木块拖动到脚本区，就可以来构建脚本。有些积木内部还可以包含其他积木——这从积木的形状上就可以看得出来。Scratch 中有 3 种类型的积木，如图 5.2 所示。



图5.2 Scratch中的3种积木：帽子型（左）、可堆叠型（中）和报告型（右）

帽子型积木

“当绿色小旗被点击”（when green flag clicked）积木就是帽子型积木的一种，它应该被安置在其他积木的最上方，等待相应事件的发生。

可堆叠型积木

上部有个凹槽、下部有个突起的积木块属于可堆叠型积木，它们可以被层叠组合在一起。所有的可堆叠型积木会按从上到下的顺序执行。

报告型积木

具有圆角或箭头型边框的积木属于报告型积木，它们可以安装在其他积木内部。报告型积木可以是一个变量，也可以用来获取鼠标坐标或其他信息。



要开始构建一个新的脚本，点击“控制”（Control）积木盒，把图 5.3 所示的积木拖到脚本区。

下一步是添加一个“重复执行”（forever）的可堆叠型积木，这块积木会不断地循环执行它内部的所有脚本，直到脚本的运行被中止为止，如图 5.4 所示。

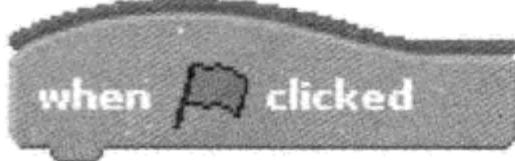


图5.3

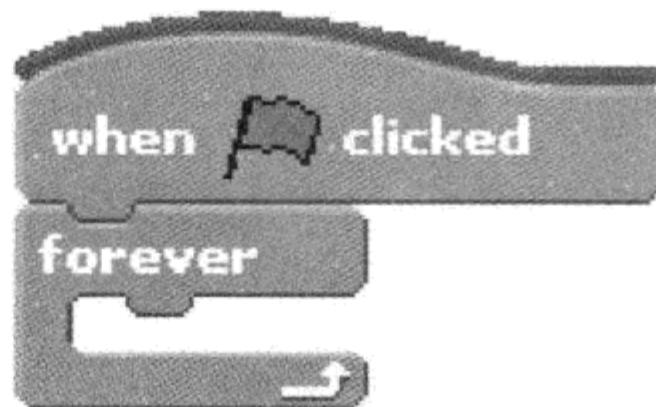


图5.4

你需要通过不断地练习才能熟悉各种积木的功能。当你选中一块积木时，也就同时选中了与它相接的后续所有积木。所以，要移除积木堆中的一块积木，你得先把这块积木后续的积木堆移开。你也可以在积木上点击鼠标右键来复制它们或获取相关的帮助信息。

再下一步，选择“动作”（Motion）积木盒，拖动一块“旋转”（turn）积木，并放置在“重复执行”（forever）积木中，如图 5.5 所示。

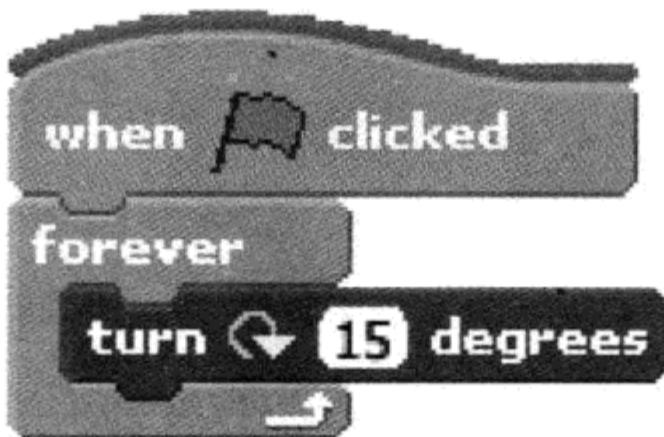


图5.5



现在，你可以点击舞台右上角的绿色旗子图标来运行这个脚本了。Scratch有一个优点——在脚本的执行过程中你也可以对脚本做出修改，并且这些修改会立即在舞台上展现出来。这对于调试脚本而言十分方便。



当你点击绿色旗子图标时，Scratch会向你的项目中的所有脚本发送开始运行的信号（如果点击的是停止图标，则会发送停止信号）。

如果你注意观察“旋转”（turn）积木，你会发现旋转角度的数值是在一个圆角矩形框中显示的，你可以直接修改这个数值，也可以用一块形状相同的积木来替换这个数字。下面我们来做一点修改，让角色随机运动起来。

首先，从“数字和逻辑运算”（Operators）积木盒中找出“随机选一个数”（pick random）积木，用它替换掉“旋转”（turn）积木中的默认值。替换后，修改“随机选一个数”（pick random）积木中的值，使它可以产生一个-10 ~ 10的随机数，如图5.6所示。

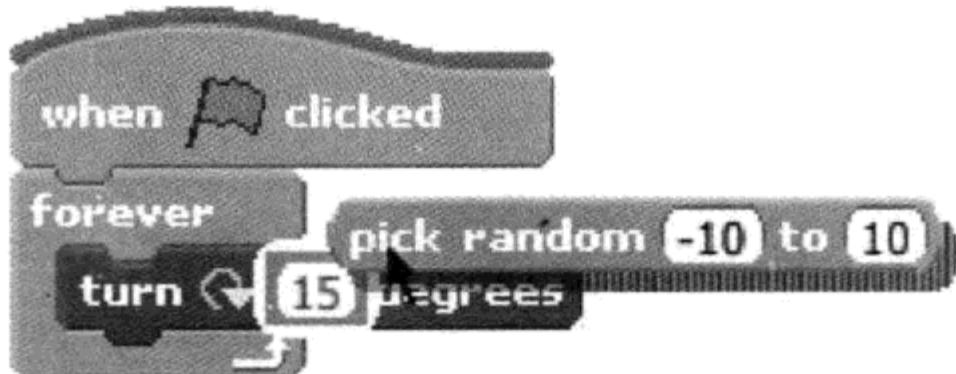


图5.6

从“动作”（Motion）积木盒中找出“移动”（move）积木，这个积木会控制角色以当前的角度向前移动指定个数像素的距离。



当前的角度会在脚本区上面的蓝色区域中以数字形式显示出来。当你把这块积木放置到你的脚本中，你马上会看到角色开始了随机移动，如图 5.7 所示。

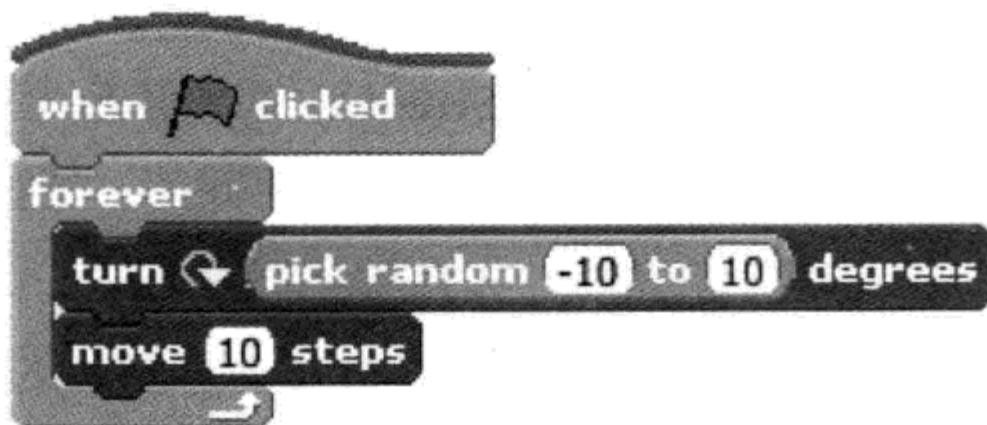


图5.7

最后，为了不让角色跑出舞台边框，添加一块“动作”（Motion）积木盒中的“碰到边缘就反弹”（if on edge, bounce）积木，如图 5.8 所示。

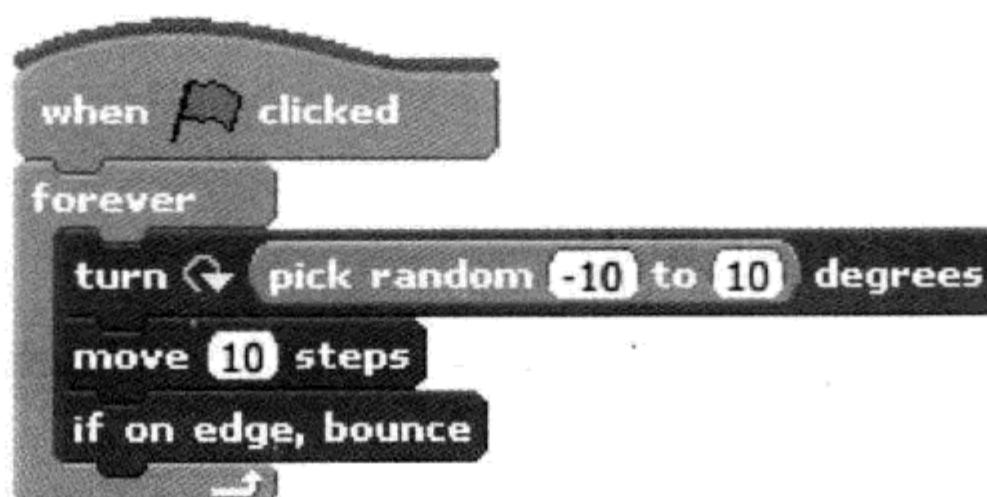


图5.8

以上就是编写 Scratch 脚本所需的全部基础知识！

舞 台

舞台是指界面右上角的区域，程序中的角色在舞台区域完成脚



本中所指定的动作和交互。与角色一样，也可以用脚本来改变舞台的外观和行为。可以通过脚本区的“多个背景”（Backgrounds）页来绘制舞台的背景图片。

Scratch 中使用的坐标系统与其他常见的多媒体环境（如 Pygame）中的坐标系统不一样，Scratch 中的坐标系统采用了数学

中常用的方式，即坐标 (0,0) 位置对应舞台的中心位置。如图 5.9 所示，舞台的可视区域是 (-240,180) 和 (240,180) 为对角顶点的矩形区域。你可以用程序控制角色移动到舞台的可视区域以外的位置。当选中一个角色时，可以在它的脚本区域顶部看到它所处的当前位置，

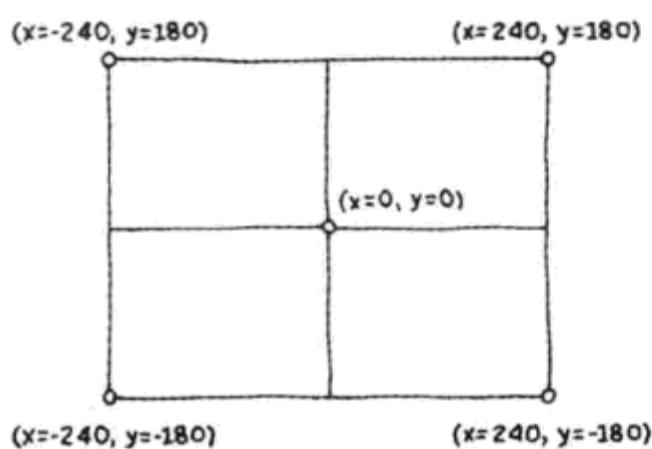


图5.9 Scratch的坐标系统，原点设定在舞台的正中央

在舞台的底部可以看到当前的光标位置。

有关角色的两点知识

在脚本区，还有两个页面可以使用：造型（Costumes）和声音（Sound）。

一个角色的造型包含了多幅图像，这些图像可以用于在制作这个角色相关的动画时切换使用，用于表现一个角色的不同状态（如一艘爆炸的飞船）。例如，创建一个新的角色，它的图片是一只睁开的眼睛，我们可以通过创建造型来让这只眼睛眨一下。点击“绘制新角色”（Paint New Sprite）按钮——状态区中“新增角色”（New Sprint）右侧的图标，并按图 5.10 绘制，完成后点击“确认”（OK）。

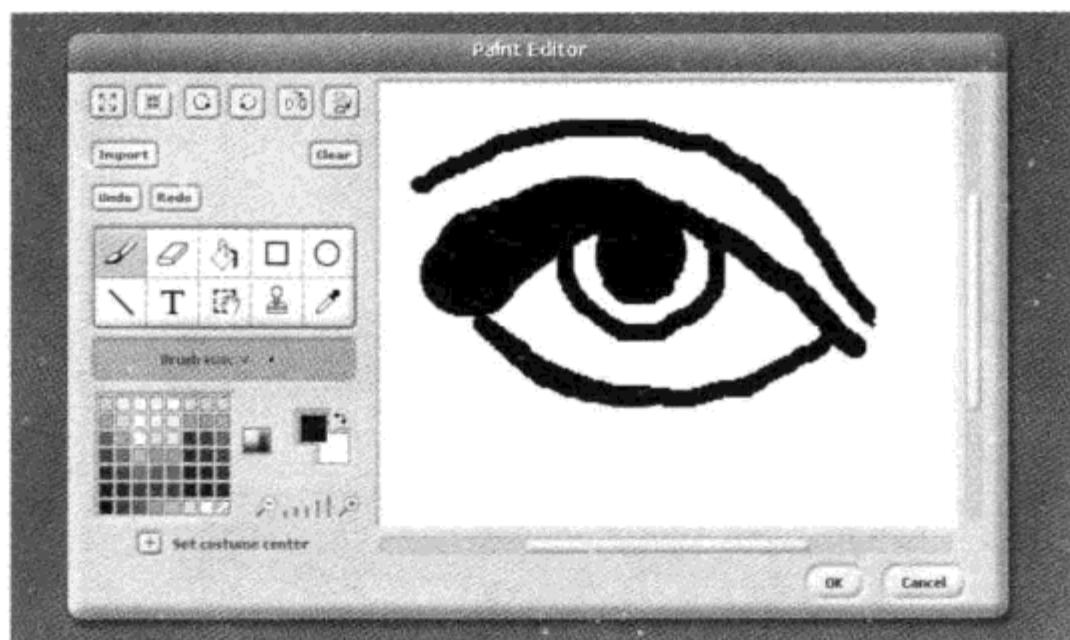


图5.10

选择“造型”（Costumes）页（这里会显示出你刚才绘制的图片），把这个造型的名字改为open。点击“复制”（Copy）按钮创建一个新的造型，点击“编辑”（Edit），擦除眼珠并重新画上一个闭上的眼皮，然后点击“确认”（OK）并把这个新的造型名字改为shut，如图 5.11 所示。

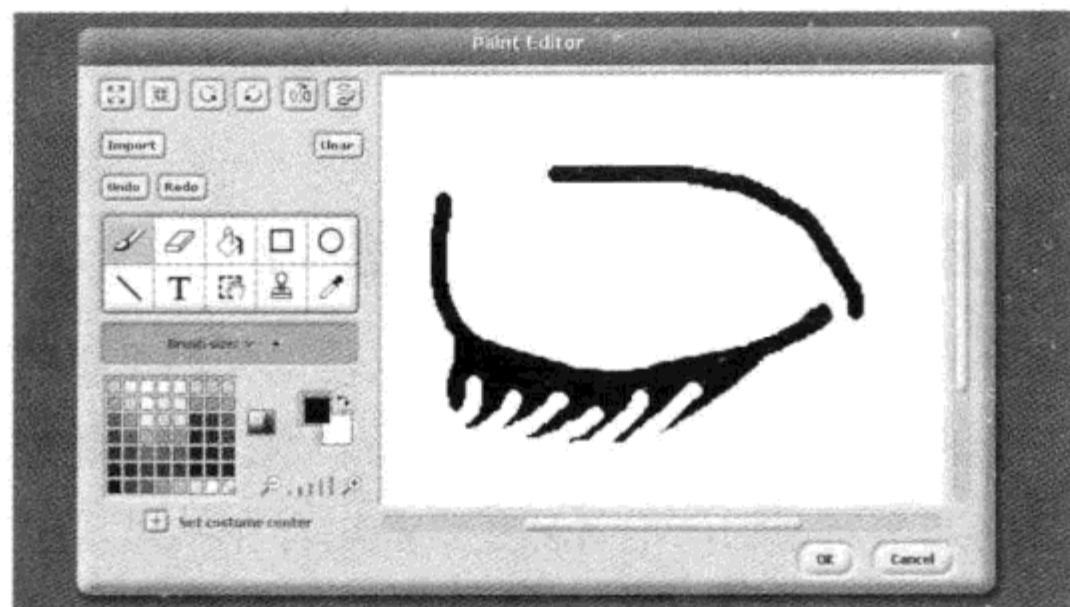


图5.11

然后，选择“脚本”（Script）页面，并把积木拖到脚本区组成图 5.12 所示的脚本，这个脚本会控制眼睛每秒钟眨一次。

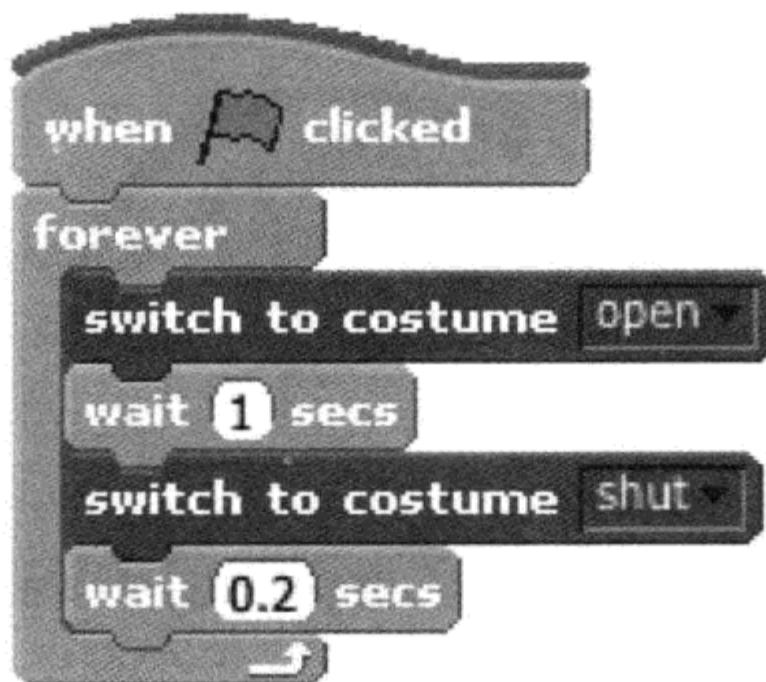


图5.12

最后，你可以用“声音”（Sound）标签为角色添加声音效果。点击“录音”按钮就可以打开Scratch内置的录音机（Sound Recorder）程序。你需要在Raspberry Pi使用外置USB声卡和麦克风才可以录音。为这个声音起一个名字，然后就可以通过“播放声音”（play sound）或“播放声音直到播放完毕”（play sound until done）积木来播放这段声音。

更复杂的例子：星际入侵者游戏

在星际入侵者游戏中，玩家需要对入侵的外星飞船射击，如果不能成功地阻挡外星飞船撞击到玩家的加农炮，游戏就结束了。我们会在这个游戏中展现如何把各种元素都组合在一起。

先点击“文件（File）→新建（New）”，然后点击“文件（File）→另存为（Save As）”保存文件。下面可以开始创建角色。先按住Control键点击或右键点击角色列表中的小猫角色，从菜单选择“删除”（Delete）把它删掉，然后创建5个角色，你可以自己画这些角色的图片，也可以加载现成的图片文件。我画了如图5.13的5个



角色，并把它们扫描成 PNG 图片。

创建完角色后，它们会被添加到舞台上。如果它们的大小不是很合适，可以在舞台顶部的工具栏中选择“放大角色”（Grow Sprit）或“缩小角色”（Shrink Sprite）来调整它们的大小。你还需要对这些角色进行重命名，可以按图 5.13 中的名字来给它们命名。

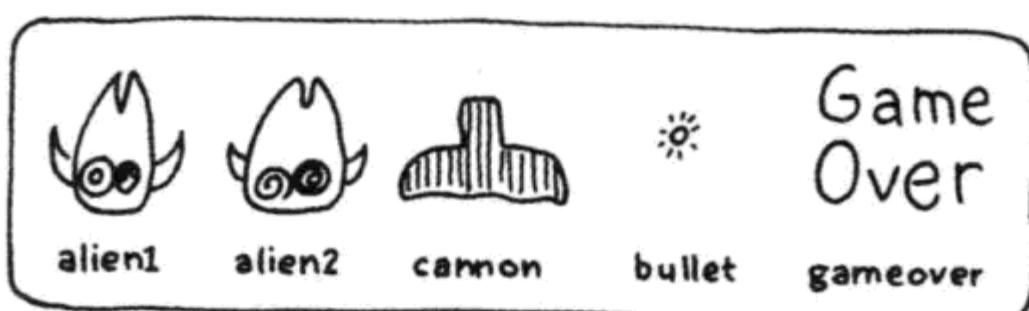


图5.13 星际入侵者中的5个角色

外星飞船和加农炮都有两种状态：正常状态和爆炸状态。因此，你需要为这几个角色添加额外的造型，如图 5.14 所示。

需要为每个角色分别编写脚本，用来控制它们在游戏中的行为。为了清晰起见，每一个动作都需要定义一个独立的脚本，这些脚本会被同时运行。

附录 B 中列出了为每个角色定义的完整脚本。



图5.14 为每个角色创建两个造型：一个用于正常状态，一个用于爆炸状态。按本图中的名字给这些造型命名



我们从第一个外星飞船开始，选中这个角色，然后把图 5.15 所示的积木拖入脚本区。



图5.15

这段脚本把相关角色移动到它的起始位置，确保它是可见的，并且处于没有爆炸的模式。

然后，拖动图 5.16 所示的积木并在脚本区中把它们组合好。

点击绿色小旗后你会发现，现在角色的行为与我们在本章开始时所创建的“盒子里的猫”的程序相当一致。由于现在舞台上有 3 个角色共存，所以你需要处理角色之间发生碰撞时的行为。当两艘外星飞船即将发生碰撞时，可以让它们各自旋转 90°（图 5.17），就可以避免两个角色重叠在一起了。

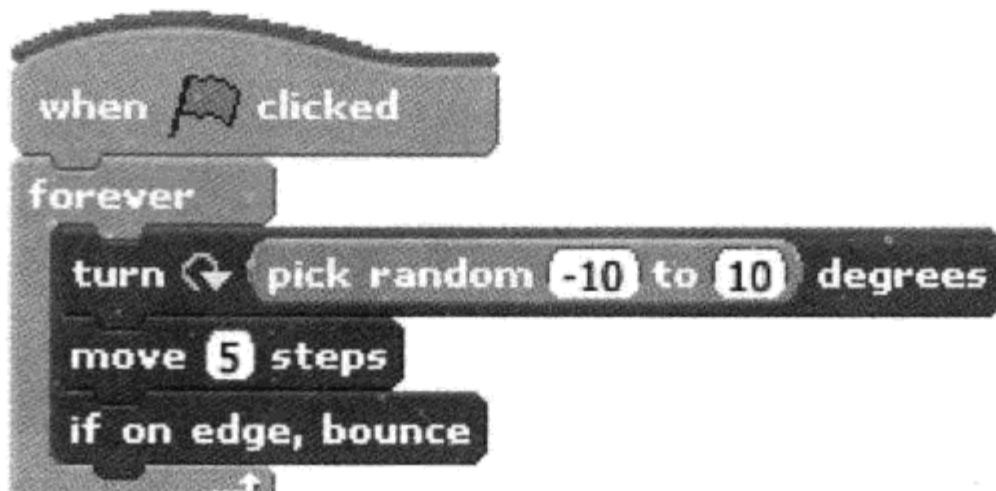


图5.16

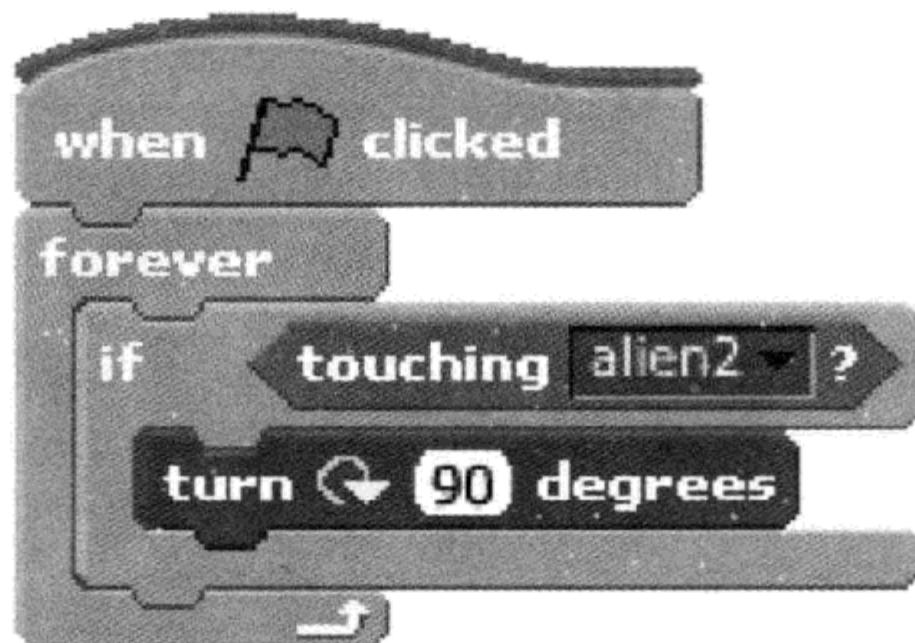


图5.17

如果子弹与外星飞船发生碰撞，我们需要让飞船发生爆炸，如图 5.18 所示。



图5.18

爆炸通过把造型改变成 explosion 来实现，并且添加一小段时间的特效，然后把这个角色隐藏起来，这样它就从舞台上消失了。



几秒钟之后，让外星飞船重新出现在舞台上部的一个随机位置上。

这就是我们为外星飞船设定的全部脚本，如果你有多个角色需要使用类似的脚本，可以通过在积木上点击鼠标右键并选择“复制”（Duplicate）来复制脚本。重复这个操作，并把复制出来的每一段脚本拖到相应的外星飞船上。然后选择第二艘外星飞船，并在脚本中修改它的起始位置和碰撞处理行为，如图 5.19 所示。

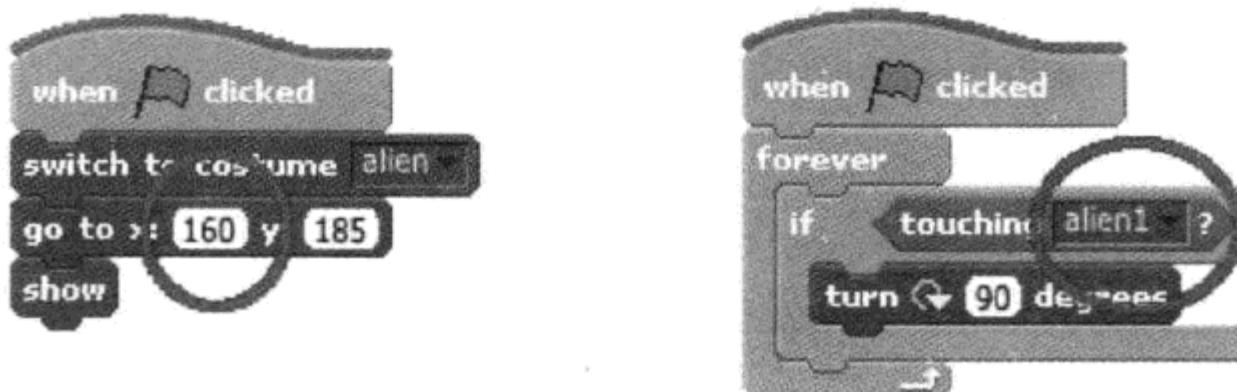


图5.19

接下来，选择加农炮角色。首先把它移动到一个合适的位置，并确保它处于非爆炸的状态，如图 5.20 所示。



图5.20

加农炮只能用键盘上的方向键来控制左右移动，图 5.21 是处理键盘按键的脚本。

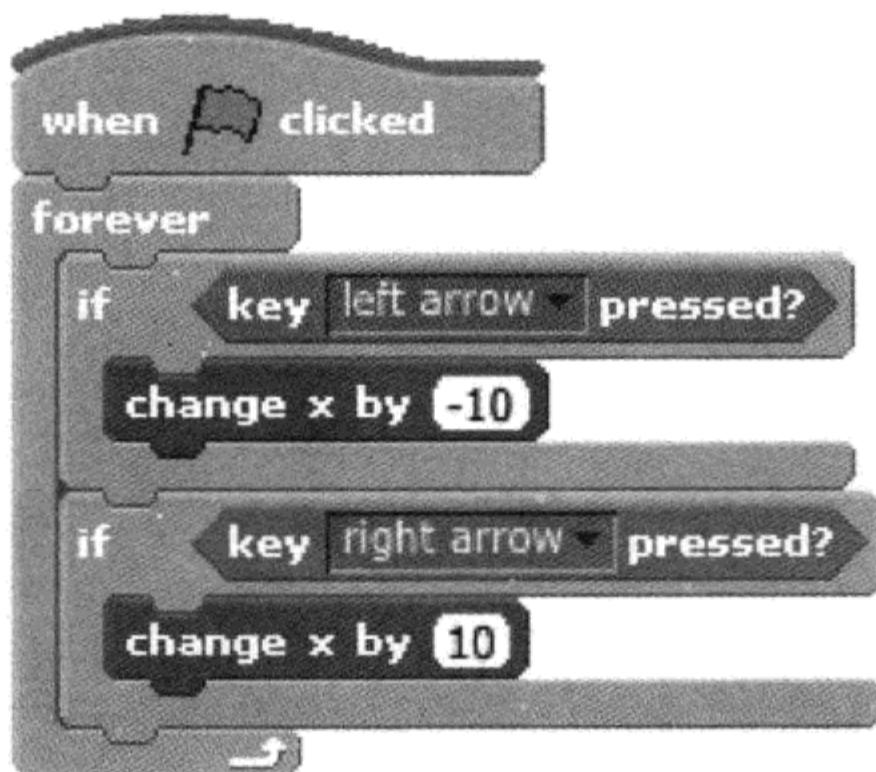


图5.21

使用“侦测”（Sensor）积木盒中的“碰到”（touching）积木，可以判断外星飞船是否撞到了加农炮。如果撞到了，就把加农炮设置为爆炸状态，具体的做法与让飞船爆炸一致。加农炮爆炸后，会向所有的角色发送一个游戏结束的消息。这个消息会被 GameOver 角色捕获，并停止脚本的运行，如图 5.22 所示。

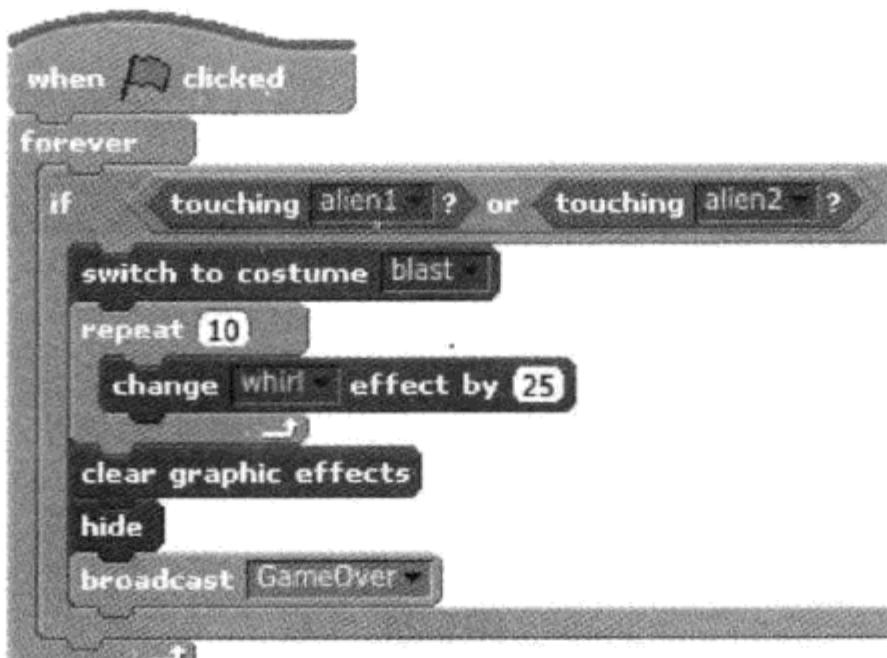


图5.22



下面来看子弹角色，为子弹角色添加图 5.23 所示的脚本。在按下空格键之前，子弹是不可见的。

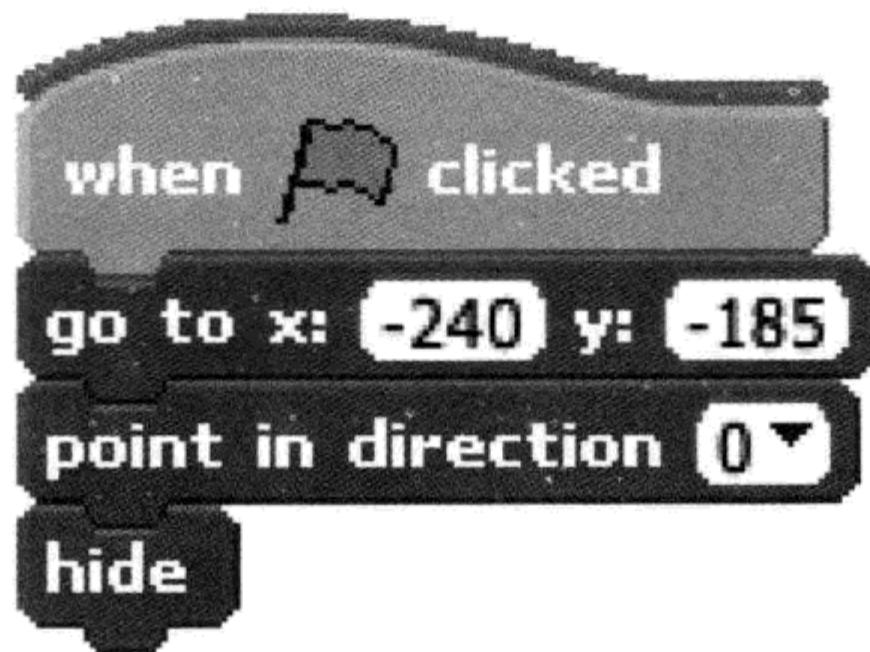


图5.23

每次按下空格键时，子弹会移动到加农炮的当前位置，变为可见并开始向舞台顶端移动，直到遇到舞台边界为止。

图 5.24 就是处理空格键按下消息的脚本。

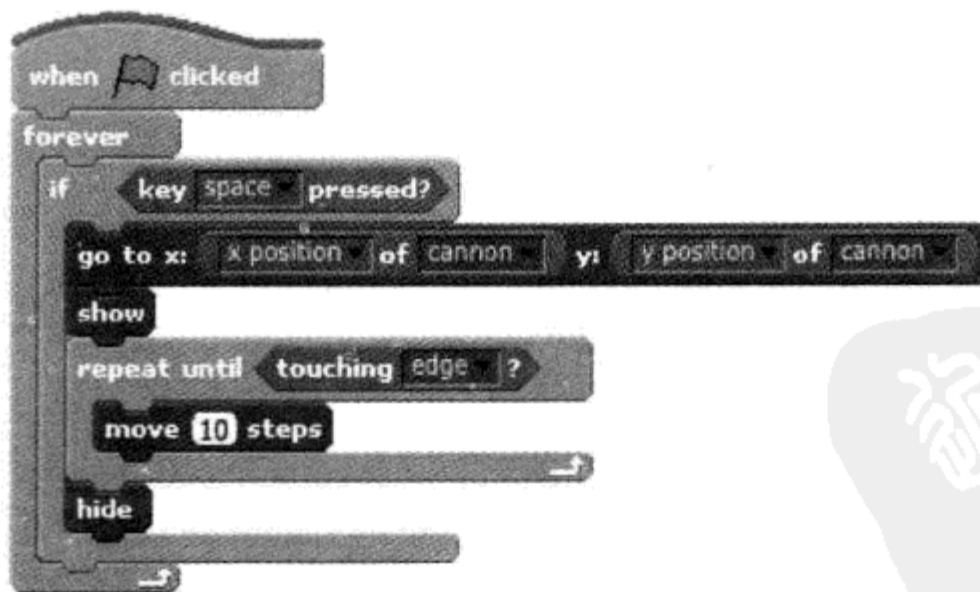


图5.24

最后一个角色是游戏结束角色。这个角色行为很简单：在接



收到 GameOver 消息以前，保持不可见的状态；接收到 GameOver 消息后，把它显示出来，并停止运行脚本，如图 5.25 所示。



图5.25

这就是整个游戏的实现过程。请参考附录 B 中完整的游戏设计。

Scratch 与现实世界

如果你注意观察“侦测”（Sensor）积木盒，你会发现两块有意思的积木：“传感器”（sensor）和“传感器的值”（sensor value）。这些积木是用于与一块名为 PicoBoard（图 5.26）的外接传感器板配合使用的。PicoBoard 上有一块微处理器芯片，可以把传感器获取到的值通过 USB 连接返回给 Scratch。

PicoBoard 上有一个按钮、一个滑杆、一个声音传感器和 4 个可以连接其他任何模拟传感器的接口。可以使用鳄鱼夹来连接各种传感器，读取模拟信号值。

PicoBoard 与 Scratch 之间通过一种专有的 PicoBoard 协议进行数据通信。S4A（Scratch for Arduino）项目（<http://seaside.citilab.eu/scratch/arduino>）为 Arduino 实现了相同的协议，所以你也可以用 Arduino 把 Scratch 与现实世界相接。S4A 需要一个专用版本的 Scratch 才能支持，这个特殊版本的 Scratch 在 S4A 的主页上可以下载。

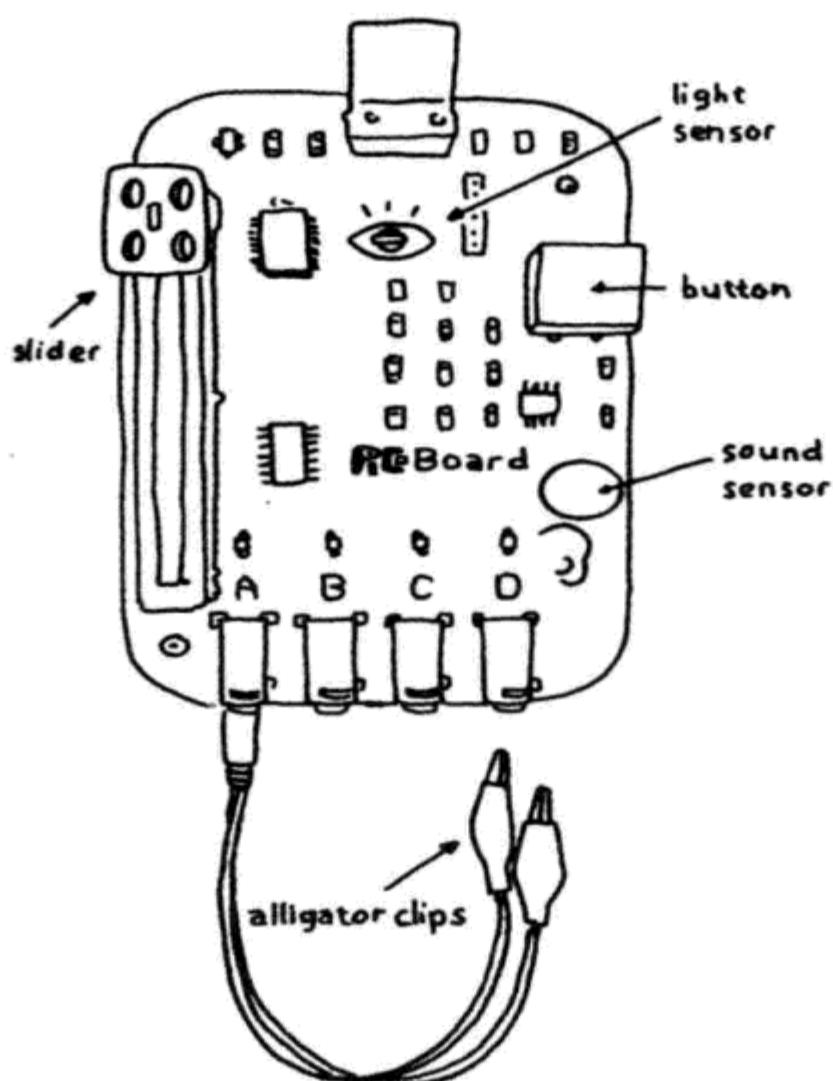


图5.26 PicoBoard是一块专门用于Scratch的传感器附件

分享你的程序

Scratch 有一个很有意思的特性就是内置了一个社区功能。除了先前提到的“来个令人惊喜的角色吧”(Get Surprise Sprite)工具(用于随机从其他 Scratch 用户那里获取角色)，它还提供了一个分享的功能，可以把你自己的 Scratch 项目打包并上传到 MIT 的 Scratch 项目页面(<http://scratch.mit.edu/>)。最新的数据显示，现在有超过 120 万的 Scratch 用户，分享了超过 280 万个 Scratch 项目。

之所以会有这么多项目被分享出来，是因为分享一个项目非常简单。在 scratch.mit.edu (<http://scratch.mit.edu/>) 注册账号后，只需点击“分享”(Share) 菜单中的“将此作品在网络上分享”(Share This Project)



Online），输入一些必要的信息后（图 5.27）你的项目就被上传到网站上了。每个上传到网站上的项目不能超过 10M，所以你可能需要把项目中的图片和声音进行压缩，参见“编辑”（Edit）菜单中的选项。通过 Scratch 项目主页，你可以了解到这个平台的各种玩法。

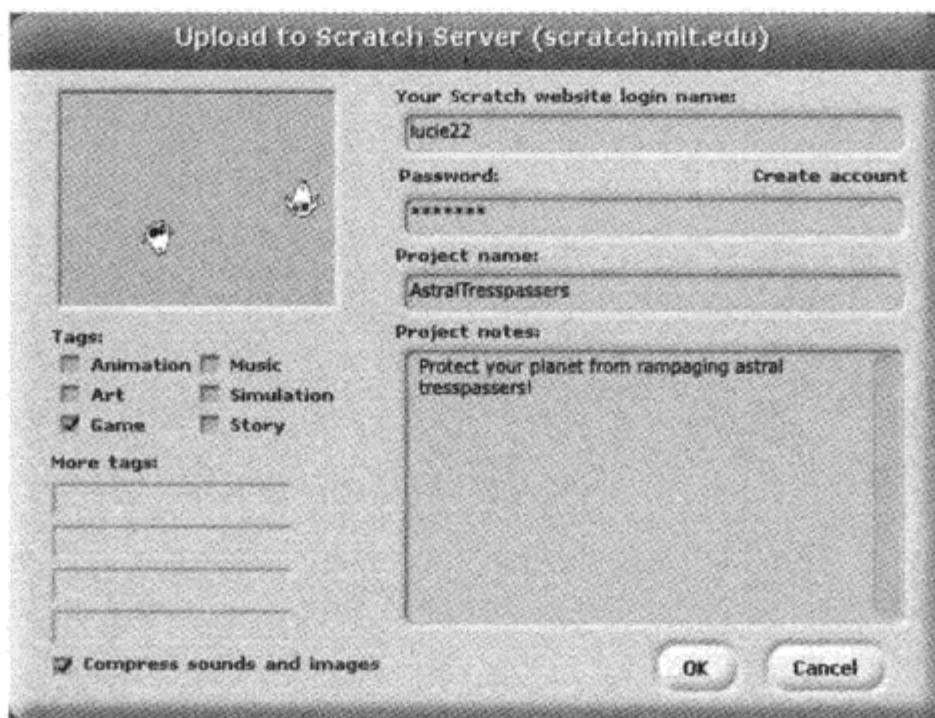


图5.27 当你完成了一个作品，可以用Scratch内置的功能把它分享到MIT Scratch项目的页面上。不过遗憾的是这个网页上使用了Flash动画，所以在Raspberry Pi上无法直接访问

进一步学习

Scratch 的 Wiki 页面

(http://wiki.scratch.mit.edu/wiki/Scratch_Wiki)

MIT 官方的 Scratch 参考资料。

MIT 的 Scratch 页面

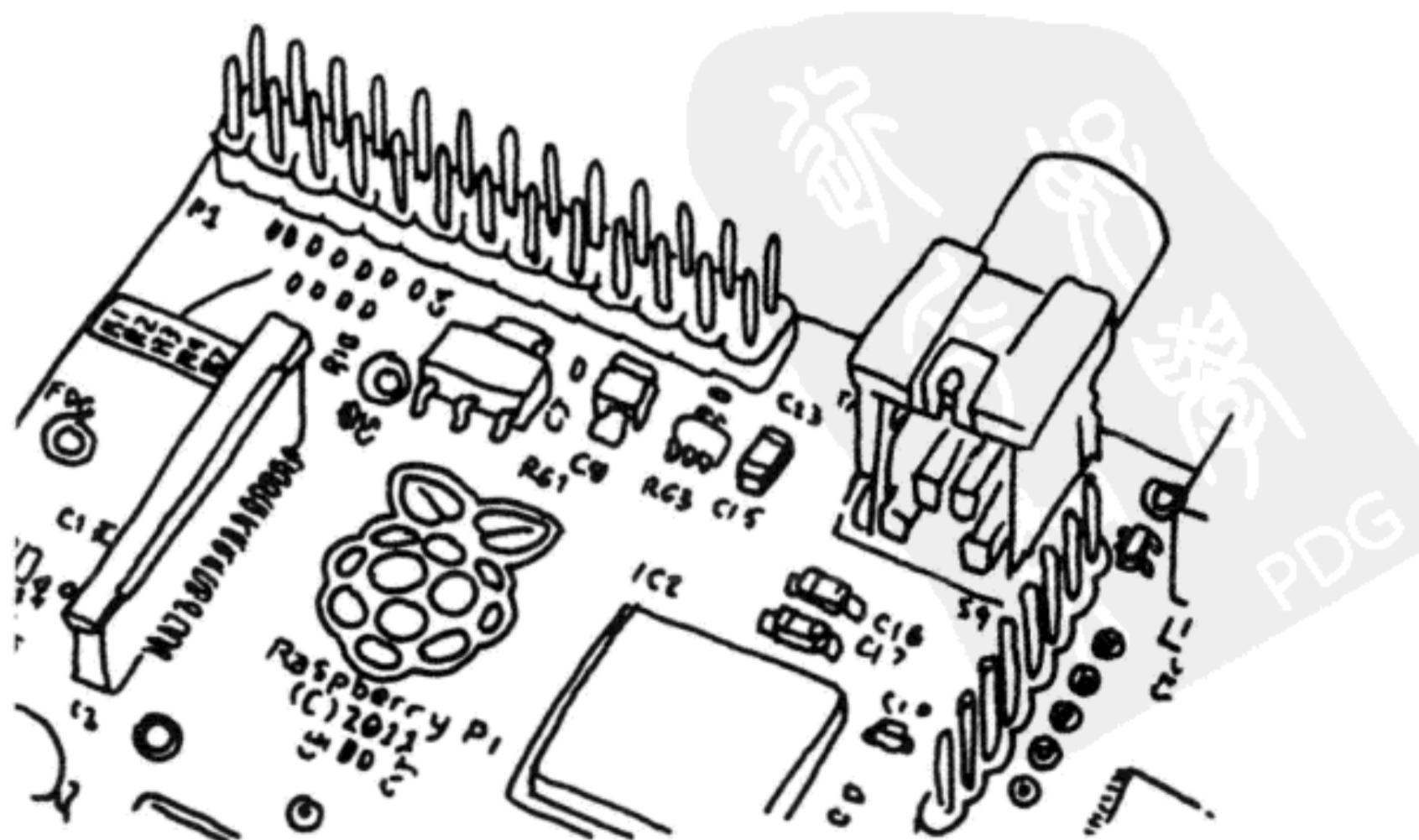
(<http://scratch.mit.edu/>)

Scratch 的社区网站，上面有成千上万的项目和注册会员。

第6章

Arduino与Pi

Arduino and the Pi





后续的几个章节中，我们会使用 Raspberry Pi 的 GPIO 接口连接一些传感器、LED 或电动机。如果你有使用 Arduino 开发板的经验，你可以把它与 Raspberry Pi 配合起来一起使用。

当 Raspberry Pi 刚刚发布时，很多人把它看成“Arduino 杀手”。你可以用相似的价格购买到更为强大的处理能力，有了 Pi，为什么还需要 Arduino？其实，Raspberry Pi 与 Arduino 这两个平台是互补的，Raspberry Pi 不但可以用做 Arduino 的上位机，把它们组合在一起还可以有更多的应用前景（图 6.1）。

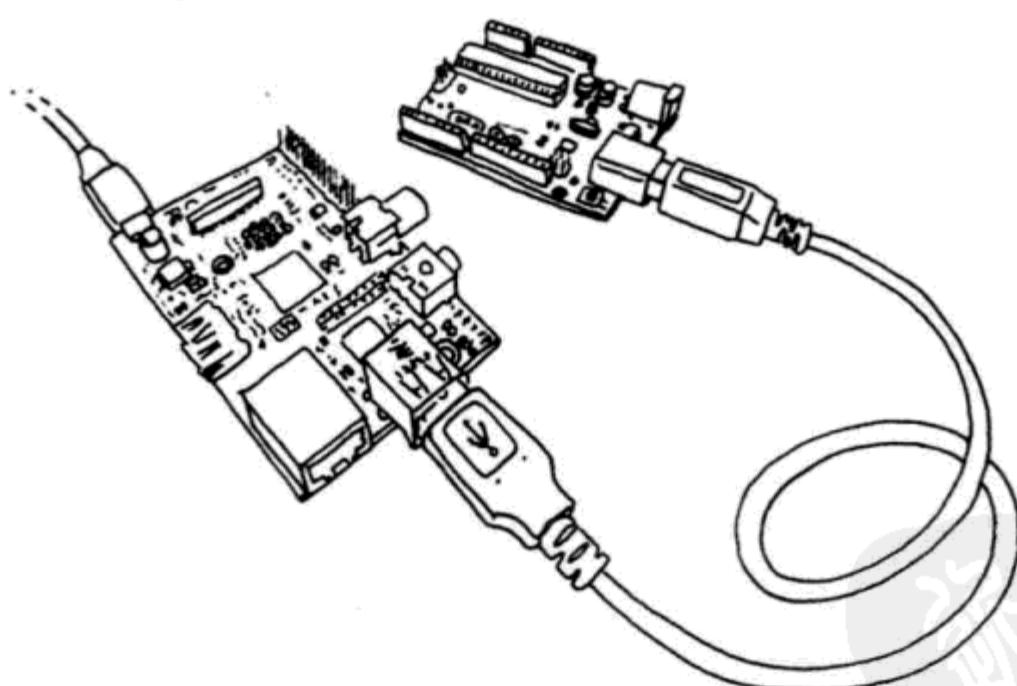


图6.1 Arduino与Raspberry Pi是“史上最好的朋友”

- 它们可以共享很多有用的开发库或示例程序。
- 如果你已经熟悉 Arduino，但你需要更强大的处理能力，如你有一个 MIDI 控制器连接到一个音序器上，你现在可以把它升级



为直接在 Pi 上合成音乐。

- 你需要操作 5V 的逻辑电路。Pi 工作在 3.3V 上，它的接口不能连接 5V 的信号。
- 尝试一些你比较陌生的电路设计时，可能会把芯片烧毁。我曾经见过学生错误地用 Arduino 的输出接口去驱动电动机（请不要尝试这么做），这样会烧坏控制芯片。在 Arduino 上，你只需把烧坏的芯片拔掉换一块新的（不到 10 美元），但在 Raspberry Pi 上，你做不到这一点。
- 有时你需要真正意义上的实时控制，如做一个 3D 打印机的控制器。我们在第 3 章中提到过，Raspbian 不是一个实时操作系统，所以程序不能像单片机上那样每次都严格按照相同的时钟周期来运行。

本章内容的安排基于你已经具备了 Arduino 开发板和它的集成开发环境（IDE）基础知识的假设，如果你还不了解这些基础知识，Massimo Banzi 所著的 *Getting Started with Arduino* (<http://shop.oreilly.com/product/0636920021414.do>) 是一本很好的入门书。官方的 Arduino 教程 (<http://arduino.cc/en/Tutorial/HomePage>) 也是一个很好的参考资料，它提供了很多可以直接使用的代码示例。

在 Raspbian 上安装 Arduino

开发 Arduino 程序，你需要通过 USB 接口把它与电脑相连，然后在 Arduino IDE 中编译程序并烧录到 Arduino 上。你可以在任何一台电脑上完成这个操作，同样也可以在 Raspberry Pi 上完成。

用 Raspberry Pi 来进行 Arduino 开发会便于调试错误，但与笔记本或台式电脑相比，编译程序的速度可能会慢一些。这不是一个严重的问题，由于 Arduino 只会编译那些被改动过的代码，因此完



成一次编译以后，后面再次编译时速度会快很多。

在 Raspberry Pi 上安装 Arduino IDE 的命令：

```
sudo apt-get update①
sudo apt-get install arduino②
```

① 把软件仓库更新到最新版本。

② 下载安装 Arduino 软件包。

这个命令会安装 Java 和其他很多依赖包，安装完成后 Arduino IDE 的图标会出现在程序菜单的 Electronics 子菜单中（现在请暂时不要运行它）。

如果使用无显示器运行 Pi 的模式，你可以直接把 Arduino 连接到 Pi 的某个 USB 接口上。如果你的 Pi 上已经没空闲的 USB 接口，你可以把 Arduino 连接到你键盘上的 USB 扩展口（如果有的话），或者使用一个 USB Hub 来连接这些设备。正常情况下，USB 接口可以给 Arduino 提供足够的电流，但为了保险起见，你也可以给 Arduino 单独供电。



注意，你可能需要在 Raspberry Pi 启动完成后再连接 Arduino 的 USB 电缆。如果在 Raspberry Pi 的启动过程中就连接了 Arduino，Raspberry Pi 可能会因为试图识别 USB 总线上的所有的设备而在启动时失去响应。

当你运行 Arduino IDE 时，它会轮询所有的 USB 设备并在 Tools → Serial Port 中列出一个设备列表。你需要确保 pi 用户拥有足够的权限去操作串口设备，可以通过把 pi 用户加到 tty 和 dialout 用户组来满足这一点。你需要在启动 Arduino IDE 前完成这些操作。



```
sudo usermod❶ -a -G❷ tty pi
sudo usermod -a -G dialout pi
```

- ❶ usermod 是 Linux 中用于管理用户的命令。
- ❷ -a -G 参数用于把指定用户 (pi) 加入指定的用户组 (tty 和 dialout)。

现在，你可以运行 Arduino 了。点击 Tools → Serial Port，然后选择需要的串口（通常是 */dev/ttyACM0* 这种格式的），然后点击 Tools → Board 选择你的 Arduino 板的型号（如 Uno）。点击 File → Examples → 01.Basics → Blink 加载一个基本的示例 Sketch，点击工具条上的 Upload 按钮或选择 File → Upload 把 Sketch 下载到 Arduino 上，当这个 Sketch 被加载运行后，Arduino 上的指示灯会开始闪烁。

定位串口

有时候，由于某些原因，*/dev/ttyACM0* 并不是 Arduino 所对应的串口设备名，这时你就需要一些额外的方法来找到正确的串口设备。可以使用下面方法，在不查看 Arduino IDE 上的菜单的情况下，识别出连接了 Arduino 的串口。先不要连接 Arduino，输入：

```
ls /dev/tty*
```

连接 Arduino，然后再次输入相同的命令，看输出的结果有什么变化。在我的 Raspberry Pi 上，起初只列出了 */dev/ttAMA0*（这实际上是板载的 USB Hub），当我插入 Arduino 后，*/dev/ttyACM0* 设备就出现在列表中了。



改善用户体验

当你把 Arduino IDE 启动成功后，你可能会发现 Arduino 代码编辑器的默认字体并不美观。你可以通过下载开源字体 Inconsolata 来改善显示效果。输入：

```
sudo apt-get install ttf-inconsolata
```

然后，编辑 Arduino 的配置文件：

```
nano ~/.arduino/preferences.txt
```

再修改如下两行：

```
editor.font=Inconsolata,medium,14  
editor.antialias=true
```

下次启动 Arduino 时，代码编辑器就会使用新设置的字体了。

串口通信

要让 Raspberry Pi 和 Arduino 通过串口通信，在 Arduino 端你需要使用内置的 Serial 库，在 Raspberry Pi 端需要使用 Python 的 pySerial 串口通信模块 (<http://pyserial.sourceforge.net/>)。用下面的代码安装 Python 的串口通信模块：

```
sudo apt-get install python-serial python3-serial
```

打开 Arduino IDE，把下面的代码下载到 Arduino 上：

```
void setup() {  
  Serial.begin(9600);  
}
```



```
void loop() {
    for (byte n = 0; n < 255; n++) {
        Serial.write(n);
        delay(50);
    }
}
```

这段程序把一个递增的数字序列发送到串口上。



Arduino 的 `Serial.write()` 函数发送的是实际的数字而不是字符串，所以在我们的例子中，发送的是 123 这样的数字而不是 "123" 这样的字符串，如果要发送字符串，需要使用 `Serial.print()` 函数。

然后，你需要知道 Arduino 连接到了哪个串口上（参考“定位串口”的内容）。下面是在 Raspberry Pi 上运行的 Python 脚本，如果你所连接的串口不是 `/dev/ttypACM0`，请把 `port` 的值做相应的修改（参考第 3 章了解更多有关 Python 的知识）。把它保存为 `SerialEcho.py`，并用 `python SerialEcho.py` 命令运行它：

```
import serial

port = "/dev/ttypACM0"
serialFromArduino = serial.Serial(port, 9600)❶
serialFromArduino.flushInput()❷
while True:
    if (serialFromArduino.inWaiting() > 0):
        input = serialFromArduino.read(1)❸
        print(ord(input))❹
```

- ❶ 打开串口，连接到 Arduino 上。
- ❷ 清空输入缓冲区。
- ❸ 从串口缓冲区读入一个字节数据。



④ 用 `ord()` 函数把读入的字节数据转换为实际的数值。



当 Python 程序把串口打开时，你无法向 Arduino 下载数据，所以在向 Arduino 再次下载 Sketch 前，需要先按 Control-C 键中断 Python 程序的运行。如果你使用的是 Arduino Leonardo 或 Arduino Micro 型号的 Arduino 板，则可以在 Python 脚本运行过程中下载 Sketch，但这样做会导致 Python 程序中的串口连接断开，所以你总是需要重启这个 Python 脚本。

在上面的例子中，Arduino 向 Python 脚本发送数字，而 Python 脚本会把这个数字解释为字符串。程序中的 `input` 变量中存放的是这个数字在 ASCII 表(<http://en.wikipedia.org/wiki/ASCII>)中对应的字符。为了更好地理解这一点，把上面 Python 脚本中的最后一行改写为

```
print(str(ord(input)) + " = the ASCII character " + input + ".")
```

把串口设备名作为参数传递

如果你想把串口的设备名作为命令行参数传递给 Python 脚本，可以使用 `sys` 模块在程序中获取传入的第一个参数：

```
import serial, sys

if (len(sys.argv) != 2):
    print("Usage: python ReadSerial.py port")
    sys.exit()
port = sys.argv[1]
```

经过这样的改动以后，你可以通过这样命令来运行这个脚本：`python SerialEcho.py /dev/ttyACM0`。



在前面的例子中，程序每次只发送一个字节，如果你只是想从 Arduino 发送一些消息代码，这是可以的。例如，当左边的按钮被按下，发送 1；右边的按钮被按下，发送 2。然而，这样的方式最多只能支持 255 种不同的消息，而在很多时候，你可能需要发送一个更大的数字或者一个特定的字符串。例如，想通过 Arduino 读取一个输出模拟信号的传感器的值，就需要发送 0 ~ 1023 的数值到 Raspberry Pi 上。

在很多编程语言中，一个字节一个字节地处理传入的信息会比你所想象的更复杂。不过，有了 Python 和 PySerial 模块，处理字符串就变得非常容易。下面是一个简单的例子，上传到 Arduino 后可以让它依次发送 0 ~ 1024 的值。

```
void setup() {
    Serial.begin(9600);
}

void loop() {
    for (int n = 0; n < 1024; n++) {
        Serial.println(n, DEC);
        delay(50);
    }
}
```

与之前的程序相比，这个程序最大的区别在于 `println()` 命令。之前的 `Serial.write()` 命令可以把原始的数字直接发送到串口上，在这个程序中，通过使用 `println()`，Arduino 把数字以十进制的方式转换为字符串，并以 ASCII 码的方式发送。所以这个程序不会再发送数字 254，而会发送字符串 "254\r\n"。其中，`\r` 表示回车，`\n` 表示换行（回车和换行这两个概念起源于老式的机械打字机：回车表示移动到一行的起位置，换行表示移动到新的一行上）。



在 Python 脚本中，可以用 `readline()` 替换先前的 `read()`，`readline()` 会一次性读入一行字符串，直到遇到（并包含）回车和换行为止。Python 提供了一系列灵活的函数，可以把各种类型的数据与字符串进行转换。在这个例子中，可以用 `int()` 函数把读到的字符串转换为整型数：

```
import serial
port = "/dev/ttyACM0"

serialFromArduino = serial.Serial(port, 9600)
serialFromArduino.flushInput()
while True:
    input = serialFromArduino.readline()❶
    inputAsInteger = int(input)❷
    print(inputAsInteger * 10)❸
```

- ❶ 把整个字符串读入 `input` 变量。
- ❷ 把它转换为整型数。
- ❸ 把它乘以 10 以后显示出来。在这里做乘 10 的操作是为了证明它是一个整型数而不是一个字符串。

下面是一个简单的示例程序，用于从 Arduino 的模拟输入口上读取模拟值并发送到串口。只需改动前面的程序中的循环部分即可：

```
void setup() {
    Serial.begin(9600);
}

void loop() {
    int n = analogRead(A0);
    Serial.println(n, DEC);
    delay(100);
}
```

如果修改你的 Python 程序，把输出 `inputAsInteger * 10` 改



为直接输出 `inputAsInteger`，在 Arduino 的模拟输入口 0 上没有连接任何电路的情况下，你应该会接收到一个小于 200 且不断变化的值。如果你用接线把模拟输入口与地线（GND）相接，接收到的值应该变为 0；如果与 3.3V 电源（3V3）相接，应该接收到 715 左右的一个值；如果与 5V 电源相接，应该接收到 1023。

进一步学习

一旦你安装了正确的软件，你会发现所有的操作都很简单，很多实际的电子项目的代码也都与我们例子中的串口通信代码非常相似。不过，任何形式的通信，在你学会了“hello world”这样的人门知识后，都会慢慢变得很复杂，你必须定义或使用一种通信协议，让通信的双方可以相互理解。有关串口通信的具体协议内容已经超出了本书的范围，但是在 Arduino Playground (<http://www.arduino.cc/playground/>) 的 Interfacing with Software (<http://www.arduino.cc/playground/Main/InterfacingWithSoftware>) 部分有很多示例代码，演示了人们是如何解决实际问题的。

Firmata

Hans-Christoph Steiner 的 Firmata (<http://arduino.cc/en/Reference/Firmata>) 是一个久经考验的全功能的串口协议。Firmata 协议非常简单，并且协议内容是可读的。虽然它可能并不适合所有的应用，但用它来入门还是非常出色的。

MIDI

如果你的项目与音乐有关，就可以考虑使用 MIDI 消息作为你的串口协议。因为 MIDI 消息（基本上）是串行的，它应该可以直接使用。



兼容 Arduino 的 Raspberry Pi 扩展板

市面上有一些扩展板可以把 Raspberry Pi 的 GPIO 口与 Arduino 兼容的开发板连接。例如，WyoLum 的 AlaMode 扩展板 (<http://baldwisdom.com/projects/alamode/>) 就是一个不错的解决方案，它还提供了 RTC 等附件。

通过网络通信

你也可以摆脱串口通信，通过网络来连接 Arduino 和 Raspberry Pi。有很多有趣的应用都使用 Node.js (<http://nodejs.org/>) 这个基于 JavaScript 的平台加上 WebSocket 协议 (<http://www.websocket.org/>) 来实现。如果对这项技术有兴趣，可以先学习一下 Noduino 项目 (<http://semu.github.com/noduino/>)。

使用 Raspberry Pi 上的串口引脚

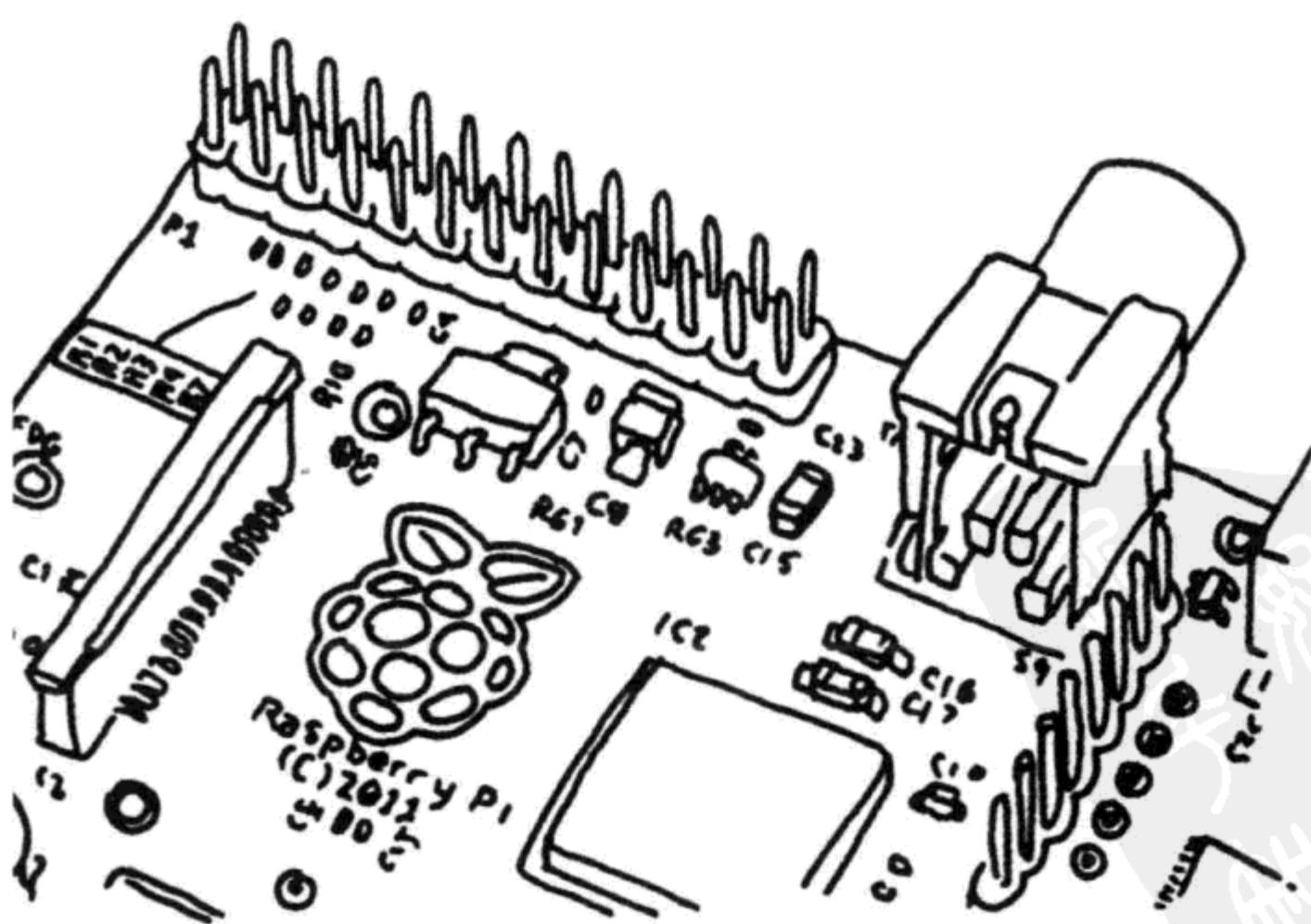
Raspberry Pi 上有一些引脚提供了输入输出的功能，其中包含了两个用于跳过 USB 接口直接收发串口信息的接口。如果你需要使用它们，可以先参考一下第 8 章中的内容，并且你需要用电平转换电路把 Arduino 的 5V 信号转换为 Raspberry Pi 所能接受的 3.3V 信号。

如果你还想更深入了解物理设备之间通信的相关知识，可以阅读 Tom Igoe 的 *Making Things Talk* 一书 (<http://shop.oreilly.com/product/9780596510510.do>)。

第 7 章

基本输入输出

Basic Input and Output





Raspberry Pi 本质上是一台廉价的 Linux 电脑，但它与我们平时用于上网、写邮件或进行文字处理的台式电脑或笔记本仍有一些不同之处。其中一个最明显的差别就是，Raspberry Pi 的主板上提供了一组 GPIO (General Purpose Input & Output，通用输入输出) 接口，这些接口可以直接用于开发一些电子相关的项目，如图 7.1 所示。

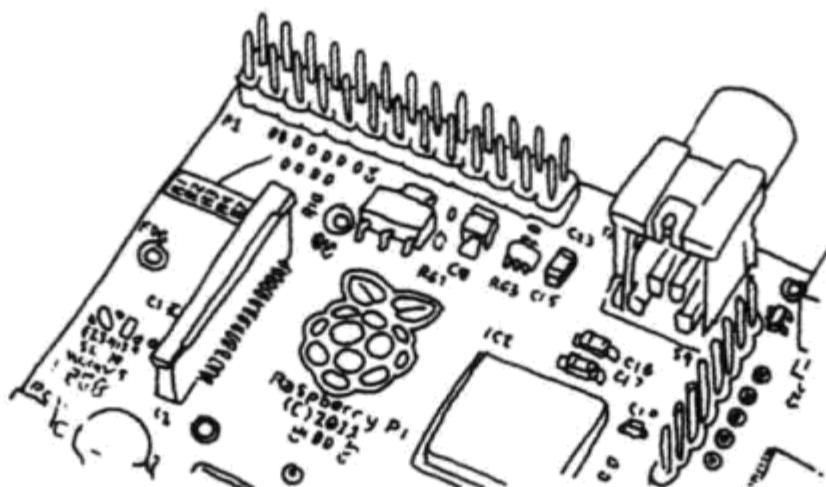


图7.1 Raspberry Pi的GPIO接口

这些 GPIO 接口可以作为控制信号输出口来控制一些硬件设备，如发光二极管 (LED) 、电动机或继电器。它们也可以作为信号输入接口，用于读取按钮或开关的状态，获取键盘上按下的键，或者获得温度、光线、运动和距离等各种传感器的状态。



Raspberry Pi 的 GPIO 有一个小小的缺点——它无法直接连接到以模拟信号输出的传感器上，如模拟亮度传感器或温度传感器。必须通过一个模数转换器 (ADC) 芯片才能让 Raspberry Pi 读取模拟值。相关的内容请参考附录 C。



如果电脑具备 GPIO，它的最大优势在于你可以编写程序从 GPIO 口上读取状态或根据不同的条件向 GPIO 口输出值，就与给普通台式电脑编写程序一样方便。Raspberry Pi 与其他具备 GPIO 口的单片机开发板不同之处在于，它还具备键盘、鼠标、显示器及以太网口等输入输出设备。如果你有过其他单片机的开发经验（如 Arduino 开发板），Raspberry Pi 的 GPIO 接口虽然比它要少，但 Raspberry Pi 自带的其他输入输出设备不需要额外的电路或接线就可以直接使用，用起来更为方便。

具备键盘、鼠标、显示器等接口不是 Raspberry Pi 区别于其他单片机开发板的唯一优点，在开发电子项目时，它还具备以下的优点。

文件系统

可以直接读写 Linux 文件系统，会使很多项目实现起来更为容易。比如，把一个温度传感器连接到 Raspberry Pi，每秒读取一次温度值。你可以把每次读到的值添加到一个日志文件中，这个文件后期可以很方便地下载到电脑中并用于绘制一条温度变化曲线。你甚至可以直接在 Raspberry Pi 上完成这项绘图的工作。

Linux 工具

Raspberry Pi 的 Linux 发行版中包含了一系列的命令行工具，你可以使用这些工具来操作文件、控制进程或把很多任务进行自动化处理。这些灵活的工具都可以在你的项目中直接使用。由于有很多 Linux 用户都依赖这些工具来完成日常工作，所以你遇到问题时，可以很方便地在网上搜索到答案。有关 Linux 相关的问题，你常常可以在 Stack Overflow 网站 (<http://stackoverflow.com>) 上找到答案。如果遇到了 Raspberry Pi 上特有的问题，可以尝试访问 Raspberry Pi



的论坛 (<http://www.raspberrypi.org/phpBB3/>) 或 Stack Overflow 上的 Raspberry Pi 版块 (<http://raspberrypi.stackexchange.com>)。

编程语言

在 Raspberry Pi 这样的嵌入式 Linux 系统中，系统支持很多不同的开发语言，可以选择你自己最喜欢的一种来进行开发。在本书中，主要使用 Shell 和 Python 作为开发语言，但你也可以很方便地把它们转换为 C、Java、Perl 或其他任何一种开发语言。

使用输入输出接口

除了 Raspberry Pi 本身以外，你还需要一些其他的配件，才可以完成下面的实验。这些配件通常可以在本地的电子市场里购买到，也可以在网上商店选购。下面是一个简要的清单。

- 面包板。
- 各种发光二极管（LED）。
- 杜邦线（公头对公头）。
- 母头对公头连接线，用于连接 Raspberry Pi 的 GPIO 接口到面包板上，这种线比较少见。
- 小按钮。
- 各种电阻。

为了更容易地连接 Raspberry Pi 与面包板，我们推荐使用 Adafruit 的 Pi Cobbler Breakout 套件，使用它可以省去母头对公头的连接线。这个套件是以散件的形式提供的，你需要自己进行焊接，焊接的过程并不复杂，可以参考 Adafruit 的教程 (<http://learn.adafruit.com/adafruit-pi-cobbler-kit/overview>) 一步步地完成。MAKE 的 Raspberry Pi 入门套装 (<http://oreil.ly/pikit>) 中包含了 Adafruit 的 Pi Cobbler Breakout 套件，并同时提供了上面所列出的其他相



关元件（除了母头对公头连接线，因为用了 Adafruit 的 Pi Cobbler Breakout 套件后，这种连接线就不需要了）。

图 7.2 中，我们为 Raspberry Pi 的每个 GPIO 接口引脚标记了对应的编号，在你的程序中，需要使用这些编号来操作对应的引脚。图中没有标记的引脚默认被设置为其他的用途。

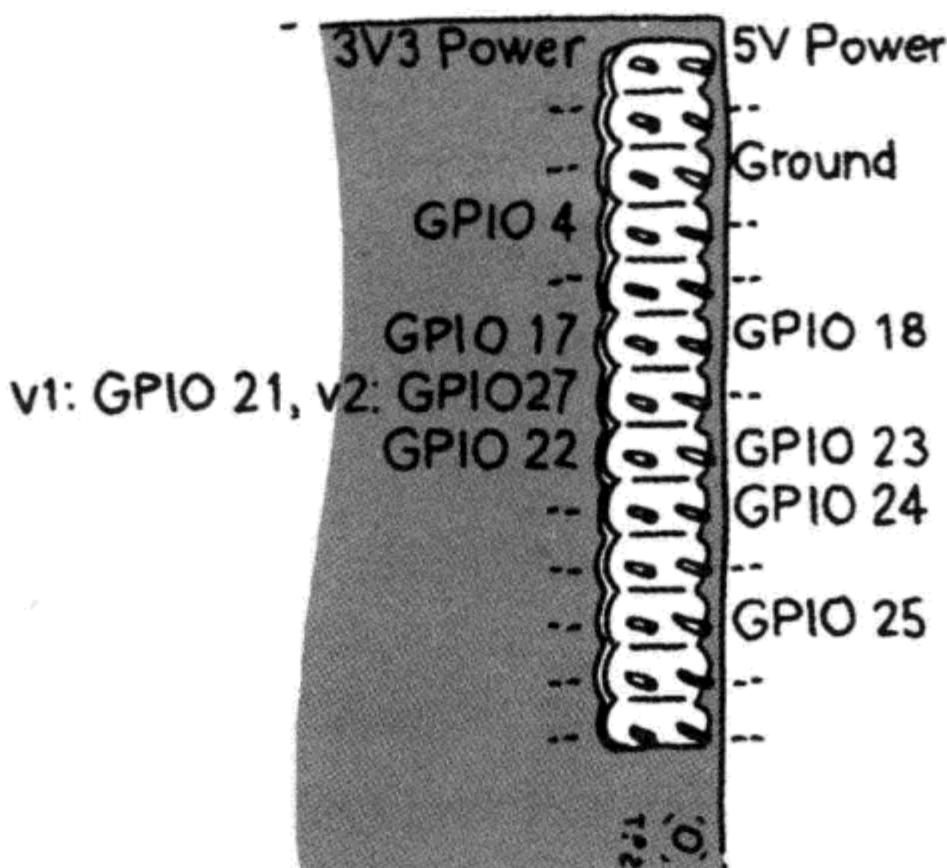


图7.2 Raspberry Pi上默认的GPIO接口。在较新版本的Raspberry Pi上，GPIO 21被改成GPIO 27

也许你注意到图 7.2 中有一个针脚上被标注了两个不同的 GPIO 口编号。在较新版本的 Raspberry Pi 主板上，GPIO 21 变成了 GPIO 27。可以在命令行中执行 `cat /proc/cpuinfo` 来获知你的 Raspberry Pi 属于哪个版本，如果显示出来的版本号是 0002 或 0003，你使用的板子就是第一版的。如果显示出来的版本号比 0003 更大或者包含字母，则你使用的板子是第二版的。





数字信号输出：点亮 LED

使用 GPIO 接口的最简单的方法就是连接一个发光二极管（LED），你可以使用 Linux 命令行来点亮或熄灭 LED。一旦你理解了这些命令背后的工作原理，就可以很容易地使用 LED 来提醒你收到了一封新的邮件、下雨天出门前要带上雨伞或到了上床睡觉的时间。除了控制 LED 外，使用一个继电器来定时控制一个灯的开关，也是很容易的。

面包板使用方法入门

如果你以前从来没有使用过面包板，那首先要了解的是面包板上的插孔是如何连接在一起的。图 7.3 中，我们把一块常见的面包板上彼此相连的插孔用阴影包围在一起。要注意，左右两侧的供电总线之间并不相连。如果需要让两侧的供电总线同时提供电源和接地，你需要使用两根公头对公头的连接线把它们连到一起。

1. 使用公头对母头的连接线，把 Raspberry Pi 的 GPIO 25 与面包板相连。请参考图 7.2 了解 Raspberry Pi 的引脚与 GPIO 接口编号之间的关系。
2. 使用另一根连接线，把 Raspberry Pi 的接地（Ground）脚与面包板上供电总线的负极相接。
3. 现在，你可以连接 LED 了（图 7.4）。在开始连接之前，你需要知道 LED 是有极性的：LED 的两个引脚的接线不能颠倒。LED 的引脚中，较长的一根是正极，应当与 GPIO 口相连；较短的一根则是负极，需要接地。另一种区别正负的方式是从 LED 的顶部往下看，

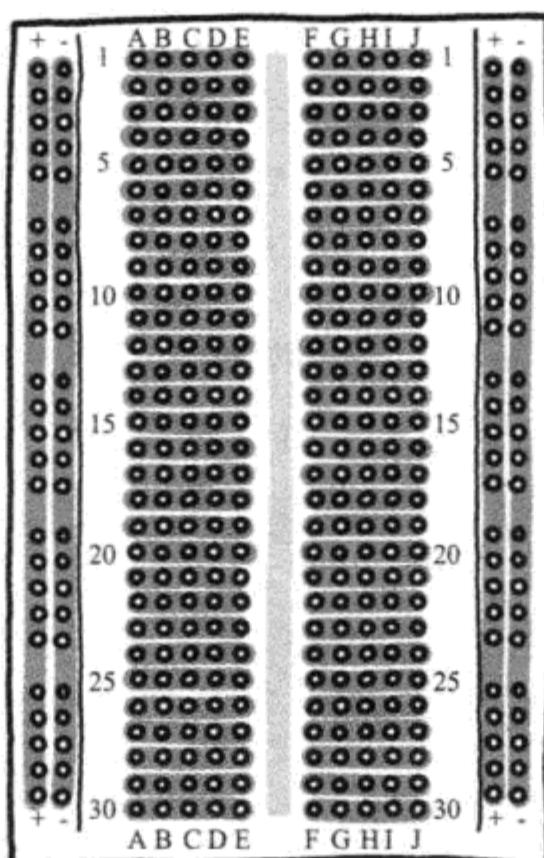
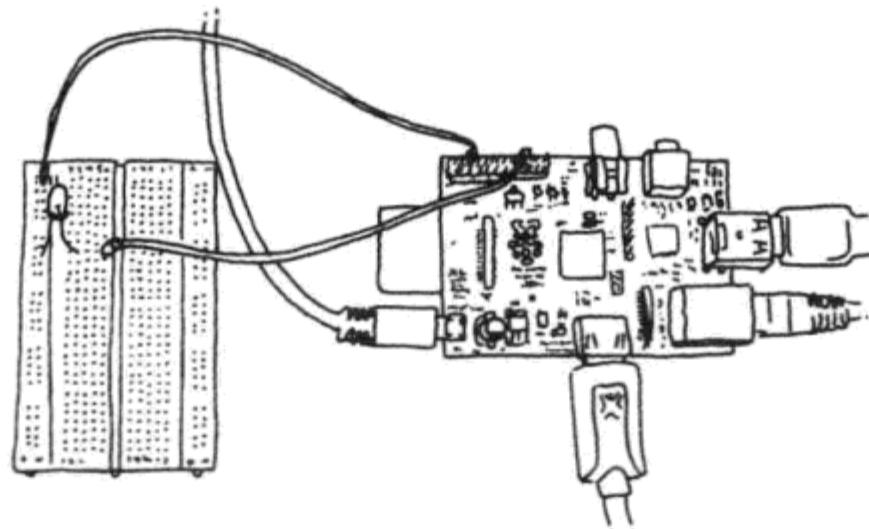


图7.3 面包板

LED 边缘被切平的一侧的引脚是 LED 的负极，需要接地。把 LED 的正极插入面包板上接 Raspberry Pi GPIO 25 的同一行，这就把它与 GPIO 25 相连了。同时，还需要把 LED 的负极与面包板上的电源总线的负极相接。

图7.4 把LED与Raspberry Pi相连^①

^① 如果书中此类手绘图中不容易看清实际的电路连接情况，请访问本书的支持网站 <http://rpi.freemindworld.com> 查看清晰的面包板连线图。——译者注



4. 接好你的键盘、鼠标和显示器，启动你的 Raspberry Pi 并登录系统。进入命令行模式，准备开始后面的操作。如果你在 X Window 环境中，双击桌面上的 LX 终端（LXTerminal）图标，打开一个命令行窗口。

5. 为了从命令行操作 Raspberry Pi 的输入输出接口，你需要以 **root** 权限（管理员权限）来运行相关的命令。通过输入 **sudo su** 并回车，可以把命令行切换到管理员权限模式下：

```
pi@raspberrypi ~ $ sudo su  
root@raspberrypi:/home/pi#
```

可以注意到，命令行的提示符发生了变化，提示你后续的命令都将以 **root** 权限运行。



root 账号具有管理员权限，可以对系统的所有文件和功能进行操作。如果你输入了一个可能破坏整个系统的命令，系统也不会对自己实施太多的保护措施，所以，当你以 **root** 权限来操作系统时，要格外小心。不过，在 Raspberry Pi 上，即使你做了一些错误的操作破坏了系统，也不需要过于担心，因为你总是可以通过重新烧录 SD 卡的镜像来安装一个全新的 Linux。当你完成了需要 **root** 权限的相关操作后，可以输入 **exit** 回到 **pi** 用户权限下。

6. 在开始使用命令来操作 GPIO 25 上所接的 LED 前，你需要把这个引脚的操作接口从内核空间暴露到用户空间中，使用这个命令：

```
root@raspberrypi:/home/pi# echo 25 > /sys/class/gpio/export
```



使用 `echo` 命令把你想要操作的 GPIO 编号 (25) 写入 `/sys/class/gpio` 下的 `export` 文件，当你把 GPIO 编号写入这个文件后，`/sys/class/gpio` 下会自动新建一个 `/sys/class/gpio/gpio25` 目录，里面包含了操作这个 GPIO 口所需要的控制文件。

7. 用 `cd` 命令切换到这个新建的目录下，并用 `ls` 命令列出文件清单：

```
root@raspberrypi:/home/pi# cd /sys/class/gpio/gpio25
root@raspberrypi:/sys/class/gpio/gpio25# ls
active_low  direction  edge  power  subsystem  uevent  value
```

`cd` 命令是 Change Directory (改变目录) 的缩写，它能把当前的工作目录切换到指定的目录下，这样就不需要每次都输入目录的完整路径。`ls` 命令可以列出目录下的文件和子目录。在这个目录下，你需要操作两个文件：`direction` 和 `value`。

8. `direction` 文件用于指定这个 GPIO 接口会被用于输入 (接一个按钮) 或是输出 (接一个 LED)。由于目前你所需要控制的 LED 连接在 GPIO 25 接口上，所以你应该把这个接口设置为输出模式：

```
root@raspberrypi:/sys/class/gpio/gpio25# echo out > direction
```

9. 使用 `echo` 命令向 `value` 文件中输出 1，你就可以点亮这个 GPIO 口上所接的 LED：

```
root@raspberrypi:/sys/class/gpio/gpio25# echo 1 > value
```

10. 输入回车以后，LED 就被点亮了！要把它熄灭，也很简单，只要用 `echo` 命令向 `value` 文件中输出 0 就可以了：

```
root@raspberrypi:/sys/class/gpio/gpio25# echo 0 > value
```



虚拟文件

在这个实验中所操作的很多“文件”其实并不真实存在于 Raspberry Pi 的 SD 卡上，它们是 Linux 虚拟文件系统的一部分。Linux 虚拟文件系统提供了用户模式下操作系统底层硬件的一种简易的接口。其实，也可以通过向 Raspberry Pi 内存的一个指定位置写入一些特定的值来点亮或熄灭 LED，但是这比使用虚拟文件系统操作要复杂得多，也更容易出错。

既然通过写入文件可以控制向 GPIO 接口输出信息，那应该如何从 GPIO 读入数据呢？也许你会猜，是不是该通过“读取文件”呢？完全正确，下面我们就来完成这项工作。

数字信号输入：读取按钮状态

如图 7.5 所示，使用简单的按钮开关来产生数字输入信号非常合适，它们可以稳妥地插入到面包板上。

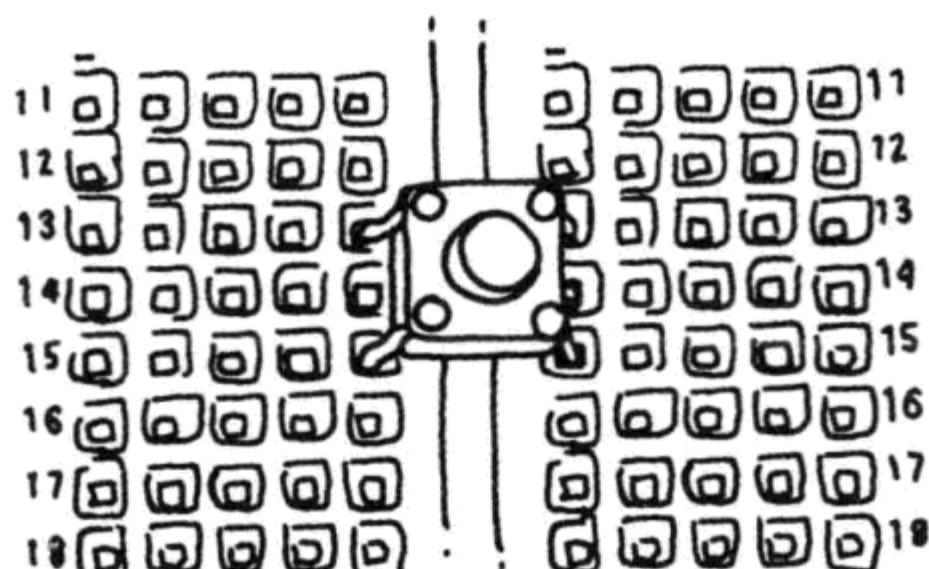


图7.5 按 钮



这类按钮开关在电子实验中很常用，了解它们的内在构造可以更好地帮助你设计电路。观察面包板上这些按钮（图 7.5）：上部的两个引脚之间永远相通，下部的两个引脚亦然。当你把按钮按下的时候，这两组引脚之间就被连通了。

当从 Raspberry Pi 的 GPIO 接口上读取数据时，实际上你是在检查这个接口是被接在 3.3V 的电源上还是被接地了。必须牢记，输入接口必须连接一个确定的信号（如 3.3V 电源或者接地），如果尝试读取一个既未接电源又未接地的引脚，你会得到一个不稳定的结果。当学会了按钮的工作原理，你可以试着将磁控开关、游戏手柄，甚至自动售货机上的硬币识别器作为数字输入信号连接到 GPIO 接口上。

1. 把按钮插入面包板，并使它横跨在面包板中央绝缘条的两边。
2. 使用连接线，把 Raspberry Pi 的 GPIO 24 与按钮上部的一个引脚相连。
3. 把 Raspberry Pi 的 3.3V 供电口与面包板上供电总线的正极相连。



注意：需要输入高电平时，只能把 3.3V 的电源与按钮相接，不能接 5V 的电源。向 Raspberry Pi 的 GPIO 接口输入大于 3.3V 的电压将对会彻底烧坏 Raspberry Pi。

4. 把按钮下部的一个引脚接到供电总线的正极上，这时，当你按下按钮时，3.3V 的电压会接入 GPIO 24 接口。
5. 还记得我们说过要保证数字输入接口应该是接 3.3V 电源或



接地吗？在目前的情况下，你没有按下按钮，那么 GPIO 24 是没有接到电源或接地的，因此，它是悬空的。这种情况会带来不稳定的数据读取结果，所以必须解决这个问题。使用一个 $10k\Omega$ 的电阻（电阻色环为：棕、黑、橙^①，最后一环为银色或金色），把按钮靠近输入的一端用电阻与 Raspberry Pi 的地线相接，这样当你没有按下按钮时，Raspberry Pi 的 GPIO 是通过电阻接地的。

由于电流总是会往接地电阻最小的通路流动，当你按下按钮时，3.3V 的电压会接入 Raspberry Pi 的输入接口，因为这条通路的电阻小于 $10k\Omega$ 。

当所有的准备工作都完成时，整个电路看上去应该如图 7.6 所示。

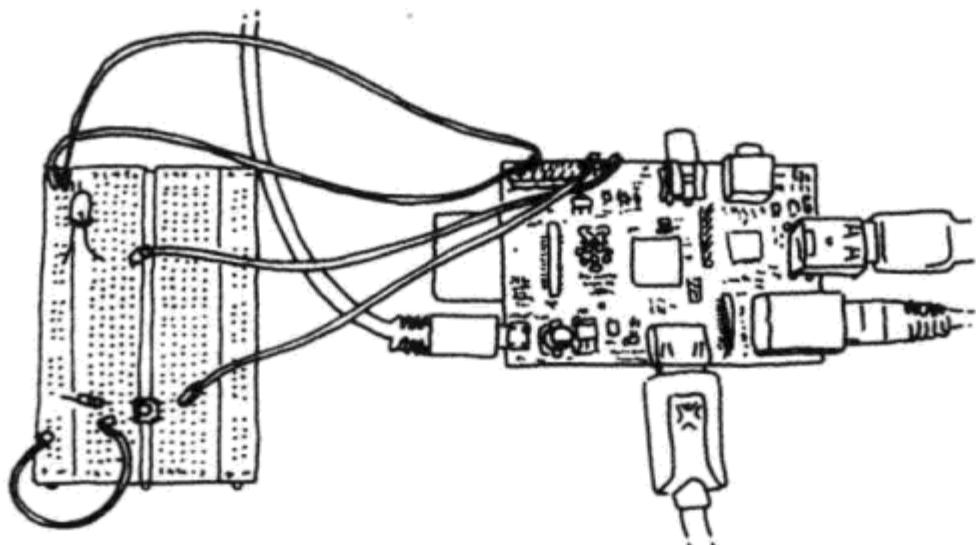


图7.6 把按钮与Raspberry Pi相连

6. 电路已经连接好了，这时我们就可以从命令行上读取按钮的状态了。如果你还没有取得 root 权限，就先执行 `sudo su`。
7. 与前一个例子一样，第一步先要把输入接口暴露到用户态：

```
root@raspberrypi:/sys/class/gpio/gpio24# echo 24 > /sys/class/gpio/export
```

^① 现在市面上有些电阻采用 5 色环标注阻值， $10k\Omega$ 电阻对应的前 4 个色环是棕、黑、黑、棕，第 5 个色环表示阻值精度。——译者注



8. 把工作目录切换到上一步操作所生成的目录中：

```
root@raspberrypi:/sys/class/gpio/gpio25# cd /sys/class/gpio/  
gpio24
```

9. 设置接口的工作模式为输入：

```
root@raspberrypi:/sys/class/gpio/gpio24# echo in > direction
```

10. 现在，你可以用 `cat` 命令来读取按钮的状态了，`cat` 命令用于在命令行下显示文件的内容。`cat` 这个命令的命名来源于它可以用 来把多个文件拼接 (Concatenate) 在一起，它也可以用于显示文件内容。

```
root@raspberrypi:/sys/class/gpio/gpio24# cat value  
0
```

11. 显示的 `0` 表示这个接口目前是接地的。你可以按住按钮不放，同时再次执行这个命令：

```
root@raspberrypi:/sys/class/gpio/gpio24# cat value  
1
```

12. 如果你看到它显示了 `1`，就表示你的电路已经正常工作了！



如果需要执行一个以前执行过的命令，可以通过不断地按光标向上键查询执行过的命令，找到你想要的命令时，按回车键执行。

现在你已经学会了如何通过 Linux 的命令行来控制 LED 或读取按钮的状态，下面让我们一起来用一些常用的 Linux 内置工具操作数字输入输出接口来完成一个简单的项目。



项目：定时台灯

假设明天一早你需要离开家去外地度假，希望你不在家的这段时间里，不要吸引小偷来光顾。用一个定时开关定时控制家里的台灯可以实现这个目的，可是附近的电器商店已经关门了，明天早上飞机起飞前你也没有足够时间去商店购买现成的定时开关。还好，你是 Raspberry Pi 的爱好者，你手头有一些实用的器件：

- Raspberry Pi 主板；
- 面包板；
- 连接线；
- PowerSwitch Tail II；
- 电线。

有了这些器材，你就可以自己动手编写一个定时开关，需要使用两个强大的 Linux 工具：Shell 脚本和 cron。

脚本命令

所谓 Shell 脚本，是指包含了很多 Shell 命令（如我们在上一节中用来控制 GPIO 接口的命令）的文件。看看下面的 Shell 脚本和我们对关键行的解释。

```
#!/bin/bash #❶
echo Exporting pin $1. #❷
echo $1> /sys/class/gpio/export #❸
echo Setting direction to out.
echo out > /sys/class/gpio/gpio$1/direction #❹
echo Setting pin high.
echo 1 > /sys/class/gpio/gpio$1/value
```



- ① 所有的 Shell 脚本都以这一行开头。
- ② \$1 指代第一个命令行参数。
- ③ 这个脚本不是固定地暴露某个特定的 GPIO 接口，而且通过命令行参数来指定接口号。
- ④ 注意，这里的 GPIO 接口号也会被命令行参数替换。

把这个文本文件存为 `on.sh`，并通过 `chmod` 命令把它设置为可执行：

```
root@raspberrypi:/home/pi# chmod +x on.sh
```



你仍然需要通过 `root` 权限来执行这个脚本。如果你在执行脚本时遇到“Permission denied.”这样的出错提示，就先执行一下 `sudo su`。

命令行参数是一种向程序或脚本传递信息的方式，通过在命令后面加上参数，程序或脚本就可以在运行时获取到相应的值。当你在编写一个 Shell 脚本时，`$1` 代表了第一个命令行参数，`$2` 代表了第二个命令行参数，以此类推。在上面 `on.sh` 的例子中，你可以输入想要暴露和打开的 GPIO 接口编号（25）作为命令行参数。通过命令行参数来指定接口编号，而不是在 Shell 脚本中硬编码（Hard-code）具体的接口编号，可以让我们的 Shell 脚本变得更加通用。当你需要对 GPIO 25 进行操作时，只需执行：

```
root@raspberrypi:/home/pi/# ./on.sh 25
Exporting pin 25.
Setting direction to out.
Setting pin high.
```

- ① 文件名前的 `./` 表示你正在执行当前工作目录下的脚本。



如果你的 GPIO 25 上仍然连接着 LED，脚本执行后，它应该会被点亮。我们还可以编写另一个名为 *off.sh* 的脚本，用来把 LED 熄灭。这个脚本看起来应该是如下的样子：

```
#!/bin/bash
echo Setting pin low.
echo 0 > /sys/class/gpio/gpio$1/value
echo Unexporting pin $1
echo $1> /sys/class/gpio/unexport
```

为它加上可执行权限并执行它：

```
root@raspberrypi:/home/pi/temp# chmod +x off.sh
root@raspberrypi:/home/pi/temp# ./off.sh 25
Setting pin low.
Unexporting pin 25
```

如果一切正常，先前被点亮的 LED 现在会熄灭。

连接台灯

显然一个小小的 LED 所发出的光线不足以欺骗一个小偷让他误以为你在家中，所以下一步我们需要把台灯与 Raspberry Pi 连接起来。

1. 断开 LED 与 GPIO 25 的连接。
2. 从面包板上引出两条接线，分别引自 Raspberry Pi 的 GPIO 25 与地线。
3. 把从 GPIO 25 上引出的接线，接到 PowerSwitch Tail 上标有“+in”标记的引脚上。
4. 把从地线引出的接线，接到 PowerSwitch Tail 上标有“-in”标记的引脚上。
5. 把 PowerSwitch Tail 插到插座上，并把台灯插到 PowerSwitch Tail 上。请注意，需要把台灯的电源开关打开。



6. 这时, 如果执行 `./on.sh 25`, 台灯就会被打开; 如果执行 `./off.sh 25`, 台灯就会被关闭。

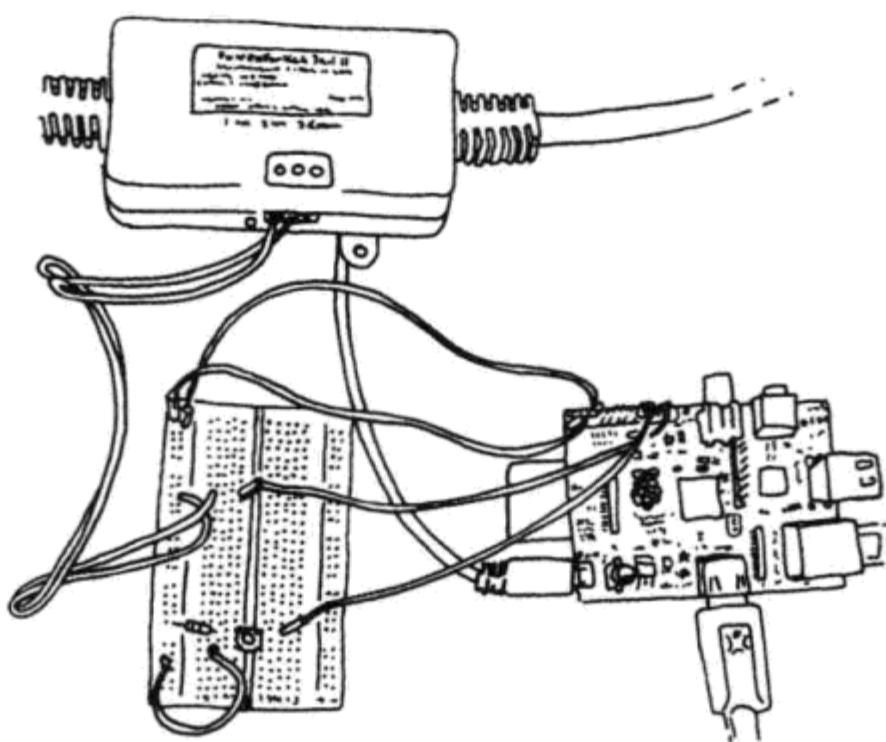


图7.7 连接PowerSwitch Tail与Raspberry Pi

在 PowerSwitch Tail 内部, 有一系列电子元件可以帮助你实现用低电压信号 (如来自 Raspberry Pi 的 GPIO 输出信息) 去控制类似于台灯或电动搅拌机这样的高电压电器。当 PowerSwitch Tail 接通电器电源时, 你可以听到“嗒”一声, 这个声音来自它内部的核心器件——继电器。继电器相当于是高电压电器的开关, 它的开关状态可以由 Raspberry Pi 的低电压信号来控制。

用 cron 设置定时任务

现在你已经把控制 GPIO 接口开关的命令整合成了两个简单的脚本, 台灯也已经通过 PowerSwitch Tail 连接到了 Raspberry Pi 并且可以用单条的命令来控制它的开或关。下面就可以通过 cron 来设置每天定时开关灯的时间了。cron 是 Linux 中用于定时执行任



务的程序。通过使用 **cron**，你可以设置在指定的日期或时间执行指定的命令，或者以指定的时间间隔（如一小时）来执行命令。在我们的项目中，我们可以设定两个定时任务：每天晚上8点打开台灯，每天凌晨2点关闭台灯。



与其他时间相关的程序一样，使用 **cron** 前，需要确保你的 Raspberry Pi 已经按照第 2 章中所描述的方法，设定了正确的日期、时间。

要添加这些定时任务，我们需要修改 Linux 的 **cron** 表，**cron** 表中包含了一系列要求 Linux 在指定时间执行的命令^①：

```
root@raspberrypi:/home/pi/# crontab -e
```

这条命令会打开一个文本编辑器，用于修改 **root** 用户的 **cron** 表。在这个文件的顶端，你能看到一些关于如何修改 **cron** 表的描述。用光标移动键把光标移动到文件的末尾，然后添加下面两行代码：

```
0 20 * * * /home/pi/on.sh 25
0 2 * * * /home/pi/off.sh 25
```



cron 会忽略任何以井号 (#) 开头的配置行。如果你需要临时禁用某一个定期任务但又不想删除它，可以

^① Linux 下每个用户都有属于自己的 **cron**，互不干涉，每个用户 **cron** 表中的任务会以该用户身份所对应的权限运行。由于运行 **on.sh** 和 **off.sh** 脚本都需要 **root** 权限来操作 GPIO 接口，所以 **crontab -e** 这条命令需要在 **root** 权限下执行才能修改 **root** 用户的 **cron** 表，使用 **root** 用户的 **cron** 表才能保证在定时时间到达时，是以 **root** 的权限来运行相应的脚本。——译者注



在这个任务前添加一个井号。井号开头的行也可以作为 cron 表中的注释。

按 Control-X 键退出，在提示你是否要保存文件时输入 y，并按回车键接受它提示的默认文件名。当系统把这个文件保存好并退回命令行状态时，屏幕上会显示“installing new crontab”，表示你对 cron 表所做的修改会被 cron 程序运行。

更多有关 cron 的知识

cron 允许你设置在指定日期时间或按指定间隔执行的任务。通过用空格隔开的 5 个时间字段值（如果要指定年份，就会有 6 个字段）以及要执行的命令就可以设定任务。某个字段值上的星号 (*) 表示这个任务在这个时间间隔上每次都要执行，如表 7.1 所示。

表7.1 每天晚上08:00开灯的cron配置

0	20	*	*	*	/home/pi/on.sh 25
分钟 (:00)	小时 (晚上 08 点)	每天	每月	一周中的 每一天	要执行的命令

如果只想在每周的周一至周五的晚上开灯，可以按表 7.2 设置 cron 表。

表7.2 每个工作日晚上08:00开灯的cron配置

0	20	*	*	1-5	/home/pi/on.sh 25
分钟(:00)	小时 (晚上 08 点)	每天	每月	周一 ~ 周五	要执行的命令

如果想写个 Shell 脚本定期检查你是否收到新的信件，有就通



过电子邮件通知你，你可以用表 7.3 中的办法来设置一个每 5min 运行一次的任务。

表7.3 每5min检查一次邮件的cron配置

*/5	*	*	*	*	/home/pi/checkMail.sh
每 5min	每小时	每天	每月	一周中的 每一天	要执行的命令

其中， */5 就表示“每 5min”。

由此可见， cron 是一个非常强大的工具，你可以用它来指定在特定的日期或时间，或以特定的时间间隔来定期执行任务。

进一步学习

eLinux 的 Raspberry Pi GPIO 参考页面

(http://elinux.org/RPi_Low-level_peripherals)

这个页面上提供了有关 Raspberry Pi 的 GPIO 接口最为完整全面的信息。

Adafruit: 通过 MCP230xx 扩展 Raspberry Pi 的 GPIO 接口

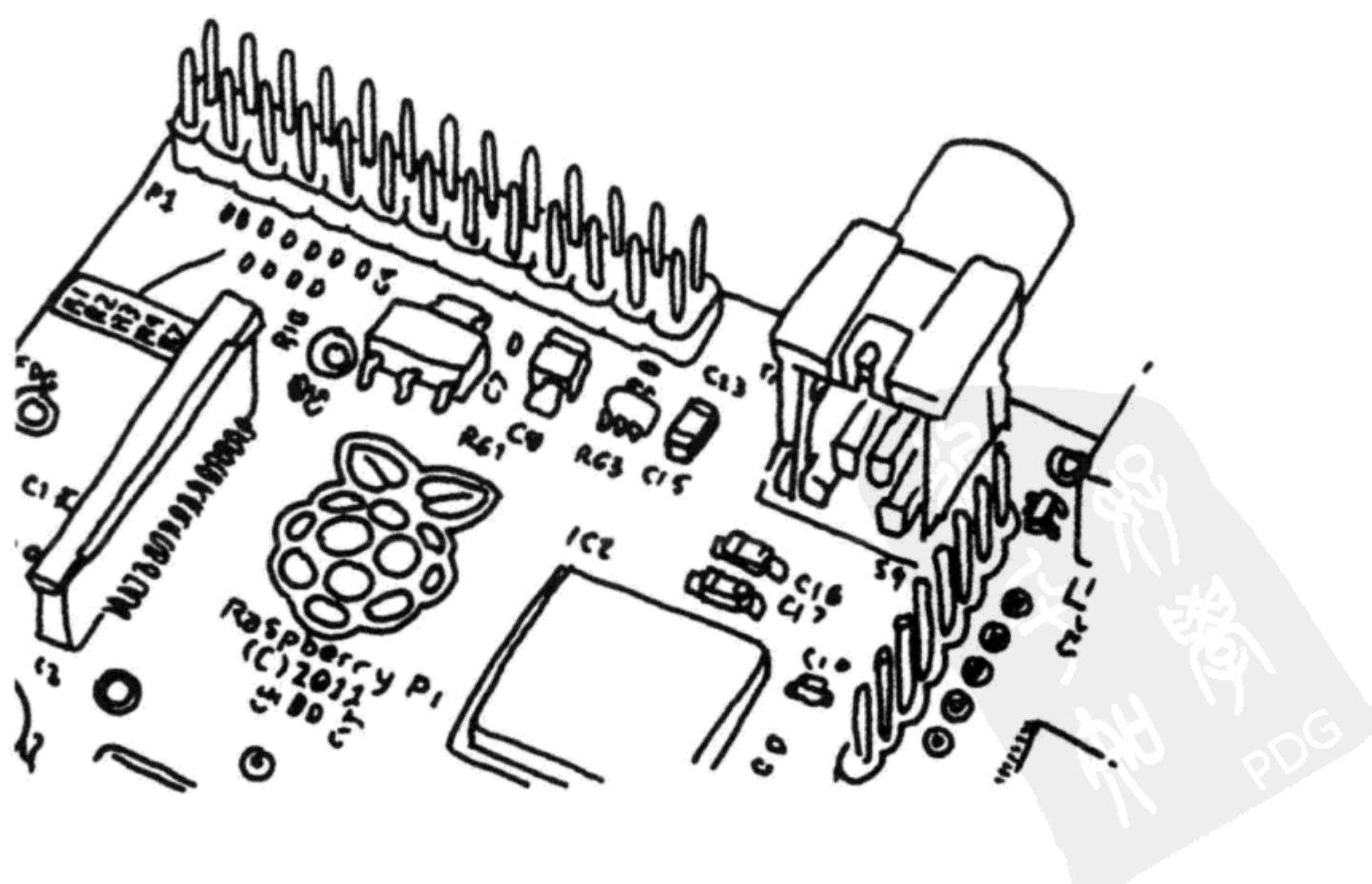
(<http://learn.adafruit.com/mcp230xx-gpio-expander-on-the-raspberry-pi>)

如果你觉得 Raspberry Pi 上的 GPIO 接口不够用，Adafruit 提供了如何通过 MCP23008 芯片增加 8 个额外的 GPIO 接口或用 MCP23017 增加 16 个额外的 GPIO 接口的方法。

第 8 章

用Python进行 输入输出编程

Programming Inputs and Outputs with Python





在第 7 章的最后，我们通过 Shell 脚本进行了一些简单的 Raspberry Pi GPIO 接口的编程。在这一章中，我们会学习如何用 Python 进行 GPIO 编程。与 Shell 脚本一样，使用 Python 也可以通过编写代码去访问和操作 GPIO 接口。

用 Python 而不是 Shell 脚本的好处在于，Python 脚本更容易编写，可读性也更强。有一些封装好的 Python 模块可以让你只用很简单的代码就能完成较为复杂的操作。表 3.2 中列出了一些有用的 Python 模块，通过使用其中的 `raspberry-gpio-python` 模块 (<http://code.google.com/p/raspberry-gpio-python/>) 可以非常方便地访问和控制 GPIO 接口。在本章中，你会学到如何去使用这个模块。

在 Python 中安装并测试 GPIO

在最新的 Raspbian 系统中已经预装了 GPIO 模块。如果还在使用较早版本的 Raspbian，你需要自行安装。可以通过 Python 的交互式解释器（参考第 3 章中的内容，交互式解释器允许你直接输入单行 Python 代码并立即运行，不需要先把代码写入文件再执行）来验证你是否已经安装了这个模块。

1. 从命令行以 `root` 身份启动 Python 的交互式解释器（因为 `raspberry-gpio-python` 需要 `root` 权限来操作 GPIO 接口，所以在启动 Python 的交互式解释器时，需要在前面添加 `sudo` 命令）。



```
pi@raspberrypi ~ $ sudo python
Python 2.7.3rc2 (default, May 6 2012, 20:02:25)
[GCC 4.6.3] on linux2
Type "help", "copyright", "credits" or "license" for more
information.
>>>
```

2. 在 >>> 提示符后面，尝试导入这个模块：

```
>>> import RPi.GPIO as GPIO
```

3. 如果没有出错，证明这个模块已经正确安装了。

如果在导入 GPIO 模块时出错了，可以通过 Raspberry Pi 的软件包管理器 `apt-get` 命令很容易地安装这个模块。

如果你的系统中没有安装 `raspberry-gpio-python`，可以用下面的步骤来安装。

1. 退出 Python 解释器（按 Control-D 或输入 `exit()` 并回车），更新 `apt-get` 软件包列表，然后执行安装命令来安装 `raspberry-gpio-python` 包：

```
>>> exit()
pi@raspberrypi ~ $ sudo apt-get update
pi@raspberrypi ~ $ sudo apt-get install python-rpi.gpio
```

2. 安装完成后，重新运行 Python 的交互式解释器并导入模块。

```
pi@raspberrypi ~ $ sudo python
Python 2.7.3rc2 (default, May 6 2012, 20:02:25)
[GCC 4.6.3] on linux2
Type "help", "copyright", "credits" or "license" for more
information.
>>> import RPi.GPIO as GPIO
>>>
```



在本章中，我们使用 Python 2.7。不使用 Python 3 的原因是，我们所用到的一些模块在 Raspberry Pi 上默认只安装在 Python 2.x 的环境中。当你在 Raspberry Pi 的命令行上运行 `python` 命令时，目前的行为是默认运行 Python 2.7，但以后可能会改成 Python 3（任何时候，你都可以显式地输入 `python2.7` 而不是 `python` 命令来运行 Python 2.7）。

Python 2.7 与 Python 3 的最明显的区别之一是向屏幕输出文本的方式：在 Python 2.x 中使用 `print "Hello, World!"` 这样的形式，而在 Python 3 中使用 `print("Hello, Wrold!")`。

如果输入 `import` 命令后没有出错，你就可以继续下面的实验了。

1. 在使用 GPIO 接口前，先要告诉 GPIO 模块你打算以何种方式来指代 GPIO 接口。在第 7 章中，我们所使用的 GPIO 接口编号与它们在主板上的引脚排列没有直接的关系，本质上是在使用 Broadcom 芯片的引脚编号。使用 GPIO 的 Python 模块时，你可以使用芯片引脚编号也可以使用主板引脚编号来访问 GPIO 接口。如果要使用主板引脚编号，使用 `GPIO.setmode(GPIO.BCM)` 命令来设置 GPIO 模块。不过在本章中，我们还是使用与第 7 章中一样的芯片引脚编号（`GPIO.setmode(GPIO.BCM)`）来操作 GPIO 接口，这种编号方式也是 Adafruit Pi Cobbler 等套件板上标注引脚编号的方式：

```
>>> GPIO.setmode(GPIO.BCM)
```

2. 把 GPIO 25 设置为输出状态：

```
>>> GPIO.setup(25, GPIO.out)
```



3. 把 LED 接到 GPIO 25 接口（与“使用输入输出接口”一节中的做法一样）。

4. 点亮 LED：

```
>>> GPIO.output(25, GPIO.HIGH)
```

5. 熄灭 LED：

```
>>> GPIO.output(25, GPIO.HIGH)
```

6. 退出 Python 交互式解释器：

```
>>> exit()  
pi@raspberrypi ~ $
```



在第 7 章，我们强调过 Raspberry Pi 的数字输入输出接口信号应该是 3.3V 或接地。在电子学中，我们把这些信号称为高电平或低电平。请记住，不是所有的电路都使用 3.3V 作为高电平信号，有些电路会使用 1.8V 或 5V。如果你要把 Raspberry Pi 与其他数字设备通过 GPIO 接口相连，必须确认那些设备也是以 3.3V 作为工作电压。

以上步骤让你初步了解了如何在 Python 的交互式解释器中用单行 Python 命令来控制 GPIO 接口。在第 7 章中，我们使用 Shell 脚本来操作 GPIO 接口，下面我们也会用 Python 脚本来自动读取或控制这些 GPIO 接口。

让 LED 闪烁

要通过 Python 让 LED 闪烁起来，你需要使用先前在交互式解



释器中实验过的命令和另外一些命令。在下面的操作步骤中，我们假设你使用的是桌面环境（图 8.1）。但如果你喜欢用命令行来编写和运行 Python 脚本的话，也完全没有问题。

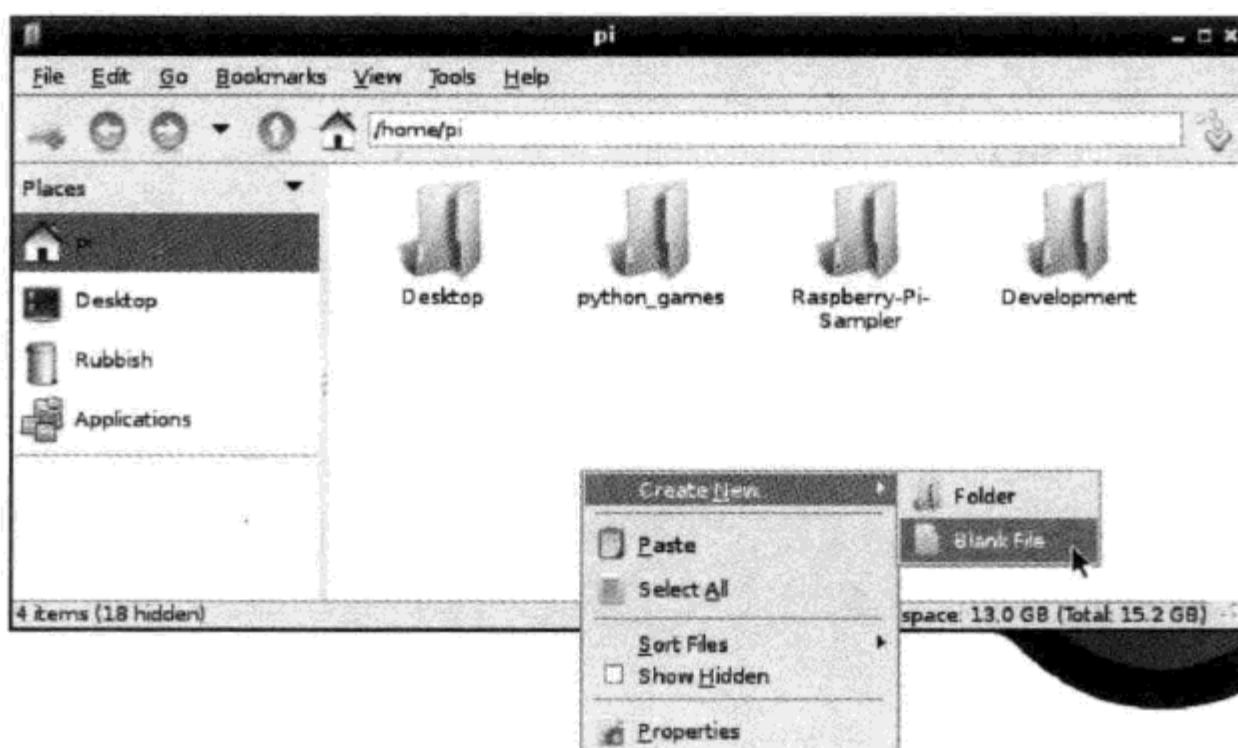


图8.1 在用户主目录中创建一个新文件

1. 通过任务栏上的按钮打开文件管理器（File Manager）。
2. 确认当前目录是你的主目录（默认为 `/home/pi`）。如果不是，点 Places 列表下的主目录图标。
3. 在你的主目录中创建一个文件，命名为 `blink.py`。操作方法是在你的主目录窗口中点击鼠标右键，选 Create New… 然后选 Blank File。把这个文件命名为 `blink.py`。
4. 双击 `blink.py`，通过默认的 Leafpad 文本编辑器打开它。
5. 输入以下的代码并保存文件：

```
import RPi.GPIO as GPIO①
import time②

GPIO.setmode(GPIO.BCM)③
GPIO.setup(25, GPIO.OUT)④
```



```
while True:①
    GPIO.output(25, GPIO.HIGH)②
    time.sleep(1)③
    GPIO.output(25, GPIO.LOW)④
    time.sleep(1)⑤
```

- ① 导入控制 GPIO 所需的代码。
- ② 导入 sleep 函数所需的代码。
- ③ 设置使用芯片引脚编号。
- ④ 把 GPIO 25 接口设置为输出模式。
- ⑤ 开始一个无限循环，执行下面缩进的代码。
- ⑥ 点亮 LED。
- ⑦ 暂停 1s。
- ⑧ 熄灭 LED。
- ⑨ 暂停 1s。



请牢记：在 Python 中，代码的缩进位置非常重要。

-
6. 打开 LX 终端（LXTerminal），执行下面的代码把当前工作目录切换到你的主目录，并执行刚才输入的脚本：

```
pi@raspberrypi ~/Development $ cd ~
pi@raspberrypi ~ $ sudo python blink.py
```

7. 你的 LED 应该开始闪烁了！
 8. 按 Control-C 中断脚本的运行并返回命令行。
- 你可以尝试在 `time.sleep()` 函数中指定小于 1 的小数使 LED 闪烁得更快。你也可以尝试连接更多的 LED，并让它们以特



定的模式来闪烁。你可以参考图 7.2 使用以下独立的 GPIO 接口中的任意一些：4, 17, 18, 21, 22, 23, 24 或 25。

读取按钮状态

如果你想在按钮按下时触发特定的操作，实现这个目标的方法之一称为轮询。轮询的意思就是不断地检查某一个状态，在我们的例子中就是指不断检查相关的 GPIO 接口是被连接到了 3.3V 还是接地。下面的实验可以实现在按下按钮时屏幕上显示一些文本。

1. 用“数字信号输入：读取按钮状态”（第 7 章）一节中相同的方法连接一个按钮，使用 GPIO 24 接口作为输入。别忘了在输入接口与地线之间加上下拉电阻。
2. 在你的主目录中创建一个新文件，名为 *button.py*，在文本编辑器中打开它。
3. 输入下面的代码：

```
import RPi.GPIO as GPIO
import time

GPIO.setmode(GPIO.BCM)
GPIO.setup(24, GPIO.IN)①

count = 0②

while True:
    inputValue = GPIO.input(24)③
    if (inputValue == True):④
        count = count + 1⑤
        print("Button pressed " + str(count) + " times.")⑥
    time.sleep(.01)⑦
```

- ① 把 GPIO 24 接口设置为输入接口。
- ② 创建一个名为 `count` 的变量，并把值设为 0。



- ③ 把 GPIO 24 接口上读到的状态存入 `inputValue` 变量。
- ④ 检查读到的值是否为 `True` (意味着按钮被按下)。
- ⑤ 如果按钮被按下, 计数器 `counter` 加 1。
- ⑥ 在屏幕上显示文字。
- ⑦ 稍稍暂停一会儿, 释放处理器的计算资源, 使得其他程序可以有机会被执行到。

4. 回到 LX 终端 (LXTerminal), 运行这个脚本:

```
pi@raspberrypi ~ $ sudo python button.py
```

5. 按动按钮。如果你的电路和程序一切都正常, 每次屏幕上都会显示出几行“Button pressed * times” (按钮被按下了*次)的提示。

在这个例子中, 程序 1s 内会检测 100 次按钮的状态, 所以你每按一下按钮都可能显示出多行信息来 (除非你的动作快到无与伦比)。代码中的 `time.sleep(.01)` 决定了多久检测一次按钮的状态。

为什么不能不间断地检测按钮状态呢? 如果你把代码中的 `time.sleep(.01)` 这行去除, 检测循环会更快地运行, 所以当按钮按下时你也可以更快地知道。这样做有一些缺点: 你会占用过多的处理器时间, 其他程序就难以正常工作, 同时也会增加 Raspberry Pi 的功耗。因为 `button.py` 与其他程序一起共享 Raspberry Pi 的硬件资源, 所以你要保证不能让一个程序消耗完所有的资源。

添加一些代码, 可以让程序更好地检测单次按钮动作:

```
import RPi.GPIO as GPIO
import time

GPIO.setmode(GPIO.BCM)
GPIO.setup(24, GPIO.IN)

count = 0
```



```
while True:
    inputValue = GPIO.input(24)
    if (inputValue == True):
        count = count + 1
        print("Button pressed " + str(count) + " times.")
        time.sleep(.3)①
    time.sleep(.01)
```

① 更好地处理单次按钮动作。

新添加的代码可以更好地保证每次按动按钮只被记录一次，但这不是一个完美的解决方案，如果你按住按钮不放，那 1s 内会被检测成 3 次按钮动作。如果你快速按动按钮，有几次动作就不会被检测到，因为它无法检测 1s 内 3 次以上的按钮动作。

这些都是通过轮询的方式检查数字输出接口状态时会遇到的问题。解决这些问题的一个常见思路是使用中断。中断是指在硬件检测到某个引脚电平状态变化时，回调一段特定的代码的方式。目前，RPi.GPIO 模块中对中断的支持还处于实验性阶段^①，可以参考这个模块的文档（<http://pypi.python.org/pypi/RPi.GPIO>）来了解如何使用这个功能。

项目：简易发音板

我们已经学习了如何在 Raspberry Pi 上读取输入接口的状态，现在你可以用 Python 的 Pygame 模块中的 sound 函数来制作一块发音板。所谓发音板，是指在你按动上面的按钮时可以播放一段指定的声音的设备。为了制作发音板，你需要以下的材料：

- 3 个按钮开关；
- 母头对公头的连接线；

^① 0.5.0a 以后版本的 RPi.GPIO 模块已经正式支持中断，可以使用 `wait_for_edge()` 函数进行边缘检测或用 `add_event_detect()` 和回调函数实现中断处理。——译者注



- 适当长度的连接线；
- 面包板；
- 3 个 $10k\Omega$ 的电阻；
- 音箱，或者具备 HDMI 接口并内置音箱的显示器。

你还需要几个未压缩的.wav 格式的声音文件。Raspberry Pi 的系统中自带了几个声音文件，你可以直接用来测试。只要你的发音板可以正常工作了，把这个声音文件替换成你所想要的声音是一件很容易的事情，当然，你也许需要先把它们从其他格式转换为.wav 格式。下面先来搭建电路。

1. 用母头对公头连接线把 Raspberry Pi 的接地接口与面包板的电源总线负极相连。
2. 用母头对公头连接线把 Raspberry Pi 的 3.3V 电源接口与面包板的电源总线正极相连。
3. 在面包板上安装 3 个按钮开关，横跨在面包板的中央绝缘条两边。
4. 用连接线把电源负极与按钮上部的引脚相接。
5. 安装下拉电阻，用 $10k\Omega$ 的电阻把按钮下部的引脚与接地相连。
6. 用母头对公头连接线把每个按钮下部的引脚（与 $10k\Omega$ 电阻相连的那个）与 Raspberry Pi 的 GPIO 接口相连。在这个项目中，我们使用 GPIO 23、24 和 25 号接口。

图 8.2 展示了完成后的电路图。我们使用 Fritzing 软件来制作这个图片，Fritzing (<http://fritzing.org>) 是一个用于硬件设计的开源软件。

把电路搭好后，就该开始编写代码了。

1. 在你的主目录中创建一个新的目录，名为 *soundboard*。
2. 打开这个目录并在里面创建一个名为 *soundboard.py* 的文件。

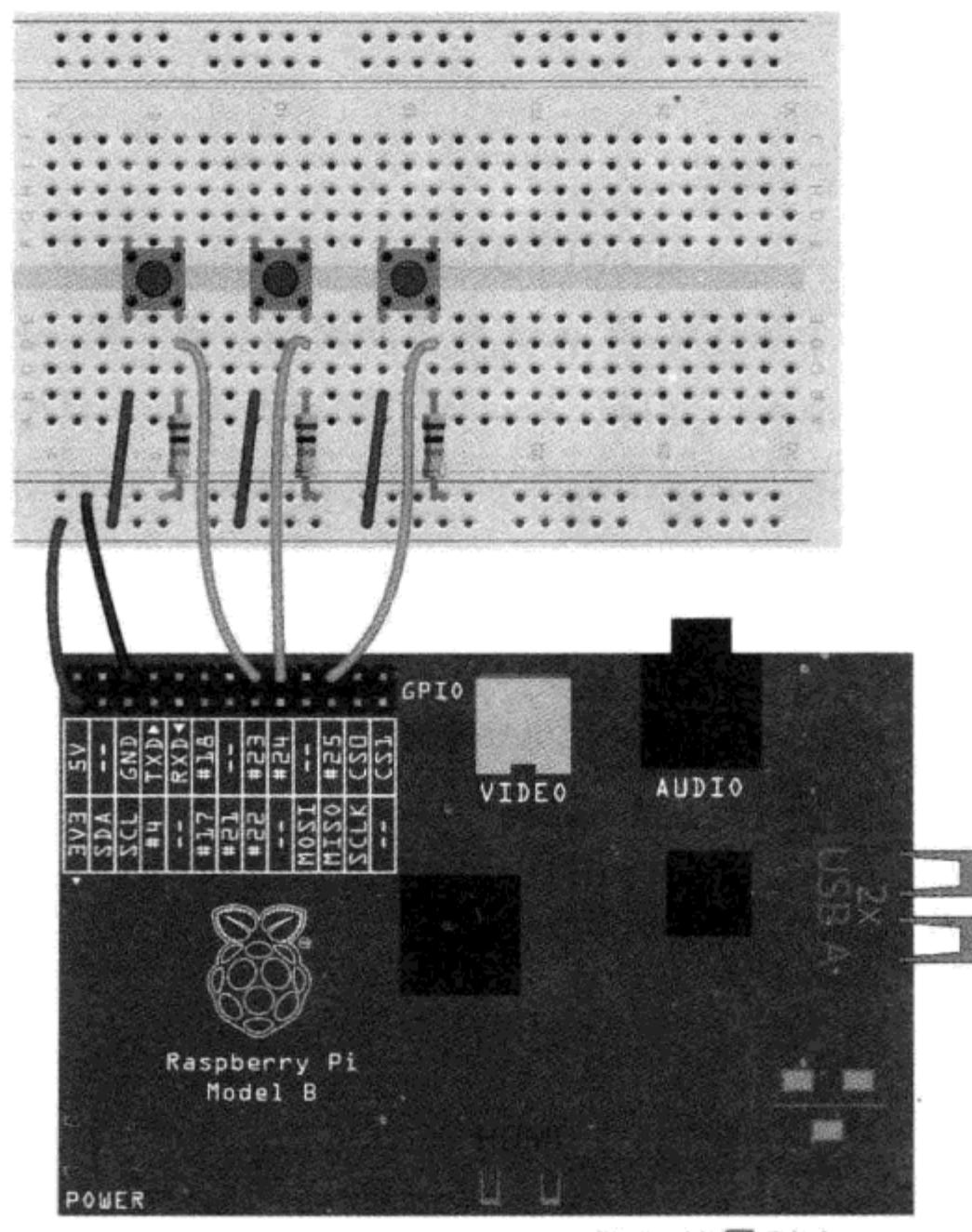


图8.2 发音板项目完成后的电路图

3. 打开 *soundboard.py* 并输入下面的代码：

```
import pygame.mixer
from time import sleep
import RPi.GPIO as GPIO
from sys import exit

GPIO.setmode(GPIO.BCM)
GPIO.setup(23, GPIO.IN)

GPIO.setup(24, GPIO.IN)
GPIO.setup(25, GPIO.IN)
```



```
pygame.mixer.init(48000, -16, 1, 1024)①
soundA = pygame.mixer.Sound("/usr/share/sounds/alsa/Front_
Center.wav")②
soundB = pygame.mixer.Sound("/usr/share/sounds/alsa/Front_
Left.wav")
soundC = pygame.mixer.Sound("/usr/share/sounds/alsa/Front_
Right.wav")

soundChannelA = pygame.mixer.Channel(1)③
soundChannelB = pygame.mixer.Channel(2)
soundChannelC = pygame.mixer.Channel(3)

print "Soundboard Ready."④

while True:
    try:
        if (GPIO.input(23) == True):⑤
            soundChannelA.play(soundA)⑥
        if (GPIO.input(24) == True):
            soundChannelB.play(soundB)
        if (GPIO.input(25) == True):
            soundChannelC.play(soundC)
        sleep(.01)⑦
    except KeyboardInterrupt:⑧
        exit()
```

- ① 初始化 Pygame 的混音器。
- ② 加载声音文件。
- ③ 设置 3 个通道，每段声音对应一个通道，这样就可以做到同时播放不同的声音。
- ④ 显示提示用户发音板已经启动完毕（这里使用了 Python 2 语法）。
- ⑤ 如果对应接口是高电平，执行下面的代码。
- ⑥ 播放声音。
- ⑦ 检测按钮的间隔不要太短，以保证不会占用太多的处理器资源。



⑧ 这个异常捕获代码可以保证在按下 Control-C 时程序可以正常退出，而不会显示一堆用于调试的栈信息。

4. 到命令行下把当前工作目录切换到 *soundboard.py* 所在的目录，然后用 Python 2 执行这个脚本：

```
pi@raspberrypi ~	soundboard $ sudo python soundboard.py
```

5. 看到屏幕提示“Soundboard Ready”后，按动面包板上的按钮就可以播放相应的声音。



Pygame 有支持 Python 3 的版本，但在 Raspberry Pi 上默认只安装了支持 Python 2 的版本。

根据 Raspberry Pi 设置的不同，声音可能会从 HDMI 接口传到显示器上输出，也可能是通过板载的 3.5mm 模拟音频接口输出。如果改变声音输出的接口，可以先按 Control-C 退出程序，然后在命令行上输入下面的命令把声音设置为从模拟音频接口输出：

```
pi@raspberrypi ~	soundboard $ sudo amixer cset numid=3 1
```

如果要把声音设置为通过 HDMI 接口输出，使用命令：

```
pi@raspberrypi ~	soundboard $ sudo amixer cset numid=3 2
```

当然，默认的那些声音并不是很有意思，你可以把它们替换成任意你喜欢的声音，如掌声、笑声、警报声或铃声。把这些声音对应的文件放入你的 *soundboard* 目录并修改代码让它使用这些文件即可。如果想让你的发音板发出更多种类的声音，也可以通过安装更



多的按钮并修改相应的代码来实现。

进一步学习

RPi.GPIO

(<http://code.google.com/p/raspberry-gpio-python/>)

RPi.GPIO 库还在不断开发中，你可以访问它的主页来了解最新的进展。

Using the MCP3008

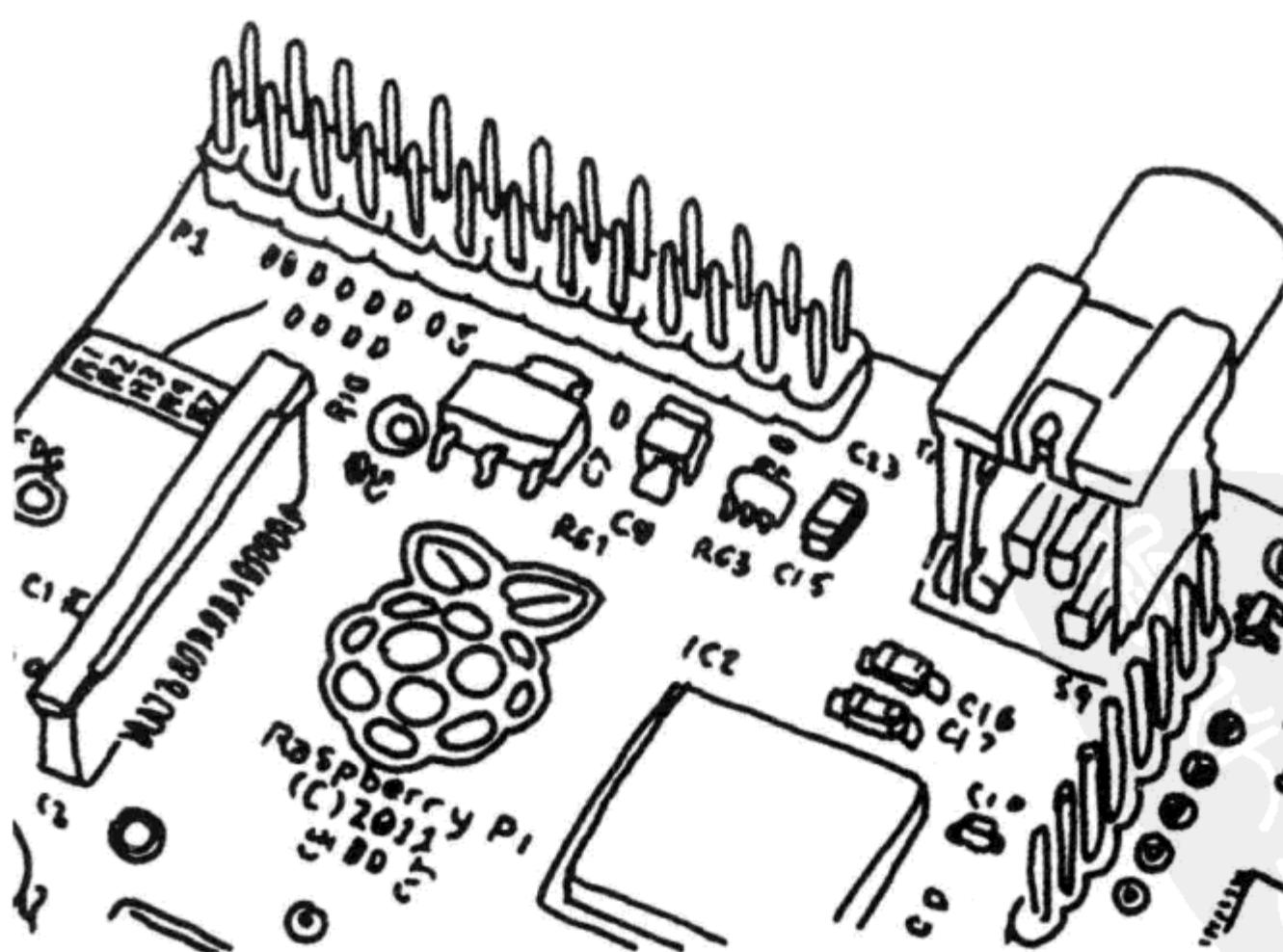
(<http://learn.adafruit.com/reading-a-analog-in-and-controlling-audio-volume-with-the-raspberry-pi/overview>)

Adafruit 的一篇不错的教程，介绍了如何使用 MCP3008 模数转换器在 Raspberry Pi 上使用输出模拟信号的传感器。

第 9 章

使用摄像头

Working with Webcams





使用像 Raspberry Pi 这样的平台开发各种项目有一个很大的优势——可以直接使用各种 USB 设备。你不但可以连接键盘、鼠标，还可以连接打印机、无线网卡、指纹识别器、读卡器、摄像头或移动硬盘等其他外设。在这一章中，我们会演示在 Raspberry Pi 上使用摄像头的一些方法。

虽然不像键盘和鼠标常见，摄像头也已经成为主流电脑的一个标准配置。这对大家来说都是一个好消息，因为这意味着我们可以用不到 25 美元的价格很容易买到一个知名品牌的摄像头，如果买非知名品牌的还可以更便宜。更重要的是，很多 USB 摄像头在 Linux 下都不需要额外的驱动程序就可以正常工作。

也许你还记得在第 1 章介绍主板接口时，我们提到过用于连接摄像头的 CSI 接口（图 9.1）。由于 Pi 所使用的 Broadcom 芯片组原本是为手机和平板电脑设计的，CSI 接口就是这些移动设备厂商

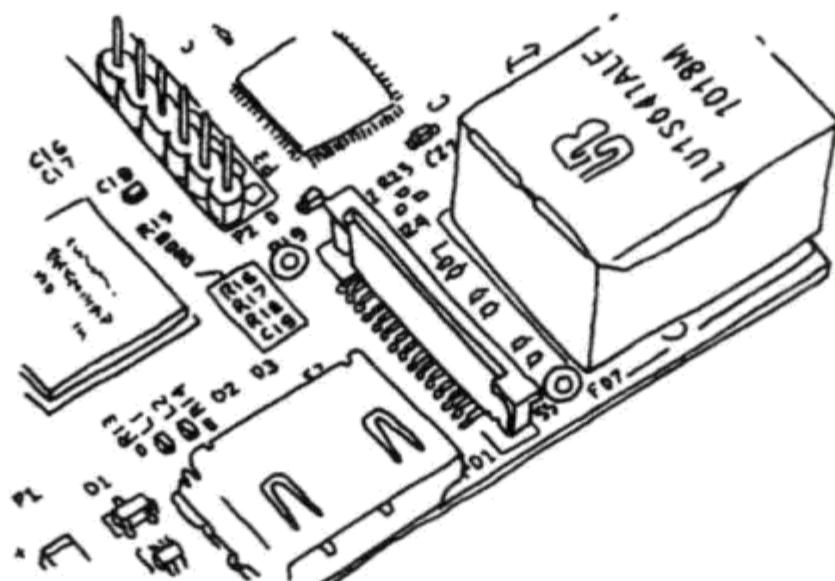


图9.1 Raspberry Pi的CSI接口



用来连接摄像头的接口。遗憾的是，作为普通消费者，CSI 接口的摄像头并不容易直接在商店中买到。截至本书出版时，Raspberry Pi 基金会仍在开发可以直接连接到 CSI 接口的摄像头^①。在这种摄像头面市之前，我们仍然建议先使用 USB 摄像头（图 9.2）。

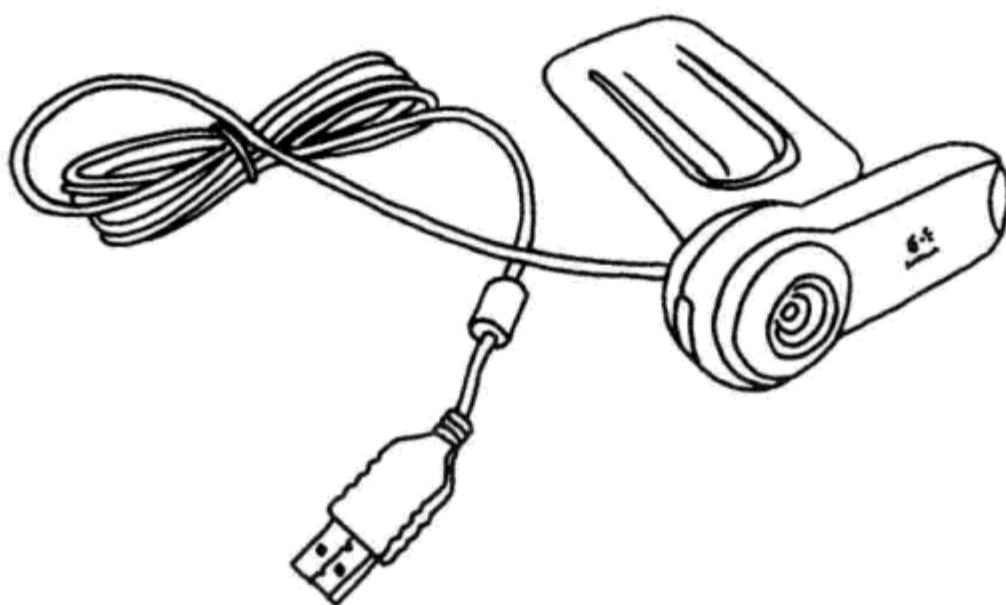


图9.2 USB摄像头

测试摄像头

市面上有很多型号的摄像头，没办法保证每一种都能直接在 Raspberry Pi 上正常工作。如果你准备购买某个型号的摄像头，最好先上网看看有没有他人成功使用这种摄像头的先例。你也可以参考 Raspberry Pi 已知可用的外设列表中摄像头相关的内容 (http://elinux.org/RPi_VerifiedPeripherals#USB_Webcams)。

注意，你需要把摄像头和键盘、鼠标一起通过一个有源的 USB Hub 连接到 Raspberry Pi。必须使用有源的 USB Hub 是因为，Raspberry Pi 的 USB 接口只能提供有限的电流，这点电流很可能不

^① 适用于 Raspberry Pi 的 CSI 接口摄像头已于 2013 年 5 月 14 日正式发布，请参考：<http://www.raspberrypi.org/archives/3890>。——译者注



足以让摄像头和键盘、鼠标一起正常工作。有源的 USB Hub 可以从独立电源获得电流给设备供电，就不再需要由 Raspberry Pi 自身来提供超出它供电能力的电流了。

如果你已经拥有了一个摄像头并准备在 Raspberry Pi 上开展测试，可以在命令行上用 `apt-get` 命令来安装一个称为 `luvcview` 的摄像头图像预览程序：

```
pi@raspberrypi ~ $ sudo apt-get install luvcview
```

用 `apt-get` 完成程序的安装后，在图形化桌面环境的终端中输入 `luvcview` 运行这个程序。`luvcview` 会打开新的窗口显示它从 `/dev` 目录下找到的第一个视频设备（通常是 `/dev/video0`）中采集到的图像。画面的尺寸会在终端上显示出来。如果显示的视频图像上有一些波纹，你可以试着缩小画面的尺寸。例如，如果默认的视频大小是 640×480 ，你可以关闭并重新运行 `luvcview`，在命令行参数中把画面尺寸设置成原来的一半：

```
pi@raspberrypi ~ $ luvcview -s 320x240
```

如果无法看到视频画面，你需要在这时候就把问题原因找到。排查问题的方法之一是，先把摄像头断开再插上，并同时用 `dmesg` 命令观察系统所显示的诊断信息，这些信息也许可以为你提供一些线索。

安装并测试 SimpleCV

我们将在 Python 中使用 SimpleCV 库（图 9.3）来操作摄像头，SimpleCV 是与电脑视觉相关的开源库。通过使用 SimpleCV，我们可以很方便地从摄像头采集图像并显示在屏幕上或保存为文



件，但 SimpleCV 最强大的功能在于它所包含的计算机视觉相关的算法，可以用来实现一些让人惊叹的效果。除了基本的图像变换，SimpleCV 还可以用来跟踪、检测和识别图像或视频中的特定对象。后面我们会完成一个使用 SimpleCV 实现的人脸检测程序，见“人脸检测”一节。

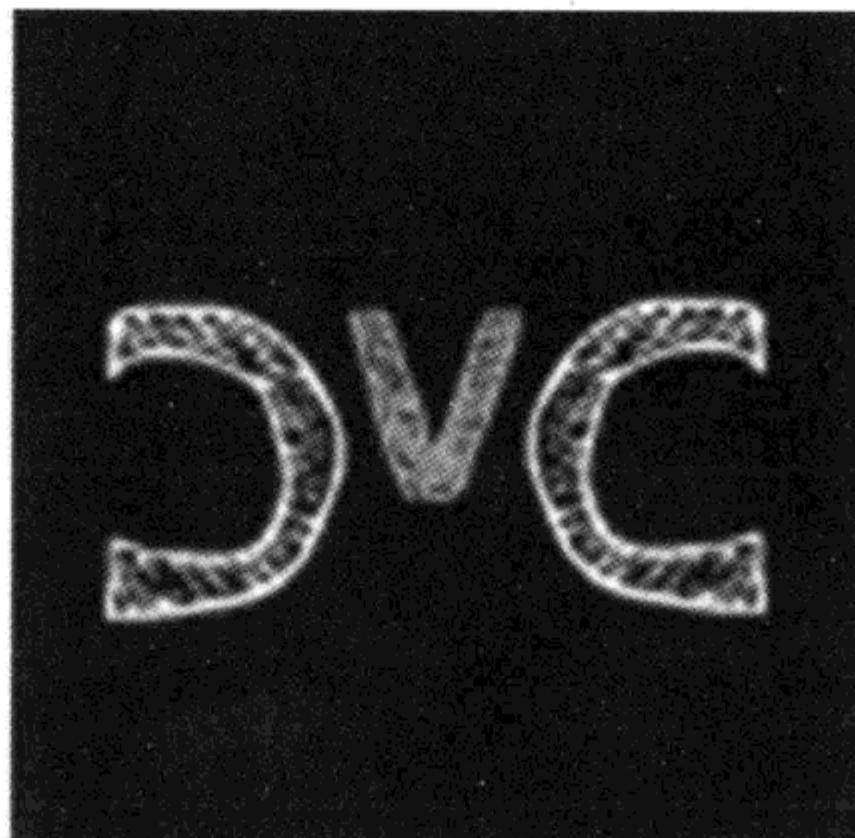


图9.3 SimpleCV的标识

在安装 SimpleCV 前，先要安装一些它所依赖的库，可以用 `apt-get` 来完成安装：

```
pi@raspberrypi ~ $ sudo apt-get install python-opencv  
python-scipy python-numpy python-pip
```

要安装的程序比较多，所以安装过程中可能需要耐心等待一下。下一步，真正开始安装 SimpleCV 库：

```
pi@raspberrypi ~ $ sudo pip install https://github.com/  
ingenuitas/SimpleCV/zipball/master
```



当安装完成后，可以用 Python 的交互式命令行通过导入这个库来验证安装是否成功：

```
pi@raspberrypi ~ $ python
Python 2.7.3rc2 (default, May 6 2012, 20:02:25)
[GCC 4.6.3] on linux2
Type "help", "copyright", "credits" or "license" for more
information.
>>> import SimpleCV
>>>
```

如果库的导入一切顺利，你就已经完成了准备工作，可以开始在 Raspberry Pi 上开展计算机视觉实验。

显示图片

本章中的很多实验都需要在图形化桌面环境下完成，因为这样你才能在屏幕上显示图片。你可以用 IDLE，也可以直接用 Leafpad 把代码保存为 *.py* 文件，然后再从终端窗口中运行。

下面我们一起来了解一些 SimpleCV 操作图像的基础知识，从中可以学到如何从摄像头上捕获图像。只有当你学会了如何从摄像头上捕获图像，后面才能尝试进行人脸检测。

1. 在你的主目录中创建一个新的目录，名为 *simplecv-test*。
2. 打开 Midori 浏览器，在网上找一张你喜爱的图片。在我们的例子中，使用 Wikipedia 上的树莓照片，把它保存为 *raspberrypiess.jpg*。
3. 在图片上点击鼠标右键，选择 Save Image。
4. 把图片保存到先前创建的 *simplecv-test* 目录中。
5. 在文件管理器（File Manager）中〔在附件（Accessories）菜单中〕打开 *simplecv-test* 目录，点击右键，选择 Create New →



Blank File。

6. 把新建的文件命名为 *image-display.py*。
7. 双击新创建的 *image-display.py* 文件，用 Leafpad 打开它。
8. 输入代码 9.1 中的代码。
9. 保存 *image-display.py* 文件并从终端运行它。如果一切顺利，你会看到如图 9.4 所示的窗口，里面显示了你选择的图片。你可以把这个窗口关闭或者在终端中按 Control-C 键中断程序运行。

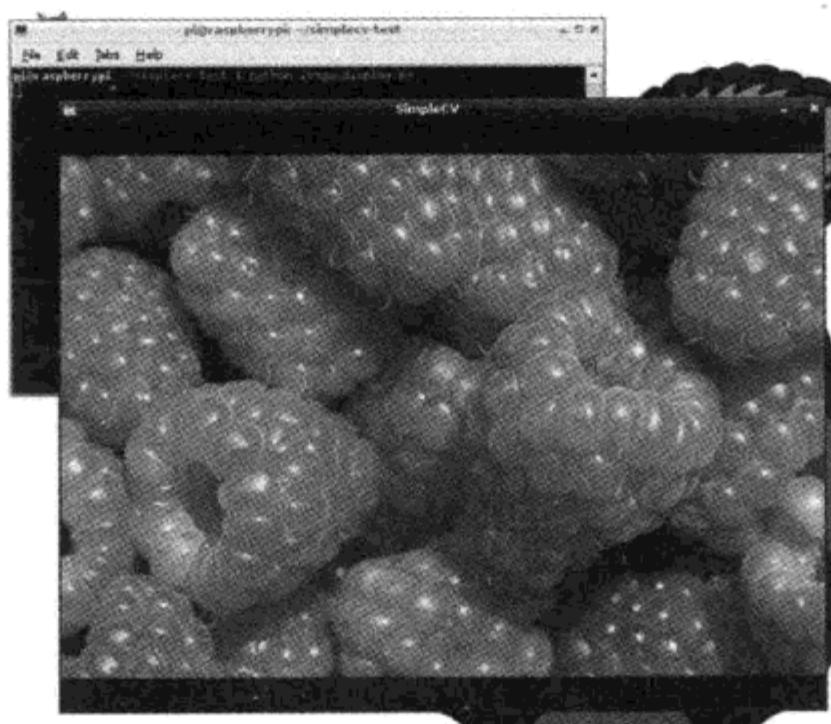


图9.4 在窗口中显示的树莓照片

代码 9.1 *image-display.py* 的源代码

```
from SimpleCV import Image, Display①
from time import sleep

myDisplay = Display()②
raspberryImage = Image("raspberries.jpg")③
raspberryImage.save(myDisplay)④
while not myDisplay.isDone():⑤
    sleep(0.1)
```

① 导入 SimpleCV 中的图像与显示相关的函数。



- ② 创建一个新的窗口对象。
- ③ 把 *raspberries.jpg* 文件加载到内存成为 *image* 对象。
- ④ 在窗口中显示这个图像。
- ⑤ 让程序在显示完图像后不立即退出，等待图像窗口关闭。

修改图片

现在你已经学会了如何把图片加载到内存并在屏幕上显示它，下一步是学习如何在显示图片前对它进行一些修改。这里所说的修改，是指修改内存中的图片对象，而并不是真正地修改磁盘上的图片文件。

1. 把 *image-display.py* 文件另存为 *superimpose.py*。
2. 按照代码 9.2 修改代码，增加处理逻辑。
3. 保存文件，并从命令行上运行它。
4. 你会看到与之前相同的图片，但图片上被加上了图形和文字。

代码 9.2 *superimpose.py* 的源代码

```
from SimpleCV import Image, Display, DrawingLayer, Color①
from time import sleep

myDisplay = Display()

raspberryImage = Image("raspberries.jpg")

myDrawingLayer = DrawingLayer((raspberryImage.width,
raspberryImage.height))②

myDrawingLayer.rectangle((50,20), (250, 60), filled=True)③
myDrawingLayer.setFontSize(45)
myDrawingLayer.text("Raspberries!", (50, 20), color=Color.
WHITE)④

raspberryImage.addDrawingLayer(myDrawingLayer)⑤
```



```
raspberryImage.applyLayers()⑥  
  
raspberryImage.save(myDisplay)  
  
while not myDisplay.isDone():  
    sleep(0.1)
```

- ① 与上一个示例一样，导入 SimpleCV 库的图像和显示相关的函数，再加上绘画层和颜色相关的函数。
- ② 创建一个与原图片一样大的绘画层。
- ③ 在绘画层上画一个矩形并填充上颜色，起止坐标是 (50,20) ~ (250,60)。
- ④ 在绘画层上用白色绘制“Raspberries!”的文字，起始坐标是 (50,20)。
- ⑤ 把 myDrawingLayer 这个绘画层添加到原始的 raspberry Image 图片对象上。
- ⑥ 把绘画层与原始图片合并（图 9.5）。

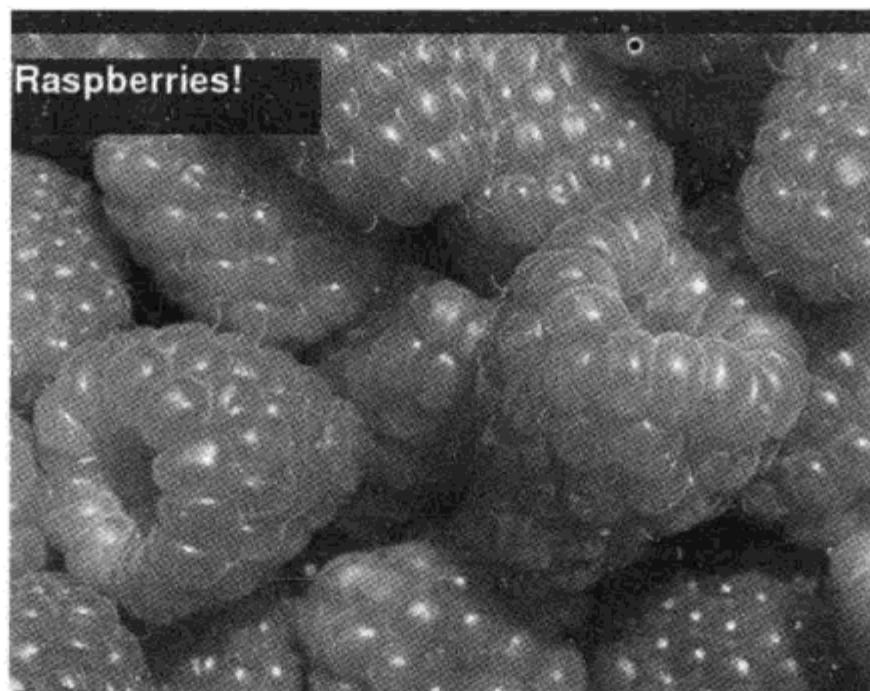


图9.5 修改过的树莓照片

除了把修改过的图片显示在屏幕上，你也可以很方便地把它保



存到文件中，如代码 9.3 所示。

代码 9.3 *superimpose-save.py* 的源代码

```
from SimpleCV import Image, DrawingLayer, Color
from time import sleep

raspberryImage = Image("raspberries.jpg")

myDrawingLayer = DrawingLayer((raspberryImage.width,
raspberryImage.height))

myDrawingLayer.rectangle((50,20), (250, 60), filled=True)
myDrawingLayer.setFontSize(45)
myDrawingLayer.text("Raspberries!", (50, 20), color=Color.
WHITE)

raspberryImage.addDrawingLayer(myDrawingLayer)
raspberryImage.applyLayers()

raspberryImage.save("raspberries-titled.jpg")❶
```

❶ 把内存中修改过的图片保存为一个新的文件，命名为 *raspberries-titled.jpg*。

上面的代码中没有打开窗口把图片显示在屏幕上，所以你可以脱离图形桌面环境直接在纯命令行模式下运行这个程序。你还可以通过修改这个程序，实现用一个命令给一批图片加上水印的功能。

你对图片的修改也仅仅局限于添加文字或绘制矩形。下面是 SimpleCV 中其他一些可以使用的绘图函数。可以在 <http://simplecv.org/docs/SimpleCV.html#SimpleCV.DrawingLayer.DrawingLayer> 上查阅它们完整的文档。

- `circle` (圆)
- `ellipse` (椭圆)
- `line` (线条)



- polygon (多边形)
- bezier curve (贝塞尔曲线)

操作摄像头

用 SimpleCV 获取摄像头的视频流与打开一个图片文件加载到内存中的方式非常相似，可以很轻松地写出属于你自己的摄像头图像预览程序。

1. 新建一个文件，名为 *basic-camera.py*，输入代码 9.4 中的代码。
2. 接上你的 USB 摄像头，运行上一步中创建的脚本。你会看到如图 9.6 所示的窗口，里面显示了你的摄像头上捕捉到的画面。
3. 在终端中按 Control-C 键可以关闭该视频窗口。

代码 9.4 *basic-camera.py* 的源代码

```
from SimpleCV import Camera,Display
from time import sleep

myCamera = Camera(prop_set={"width": 320, "height": 240})①

myDisplay = Display(resolution=(320, 240))②

while not myDisplay.isDone():③
    myCamera.getImage().save(myDisplay)④
    sleep(.1)⑤
```

- ① 创建一个摄像头对象，并把图像大小设置为 320×240 以提高程序性能。
- ② 窗口大小也设置成 320×240 。
- ③ 循环执行缩进的代码块直到窗口被关闭。
- ④ 从摄像头获取一帧图像并在窗口中显示。
- ⑤ 在显示每一帧图像之间暂停 0.1s。



图9.6 把摄像头图像显示在屏幕上

你可以把上面两个例子的代码结合起来，让 Python 脚本可以从摄像头捕获一帧图像并保存为 *.jpg* 文件：

```
from SimpleCV import Camera
from time import sleep

myCamera = Camera(prop_set={'width':320, 'height': 240})

frame = myCamera.getImage()
frame.save("camera-output.jpg")
```

人脸检测

SimpleCV 有一个很强大的函数——`findHaarFeatures`，这是一个在图像中搜索匹配某一种特定模式（或称 cascade）的算法，在 SimpleCV 中自带了几种模式，包括脸、鼻子、眼睛、嘴和身体，如果必要，你也可以下载或生成你自己模式文件。`findHaarFeatures` 可以分析图像并从中匹配出对应的模式，然后返回匹配到的部分在图像中的位置。这就意味着，你可以从像文件或摄像头捕获的图像中匹配汽车、动物或人。下面以人脸识别为例，



实验一下 `findHaarFeatures` 的功能。

1. 在 `simplecv-test` 目录中创建一个新文件，命名为 `face-detector.py`。
2. 输入代码 9.5 中的代码。
3. 连接你的摄像头并对准人的脸部，从命令行运行新建的脚本。
4. 在终端窗口中，你可以看到一系列 `findHaarFeatures` 所检测到的人脸坐标。尝试调整一下摄像头的角度，可以看到这些数据会发生变化。你还可以尝试着将人脸的照片放到摄像头前，看看会发生什么。

代码 9.5 `face-detector.py` 的源代码

```
from SimpleCV import Camera, Display
from time import sleep

myCamera = Camera(prop_set={"width":320, "height": 240})

myDisplay = Display(resolution=(320, 240))

while not myDisplay.isDone():
    frame = myCamera.getImage()
    faces = frame.findHaarFeatures('face')❶
    if faces:❷
        for face in faces:❸
            print "Face at: " + str(face.coordinates())
    else:
        print "No faces detected."
    frame.save(myDisplay)
    sleep(.1)
```

❶ 在图像中检测人脸并把检测到的人脸图像保存到 `faces` 对象中。



② 如果 `findHaarFeatures` 检测到至少一张人脸，则执行下面的代码块中的逻辑。

③ 对于检测结果 `faces` 中的 `face` 对象（对应一张检测出来的人脸），运行下面的代码（`print` 语句）。

如果你的尊容出现在了屏幕上，但你仍然看到“`No faces detected`”的提示，可以尝试用下面的步骤排查问题。

- 光线是不是够亮？如果你所处的环境比较黑暗，会影响算法的检测能力。可以尝试增加一点照明。

- 程序中使用的 Haar cascade 适用于检测一个正面的人脸。如果你把头抬得太高或摄像头摆放得不够水平，都会影响到算法检测人脸的准确性。

项目：Raspberry Pi 照相馆

使用 Python，你可以把各种库都整合在一起，设计出复杂的项目。把第 8 章中所提到的 GPIO 库与 SimpleCV 结合起来，你可以实现一个 Raspberry Pi 照相馆，这个项目一定能使你在朋友圈中出尽风头（图 9.7）。通过使用 SimpleCV 中的 `findHaarFeatures` 函数，你还可以给你的照相馆中拍摄的人像添加各种元素，如添加帽子、眼镜、胡子或胡须。这个示例是基于 Kurt Demaagd、Anthony Oliver、Nathan Oostendorp 和 Katherine Scott 所著的 *Practical Computer Vision with SimpleCV* (<http://shop.oreilly.com/product/0636920024057.do>) 一书中的例子所设计的。

这里是把你的 Raspberry Pi 变为照相馆所需要的材料：

- USB 摄像头；
- 显示器；



- 按钮，任何你所喜欢的样式都可以；
- 长度合适的连接线；
- 一个 $10k\Omega$ 的电阻。

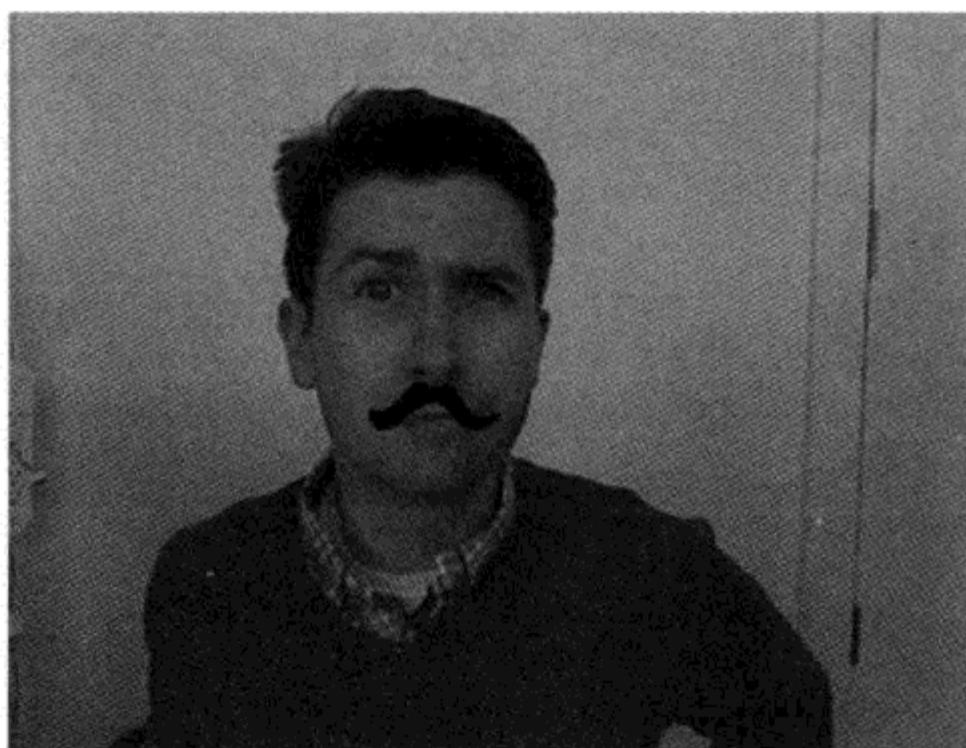


图9.7 Raspberry Pi 照相馆的输出

在开始之前，先确认你的 Raspberry Pi 上已经正确安装了 Python 的 RPi.GPIO 和 SimpleCV 库。请参考“在 Python 中安装并测试 GPIO”和“安装并测试 SimpleCV”的内容。

1. 与第 8 章中所做的一样，把按钮接在 GPIO 24 接口上。按钮的一个引脚应该接 3.3V，另一个接 GPIO 24。不要忘记在地线和接 GPIO 的按钮引脚间加上一个 $10k\Omega$ 的下拉电阻。

2. 找一张背景为白色胡子的图片，把它命名为 *mustache.png* 并保存到 Raspberry Pi 上的一个新目录中。你可以从本书的支持网站上下载各章节的代码（参考“如何联系我们”），其中 *ch09* 子目录中有我们制作好的胡子图片。

3. 在这个目录中，创建一个新文件，名叫 *photobooth.py*，输入代码 9.6 中的代码并保存。

代码 9.6 *photobooth.py* 的源代码

```
from time import sleep, time
from SimpleCV import Camera, Image, Display
import RPi.GPIO as GPIO

myCamera = Camera(prop_set={"width":320, "height": 240})
myDisplay = Display(resolution=(320, 240))
stache = Image("mustache.png")
stacheMask = \
    stache.createBinaryMask(color1=(0,0,0),
color2=(254,254,254))❶
stacheMask = stacheMask.invert()❷

GPIO.setmode(GPIO.BCM)
GPIO.setup(24, GPIO.IN)

def mustachify(frame):❸

    faces = frame.findHaarFeatures("face")
    if faces:
        for face in faces:
            print "Face at: " + str(face.coordinates())
            myFace = face.crop()❹
            noses = myFace.findHaarFeatures("nose")
            if noses:
                nose = noses.sortArea()[-1]❺
                print "Nose at: " + str(nose.coordinates())
                xmust = face.points[0][0] + nose.x -
(stache.width/2)❻
                ymust = face.points[0][1] + nose.y +
(stache.height/3)❼
            else:
                return frame❽
            frame = frame.blit(stache, pos=(xmust, ymust),
mask=stacheMask)❾
            return frame❿
    else:
        return frame❿
```



```
while not myDisplay.isDone():
    inputValue = GPIO.input(24)
    frame = myCamera.getImage()
    if inputValue == True:
        frame = mustachify(frame)⑪
        frame.save("mustache-" + str(time()) + ".jpg")⑫
        frame = frame.flipHorizontal()⑬
        frame.show()
        sleep(3)⑭
    else:
        frame = frame.flipHorizontal()⑮
        frame.save(myDisplay)
        sleep(.05)
```

- ① 创建胡子的蒙版，把黑色设置为透明（color1 和 color2 这两个参数用于设置 RGB 值，范围是 0 ~ 255）。
- ② 把蒙版取反，再绘制这个图片时就只有黑色的像素会被显示出来了，其他颜色都变成透明色。
- ③ 定义一个函数，接受一帧图像的输入，如果在这帧图像中可以检测到脸和鼻子，就给这个图像上添加一个胡子。
- ④ 创建只包含脸的图片对象，这样检测鼻子时可以处理得更快一些。
- ⑤ 如果在脸上找到了多个具有鼻子特征的对象，选择其中最大的一个作为鼻子。
- ⑥ 设置胡子的 x 坐标。
- ⑦ 设置胡子的 y 坐标。
- ⑧ 如果没有找到鼻子，就返回原始的图像。
- ⑨ 使用 blit 函数（块图像传输，Block Image Transfer）把胡子画到原始的图像上。
- ⑩ 返回添加了胡子的图像。



- ⑪ 如果没有找到人脸，直接返回原始图像。
- ⑫ 把图像传入 `mustachify` 函数。
- ⑬ 把图像保存为 JPG 文件，以当前的时间来给文件命名。
- ⑭ 在显示图像之前，先进行一次水平翻转，这样显示出来的图像是摄像头所拍摄到的画面的镜像画面。

⑮ 在屏幕上把保存下来的图片保持显示 3s。

⑯ 如果按钮没有被按下，直接把实时图像翻转后显示出来。

现在可以试一下你的作品了，把摄像头连接好，在终端上进入你的胡子图片和 `photobooth.py` 所在的目录，运行脚本：

```
pi@raspberrypi ~ $ sudo python photobooth.py
```

屏幕上会显示摄像头所捕捉到的图像。当按钮被按下时，程序会检测人脸、添加胡子并保存照片（所有的照片都会保存在与脚本相同的路径下）。

进一步学习

Practical Computer Vision with SimpleCV

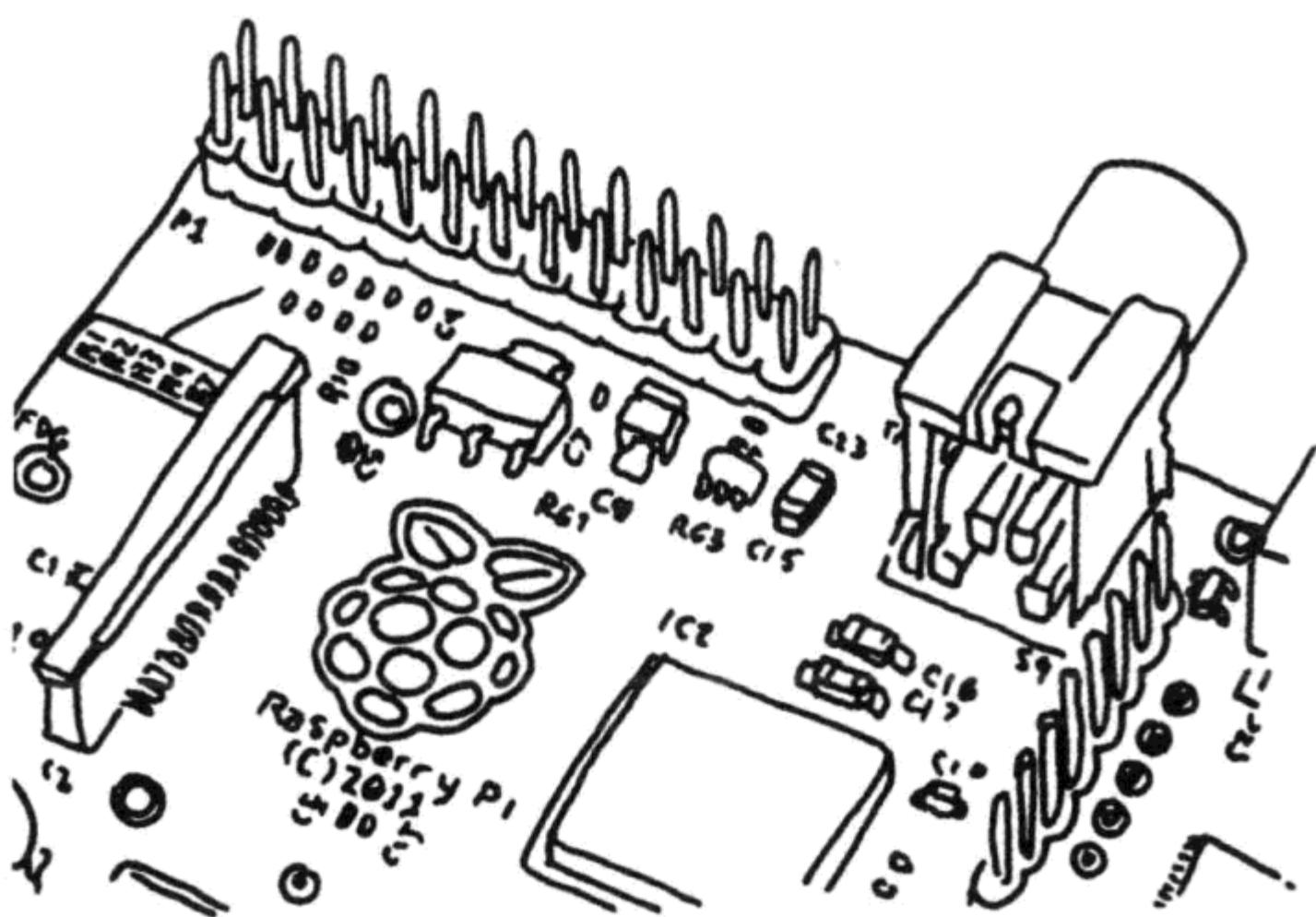
(<http://shop.oreilly.com/product/0636920024057.do>)

由 Kurt Demaagd、Anthony Oliver、Nathan Oostendorp 和 Katherine Scott 合著的《SimpleCV 实践指南》。它包含了大量示例代码和项目，适合进一步学习用 Python 进行图像和计算机视觉开发。

第 10 章

Python 与 Internet

Python and the Internet





Python 拥有一个非常活跃的开发社区，很多人乐意把他们的作品以开源库的形式贡献到社区中，这让使用 Python 完成各种任务变得更为容易。其中的一些库可以帮助我们轻松地在程序中实现连接 Internet 并从网上获取天气信息、发送电子邮件或文本信息、从 Twitter 获得热门话题或搭建一个 Web 服务器。

•

在本章中，我们会学习用 Raspberry Pi 接入网络并完成一些网络相关的应用开发。首先，我们来看看如何从 Internet 上获取信息，然后再尝试把我们的 Raspberry Pi 变成一个 Web 服务器。

从 Web 服务器下载数据

当你在网络浏览器中输入网址并按下回车键后，你的浏览器就充当了一个客户端的角色，它向服务器发出连接请求，并建立一个连接，然后服务器会返回一个 Web 页面。当然，在网络环境中，客户端不仅可以是一个浏览器，它也可以是一个电子邮件程序，或者是你的电脑或手机上的一个天气预报组件，又或者是一个向网络上传你的最高分纪录的游戏程序。在本章的前半部分，我们会专注于把 Raspberry Pi 当作一个客户端来使用。你所编写的代码，会向服务器发起连接，并获取信息。在开始编写程序前，你需要先安装一个常用的 Python 库——Requests，Requests 库提供了通过 HTTP(超文本传输协议，Hypertext Transfer Protocol) 向 Web 服务器发送请求的功能。在命令行上输入下面的命令安装这个库：



```
pi@raspberrypi ~ $ sudo apt-get install python-requests
```

确认是否正确安装了 Request 库：

```
pi@raspberrypi ~ $ python
Python 2.7.3rc2 (default, May 6 2012, 20:02:25)
[GCC 4.6.3] on linux2
Type "help", "copyright", "credits" or "license" for more
information.
>>> import requests
>>>
```

如果没有出错提示，就意味着 Requests 库已经被正确地安装，并被导入到当前的 Python 会话，你就可以尝试使用它了：

```
>>> r = requests.get('http://www.google.com/')

>>>
```

也许你会有点失望，因为看上去什么也没有发生。不过事实上并非如此，这个请求获得的所有数据都已经保存在了 `r` 对象中。你可以用下面的方法来显示这个请求的返回码：

```
>>> r.status_code
200
```

HTTP 返回码 200 表示这个请求已经成功完成。表 10.1 中列出了其他常用的 HTTP 返回码。

表10.1 常见HTTP返回码

返回码	含 义	返回码	含 义
200	正常	401	没有权限
301	永久跳转	404	文件不存在
307	临时跳转	500	服务器错误



如果你想查看服务器返回的内容，用这个方法：

```
>>> r.text
```

如果一切正常，这个命令会显示出一串很长的文本。这些文本中有些是可读的文字，有些则是理解起来有点困难的符号。这是 Google 首页的 HTML 代码，原本是应该由浏览器进行解析并渲染成最终网页展现给你的。

不过，也不是所有的 HTTP 请求都会被设计成由浏览器来渲染成网页。有时候会仅仅使用 HTTP 来传递数据本身，而不包含指定这些数据展现方式的信息。有很多网站向公众提供这样的数据协议，所以我们可以不通过浏览器，直接与这些网站交换数据。无论这个数据协议是公开的还是私有的，数据协议的规范常常被称为 API (Application Programming Interface，应用程序编程接口)。通过使用 API，可以在一个软件与另一个软件之间进行数据交换，在 Internet 的网站之间交换数据也是它的一个常见应用。

我们可以开发这样一个程序：如果天气预报有雨，则当你离开家门口时，程序会提醒你带上雨伞。当然，这不需要你建立一个气象台来预测天气是否会下雨，而是可以从很多现成的 API 获得当天的天气预报。

获取天气预报

为了可以获知今天会不会下雨，我们将使用 Weather Underground (<http://www.wunderground.com/>) 的 API 获取天气信息。



不同 API 的使用方法不尽相同，你需要通过查看它们的文档来决定某个 API 是否可以满足你的需求。另外，大部分 API 都会有在一段时间内的调用次数限制，哪怕



是那些需要付费使用的。很多 API 提供者会提供免费试用，允许你每天免费少量调用这个 API，这些免费的 API 非常适合用于实验或个人日常使用。

1. 在浏览器中，打开 Weather Underground 的 API 主页 (<http://www.wunderground.com/weather/api/>)，输入你的信息注册一个账号。
2. 用你的账号登录后，打开 Key Settings 页面。
3. 这个页面提供了一个很长的字符串，这个字符串就是属于你的 API 密钥（图 10.1）。在你每次向 API 发起请求时，都需要带上这个 API 密钥。如果你滥用网站的服务，向他们的 API 发起了过多的请求，他们就可以把这个 API 密钥放入黑名单，拒绝你的后续请求，直到你付费购买他们的服务。



图10.1 本书作者Matt的Weather Underground API账号，右上角显示的是他的API密钥

4. 打开 Documentation 中的 Forecast 链接，你可以看到天气预报请求返回的数据内容。在这个页面底部，还给出一个用于获取旧金山（San Francisco）天气预报的请求 URL。注意，需要把你的 API 密钥填入这个 URL：



```
http://api.wunderground.com/api/YourAPIkey/forecast/q/CA/San_Francisco.json
```

5. 你可以把这个 URL 输入到浏览器的地址栏来验证它是否可以正常工作，浏览器会以 JSON (JavaScript Object Notation, JavaScript 对象表示法) (代码 10.1) 格式返回天气预报信息。请注意数据的层次结构。



尽管 JSON 的首字母 J 是 JavaScript 的缩写，JSON 其实在很多语言中都可以使用，多用于在应用程序之间通过 API 传递数据的场合。

6. 下面可以尝试一下用这个 API 来获取本地的天气预报了。修改上面的 URL，填入你所在的省和市，然后把代码写入一个新的 Python 脚本，命名为 *text-forcast.py* (代码 10.2)。

7. 正如你在代码中所看到的，如果要获取当天的天气预报文本，我们需要从返回的结果中找出正确的数据 (代码 10.1)。文本格式的天气预报信息可以从 `forecast` → `txt_forecast` → `forecastday` → `0` (表示第一个结果，也就是当天的结果) → `fcttext` 节点获得。

8. 在命令行上运行脚本，输出结果应该就是你所在地当天的天气情况，如“晴。最高温度 47°F。东北风，风速 5 ~ 10 mph”。

代码 10.1 从 Weather Underground API 返回的部分 JSON 信息

```
"response": {  
  "version": "0.1",  
  "termsofService":  
    "http://www.wunderground.com/weather/api/d/terms.html",  
  "features": {  
    "forecast": 1  
  }  
}
```



```
},
"forecast":{❶
    "txt_forecast": {❷
        "date":"10:00 AM EST",
        "forecastday": [❸
            {
                "period":0,
                "icon":"partlycloudy",

                "icon_url":"http://icons-ak.wxug.com/i/c/k/partlycloudy.
                gif",
                "title":"Tuesday",
                "fcttext":
                    "Partly cloudy. High of 48F. Winds from the NNE at
                    5 to 10 mph.",❹
                "fcttext_metric":
                    "Partly cloudy. High of 9C. Winds from the NNE at
                    10 to 15 km/h.",
                "pop":"0"
            },
        ],
    },
}
```

- ❶ `forecast` 是我们需要解释的最顶层数据。
- ❷ 在 `forecast` 节点中，我们需要找到文本预报信息。
- ❸ 在文本预报信息节点中，我们需要当天的预报。
- ❹ `fcttext` 是当天天气预报的文本。

代码 10.2 `text-forecast.py` 的源代码

```
import requests

key = "YOUR KEY HERE"❶
ApiUrl = \
    "http://api.wunderground.com/api/" + key + "/forecast/q/
NY/New_York.json"

r = requests.get(ApiUrl)❷
forecast = r.json❸
print forecast["forecast"]["txt_forecast"]["forecastday"][0]
["fcttext"]❹
```



- ① 把这里的字符串替换成你的 API 密钥。
- ② 从 Weather Underground 获取纽约的天气预报（把这里的地址换为你自己所在的省和市）。
- ③ 获取返回的 JSON 值并把它解析成一个 Python 的字典对象。
- ④ 从这个字典对象中逐级获取到当天的天气预报文本。

现在，你已经完成了一个 Python 脚本，使用这个脚本可以随时从网上获取天气预报。但是，如何能让 Raspberry Pi 知道今天到底会不会下雨呢？虽然我们可以直接从天气预报的文本中寻找“rain”（下雨）、“drizzle”（小雨）、“thunderstorm”（暴风雨）、“showers”（阵雨）等关键词，但我们还有一个更好的办法：Weather Underground API 返回的结果中，有一个字段名为 `pop`，指的是降水概率。`pop` 字段的值为 0 ~ 100%，表示了下雨、下雪的可能性。

在我们的例子中，假设当降水概率超过 30% 时，就让 Raspberry Pi 提醒我们带上雨伞。

1. 把一个 LED 连接到 GPIO 25 接口（图 7.4）。
2. 创建一个名为 `umbrella-indicator.py` 的文件，输入代码 10.3 中的代码。别忘了把程序中的 Weather Underground API 的 URL 中的 API 密钥替换成你自己的。
3. 以 `root` 权限运行这个脚本：`sudo python umbrella-indicator.py`。

代码 10.3 `umbrella-indicator.py` 的源代码

```
import requests
import RPi.GPIO as GPIO
import time
GPIO.setmode(GPIO.BCM)
GPIO.setup(25, GPIO.OUT)
```



```
key = "YOUR KEY HERE"❶
ApiUrl = \
    "http://api.wunderground.com/api/" + key + "/forecast/" \
    "q/NY/New_York.json"

while True:
    r = requests.get(ApiUrl)
    forecast = r.json
    popValue = forecast["forecast"]["txt_forecast"] \
    ["forecastday"][0]["pop"]❷
    popValue = int(popValue)❸

    if popValue >= 30:❹
        GPIO.output(25, GPIO.HIGH)
    else:❺
        GPIO.output(25, GPIO.LOW)

    time.sleep(180) # 3 minutes❻
```

- ❶ 与以前一样，把这里的 API 密钥换成你自己的。
- ❷ 获取今天的降水概率并存入 `popValue` 变量中。
- ❸ 把 `popValue` 由字符串转换为数字，以便后续可以对它按数值来处理。
- ❹ 如果降水概率值大于 30，则点亮 LED。
- ❺ 否则，熄灭 LED。
- ❻ 等待 3min 后再次获取天气信息。所以，这个脚本一天对 API 的请求会小于 500 次的限制值。

实验完成后，按 Control-C 键结束程序运行。

除了 Weather Underground 以外，网上还有很多其他 API 可以使用。表 10.2 中列出了其他一些提供 API 的网站。



表10.2 常见网站的API

网 站	API 参考手册的地址
Facebook	https://developers.facebook.com/
Flickr	http://www.flickr.com/services/api/
Foursquare	https://developer.foursquare.com/
Reddit	https://github.com/reddit/reddit/wiki/API
Twilio	http://www.twilio.com/
Twitter	https://dev.twitter.com/
YouTube	https://developers.google.com/youtube/

用 Pi 提供服务（做 Web 服务器）

除了可以用 Raspberry Pi 从 Internet 服务器上获取信息以外，Pi 自身也可以作为服务器对外提供信息。在 Raspberry Pi 上有很多种 Web 服务器软件可供你选择。例如，传统的 Apache 或 lighttpd，它们都可以用来向外提供 Web 服务。这些服务器通常用于向外提供用于构成网页的 HTML 文件和图片，但它们也可以用于提供声音、视频、软件等其他类型的数据。

现在有一些新的工具可以扩展 Python、Ruby 或 JavaScript 等语言，在程序接收到 HTTP 请求时动态地生成 HTML 页面。这种方式非常适合用于从远端的 Web 浏览器中触发一个物理事件、保存数据或者读取传感器的值。你甚至可以为你的电子项目创建一个 JSON 格式的 API 服务。

Flask 入门

下面我们将使用 Flask (<http://flask.pocoo.org/>) 这个 Python 的 Web 框架把 Raspberry Pi 变为一个动态 Web 服务器。Flask 自带了



很多强大的功能，并且还可以通过扩展来支持用户验证、生成表单和访问数据库等功能。当然，在使用 Flask 时，你也可以使用大量的标准 Python 库来帮助你完成工作。

要安装 Flask，需要先安装 pip。如果你还没有安装 pip，可以这样来安装它：

```
pi@raspberrypi ~ $ sudo apt-get install python-pip
```

安装 pip 后，你可以用它来安装 Flask 和相关的依赖包：

```
pi@raspberrypi ~ $ sudo pip install flask
```

要验证 Flask 是否正确安装，创建一个名为 *hello-flask.py* 的新文件并输入代码 10.4 中的代码。不要被这些代码吓倒，如果你现在暂时不能完全理解这些代码也没关系。这段代码中，最重要的部分就是包含 "Hello World!" 字符串的那一块。

代码 10.4 *hello-flask.py* 的源代码

```
from flask import Flask
app = Flask(__name__)❶

@app.route("/")
def hello():
    return "Hello World!"❷

if __name__ == "__main__":❸
    app.run(host="0.0.0.0", port=80, debug=True)❹
```

- ❶ 创建一个名为 `app` 的 Flask 对象。
- ❷ 当有人访问网页服务器的根目录时，执行下面的代码。
- ❸ 向客户端发送 "Hello World!" 字符串。
- ❹ 判断是否这个脚本是从命令行上直接运行。



⑤ 让服务器在 80 端口上监听，并在出错时显示出错信息。



在运行这个程序之前，需要知道你的 Raspberry Pi 的 IP 地址（参考“网络”）。另一种方式是安装 `avahi-daemon`（在命令行上运行 `sudo apt-get install avahi-daemon`），这样可以让你通过 `http://raspberrypi.local` 这样的地址访问到局域网中 Raspberry Pi。如果你需要以这种方式从 Windows 上访问 Raspberry Pi，则需要在 Windows 中启用 Bonjour 服务（<http://support.apple.com/kb/DL999>）。

现在可以运行你的 Web 服务器了，这个程序也需要以 `root` 权限运行：

```
pi@raspberrypi ~ $ sudo python hello-flask.py
* Running on http://0.0.0.0:80/
* Restarting with reloader
```

在同一个局域网中的另一台电脑的浏览器地址栏中输入 Raspberry Pi 的 IP 地址，如果浏览器上显示出 "Hello World!"，就表示你已经把 Raspberry Pi 上的 Web 服务器正确地配置好了。与此同时，你也会发现，Raspberry Pi 的终端上会打印出几行信息：

```
10.0.1.100--[19/Nov/2012 00:31:31] "GET / HTTP/1.1" 200 -
10.0.1.100--[19/Nov/2012 00:31:31] "GET /favicon.ico HTTP/
1.1" 404 -
```

第一行信息表示浏览器请求了服务器上的根目录，然后我们的 Web 服务器返回了 HTTP 返回值 200，表示正常返回。第二行信息表示浏览器自动请求了收藏夹图标（通常用于在浏览器地址栏最左侧显示）。由于服务器上没有 `favicon.ico` 文件，所以返回了 404，



表示请求的文件不存在。

如果你想向浏览器返回一个 HTML 格式的网站，把所有的 HTML 代码都写在脚本中显示不是一个好主意。Flask 使用一个名为 Jinja2 的模板引擎 (<http://jinja.pocoo.org/docs/templates/>)，使你可以用一个包含占位符的页面模板来制作网页，然后再用程序把占位符替换成实际的动态数据。

如果 *hello-flask.py* 脚本还在运行，按 Control-C 键把它停止。

创建名为 *hello-template.py* 并输入代码 10.5 中的代码。在 *hello-template.py* 所在的目录下，创建一个名为 *templates* 的子目录。在 *templates* 子目录中，创建一个名为 *main.html* 的文件并输入代码 10.6 中的代码。这个 HTML 页面模板中所有括号的部分会被解析为变量名，在 Python 脚本调用 `render_template` 函数时，会被实际的数据所替换。

代码 10.5 *hello-template.py* 的源代码

```
from flask import Flask, render_template
import datetime
app = Flask(__name__)

@app.route("/")
def hello():
    now = datetime.datetime.now()❶
    timeString = now.strftime("%Y-%m-%d%H:%M")❷
    templateData = {
        'title' : "HELLO!",
        'time': timeString
    }❸
    return render_template("main.html", **templateData)❹

if __name__ == "__main__":
    app.run(host="0.0.0.0", port=80, debug=True)
```

❶ 获取当前日期时间值，并保存在 `now` 变量中。



- ② 把 `now` 中的时间值按指定的格式转换成字符串。
- ③ 创建一个字典变量（一组键值对，如 `title` 是键，它所对应的值是 `HELLO!`）用于渲染模板。
- ④ 用 `templateData` 字典中的键值对渲染 `main.html` 模板并返回给浏览器。

代码 10.6 `templates/main.html` 的源代码

```
<!DOCTYPE html>
<head>
    <title>{{ title }}</title>①
</head>
<body>
    <h1>Hello, World!</h1>
    <h2>The date and time on the server is: {{ time }}</h2>②
</body>
</html>
```

- ① 用 `title` 变量的值作为 HTML 页面的标题。

- ② 用 `time` 变量的值作为页面内容的一部分。

当你运行 `hello-template.py`（与以前一样，需要用 `sudo` 来运行它）并在浏览器中访问 Raspberry Pi 的 IP 地址时，你应该可以看到一个标题为“HELLO!”的页面，上面显示着 Raspberry Pi 上的当前日期时间。



取决于你的网络的情况，你在 Raspberry Pi 上在局域网内搭建的网站很可能无法通过 Internet 直接访问。如果你想让你的网站在局域网外也可以访问，你需要配置网络路由器上的“端口映射”功能。请参考你的路由器说明书了解如何进行相关的配置。



把 Web 与现实世界相连

可以在使用 Flask 的同时也使用其他 Python 库，以便给你的网站增加更多的功能。例如，通过使用 RPi.GPIO 模块（第 8 章），你可以创建一个与现实世界相连的网站。如果想尝试一下的话，请像简易发音板项目中那样按图 8.2 的方法在 GPIO 接口 23、24、25 上连接 3 个按钮。

下面的代码对 *hello-template.py* 进行了扩展，把 *hello-template.py* 复制一份命名为 *hello-gpio.py* 并进行如下修改：添加 RPi.GPIO 模块，并针对读取按钮状态添加一个新的 `route`。新添加的 `route` 会从 URL 中获取一个传入变量的值，用于决定读取哪一个 GPIO 接口的状态。

你还需要创建一个新的网页模板，命名为 *pin.html*。它与 *main.html* 也很相似，所以你可以基于 *main.html* 参考进行修改，参考代码 10.7。

修改过的 *hello-gpio.py*：

```
from flask import Flask, render_template
import datetime
import RPi.GPIO as GPIO
app = Flask(__name__)

GPIO.setmode(GPIO.BCM)

@app.route("/")
def hello():
    now = datetime.datetime.now()
    timeString = now.strftime("%Y-%m-%d %H:%M")
    templateData = {
        "title" : "HELLO!",
```



```
        "time": timeString
    }
    return render_template("main.html", **templateData)

@app.route("/readPin/<pin>")❶
def readPin(pin):
    try:❷
        GPIO.setup(int(pin), GPIO.IN)❸
        if GPIO.input(int(pin)) == True:❹
            response = "Pin number " + pin + " is high!"
        else:❺
            response = "Pin number " + pin + " is low!"
    except:❻
        response = "There was an error reading pin " + pin
        + "."

    templateData = {
        "title" : "Status of Pin" + pin,
        "response" : response
    }

    return render_template("pin.html", **templateData)

if __name__ == "__main__":
    app.run(host='0.0.0.0', port=80, debug=True)
```

- ❶ 添加一个新的 route，并以 GPIO 接口编号作为参数。
- ❷ 如果缩进的代码中发生了异常，就执行 except 块中的代码。
- ❸ 从 URL 中获取 GPIO 接口编号，转换为数字，并把对应接口状态设置为输入设置。
- ❹ 如果读到的接口状态是高电平，则返回表示高电平的文本信息。
- ❺ 否则返回表示低电平的文本信息。
- ❻ 如果读取接口状态时发生问题，显示相应的出错信息。

代码 10.7 *templates/pin.html* 的源代码

```
<!DOCTYPE html>
<head>
  <title>{{ title }}</title>①
</head>

<body>
  <h1>Pin Status</h1>
  <h2>{{ response }}</h2>②
</body>
</html>
```

- ① 根据 *hello-gpio.py* 中的 `title` 值渲染页面的标题。
- ② 根据 *hello-gpio.py* 中返回的 `response` 值渲染 HTML 页面。

当上面的脚本运行起来后，如果你在浏览器中访问 Raspberry Pi 的 IP 地址，你仍然会看到之前我们所创建的“Hello World!”页面。如果在访问的 URL 的最后加上 `/readPin/24`，得到一个类似于 `http://10.0.1.103/readPin/24` 这样的 URL，则会返回一个页面提示这个 GPIO 接口上读取到的状态为低电平。如果你按下 GPIO 24 上所接的按钮不放，并刷新这个页面，则页面上显示状态应该为高电平。

可以尝试按下其他按钮并更换对应的 URL。这段代码最大的优点是，我们只需要写一份读取 GPIO 接口状态并在页面上显示状态的代码，就可以操作所有的 GPIO 接口，就像我们为每一个 GPIO 接口都写了一个状态页面一样。

项目：Web 台灯

在第 7 章的“项目：定时台灯”一节中，我们演示了如何用 Raspberry Pi 制作一个定时开关来控制台灯。现在你学会了 Python 中的 Flask 库，就可以实现一个通过 Web 界面来控制的台灯了。这



一个简单的项目演示了如何用 Raspberry Pi 完成一个可以通过 Internet 来远程控制的设备。

在前一个 Flask 例子中，我们实现了用一个程序来控制多个 GPIO 接口。一旦你完成了这个例子，以后你需要通过网络来控制更多设备时，也会变得非常容易。

1. 这个项目所需的硬件与实现“项目：定时台灯”中所用到硬件完全一致。

2. 与定时台灯项目中一样，把 PowerSwitch Tail II 连接到 GPIO 25 接口上。

3. 如果你还有一个 PowerSwitch Tail II，则可以把它接到 GPIO 24 接口上，用于控制其他的电器设备。否则，在 GPIO 24 上连接一个 LED，我们用它来演示如何用一个程序去控制多个设备。

4. 在你的主目录下创建一个新的目录并命名为 *WebLamp*。

5. 在 *WebLamp* 目录中，创建 *weblamp.py* 文件并输入代码 10.8。

6. 在 *WebLamp* 目录中创建一个新的目录并命名为 *templates*。

7. 在 *templates* 目录中，创建 *main.html*，输入代码 10.9 中的代码。

在命令行下，切换到 *WebLamp* 目录中并启动服务器（如果之前已经启动过 Flask 服务器，按 Control-C 键中止）：

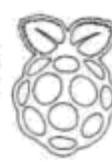
```
pi@raspberrypi ~/WebLamp $ sudo python weblamp.py
```

代码 10.8 *weblamp.py* 的源代码

```
import RPi.GPIO as GPIO
from flask import Flask, render_template, request
app = Flask(__name__)

GPIO.setmode(GPIO.BCM)

pins = {
```



```
24 : {'name' : 'coffee maker', 'state' : GPIO.LOW},
25 : {'name' : 'lamp', 'state' : GPIO.LOW}
}

for pin in pins:②
    GPIO.setup(pin, GPIO.OUT)
    GPIO.output(pin, GPIO.LOW)

@app.route("/")
def main():
    for pin in pins:
        pins[pin]['state'] = GPIO.input(pin)③
    templateData = {
        'pins' : pins④
    }
    return render_template('main.html', **templateData)⑤

@app.route("/<changePin>/<action>")⑥
def action(changePin, action):
    changePin = int(changePin)⑦
    deviceName = pins[changePin]['name']⑧
    if action == "on":⑨
        GPIO.output(changePin, GPIO.HIGH)⑩
        message = "Turned " + deviceName + " on."⑪
    if action == "off":
        GPIO.output(changePin, GPIO.LOW)
        message = "Turned " + deviceName + " off."
    if action == "toggle":
        GPIO.output(changePin, not GPIO.input(changePin))⑫
        message = "Toggled " + deviceName + "."

    for pin in pins:
        pins[pin]['state'] = GPIO.input(pin)⑬

    templateData = {
        'message' : message,
        'pins' : pins
    }⑭
    return render_template('main.html', **templateData)

if __name__ == "__main__":
    app.run(host='0.0.0.0', port=80, debug=True)
```



① 创建一个名为 `pins` 的字典对象，存放 GPIO 接口编号、名字和状态信息。

② 把每个 GPIO 接口都设为输出模式，并置为低电平。

③ 读取每个 GPIO 接口的状态，并把状态值存入 `pins` 字典中的对应项。

④ 把 `pins` 字典对象放入模板数据字典中。

⑤ 用模板数据字典渲染 `main.html` 页面并把结果返回给用户。

⑥ 当用户访问带有接口编号及操作的 URL 时，执行下面的函数。

⑦ 把 URL 中的接口编号转换为数值。

⑧ 获取发生变化的接口所对应的设备名称。

⑨ 如果 URL 中指定的操作是“on”，执行下面缩进的代码块。

⑩ 把 GPIO 接口状态设置为高电平。

⑪ 保存表示状态的字符串并把它传给页面模板。

⑫ 读取接口状态，把接口设置为与当前状态相反的状态（切换接口状态）。

⑬ 读取每一个接口的状态，并把它们的状态存入 `pins` 字典对象中。

⑭ 把 `pins` 字典对象与表示状态的字符串放入模板数据字典中。

代码 10.9 `templates/main.html` 的源代码

```
<!DOCTYPE html>
<head>
  <title>Current Status</title>
</head>

<body>
  <h1>Device Listing and Status</h1>

  {% for pin in pins %}❶
  <p>The {{ pins[pin].name }}❷
```



```
  {% if pins[pin].state == true %}③
    is currently on (<a href="/{{pin}}/off">turn off</a>)
  {% else %}④
    is currently off (<a href="/{{pin}}/on">turn on</a>)
  {% endif %}
  </p>
  {% endfor %}

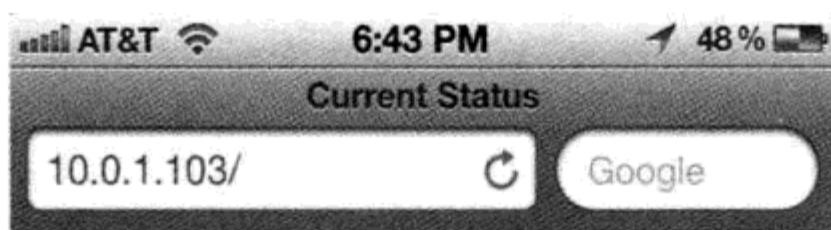
  {% if message %}⑤
  <h2>{{ message }}</h2>
  {% endif %}

</body>
</html>
```

- ① 遍历 `pins` 字典对象中的所有接口信息。
- ② 显示接口的名称。
- ③ 如果接口状态为高电平，显示对应的设备正处在打开状态，并显示一个链接，可以用于关闭设备。
- ④ 否则，显示对应的设备处在关闭状态，并显示一个链接，可以用于打开设备。
- ⑤ 如果向模板中传入了状态字符串，则把它显示出来，如图 10.2 所示。

这样设计这段代码的优点在于，增加要控制的设备时非常容易，只需把相应设备的信息加入 `pins` 字典对象中即可。当你重启服务器后，新添加的设备就会在状态列表中显示出来，并且相应的控制链接也可以正常使用。

这段代码还包含了一个很有用的功能：如果你需要切换某个设备的工作状态，只需在手机上点击一次就可以了。可以把 `http://ipaddress/pin/toggle` 这个 URL 加入浏览器的书签，访问这个 URL 会自动检查相应接口的状态并切换到相反的状态上。



Device Listing and Status

The coffee maker is currently on ([turn off](#))

The lamp is currently off ([turn on](#))

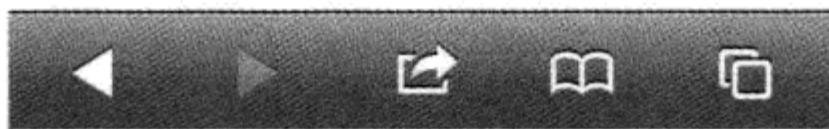


图10.2 从手机浏览器上看到设备控制界面

进一步学习

Requests 库

(<http://docs.python-requests.org/en/latest/>)

Requests 库的主页上提供了非常详尽的文档和容易理解的实例。

Flask

(<http://flask.pocoo.org/>)

本章内容中还有很多 Flask 的特性没有提到，它的官方网站上提供了它完整的功能列表。

Flask 扩展

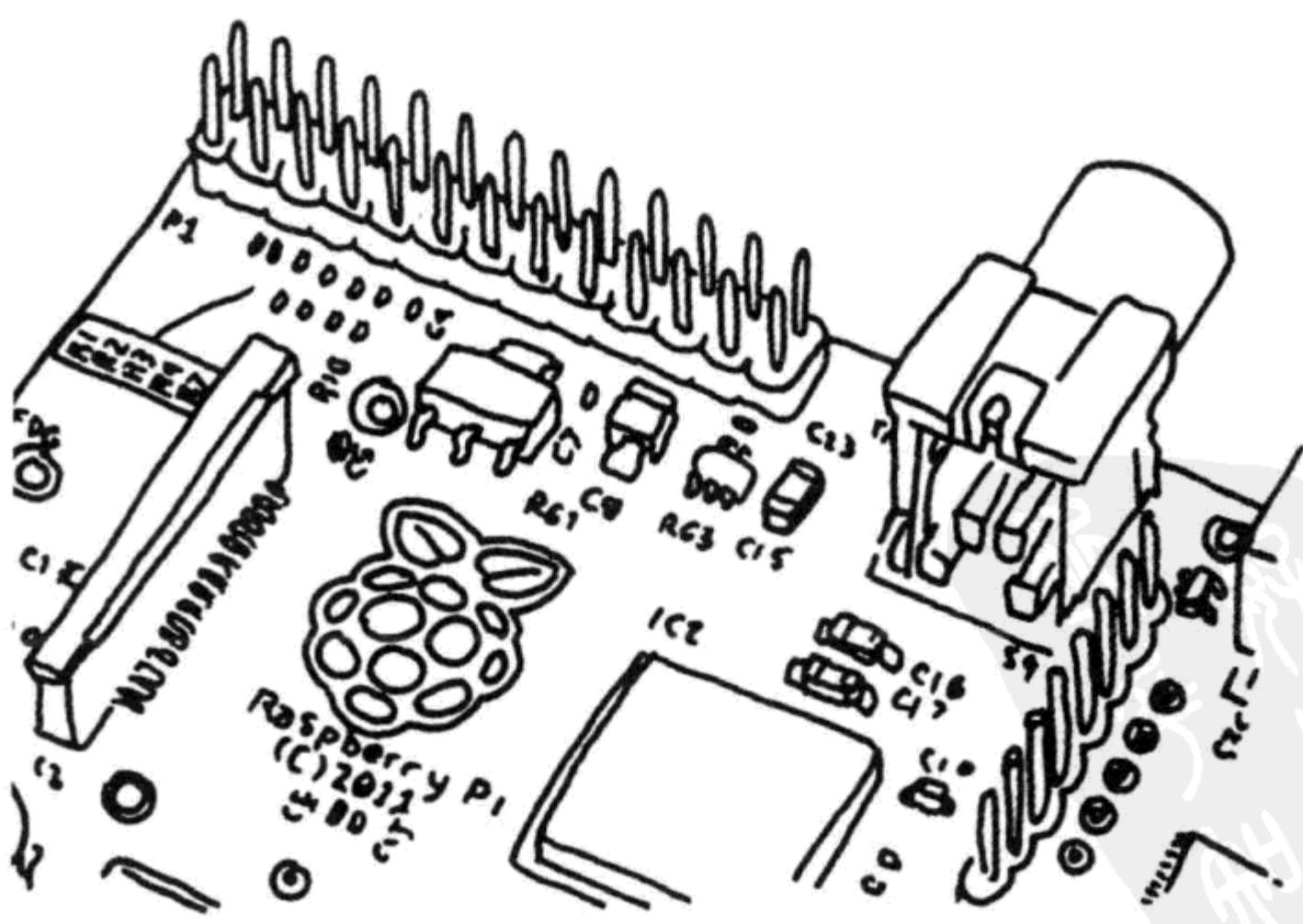
(<http://flask.pocoo.org/extensions/>)

通过使用 Flask 扩展，可以更方便地向你的网站上添加功能。

附录 A

烧录SD卡镜像

Writing an SD Card Image





虽然本书主要关注 Raspbian 操作系统，实际上 Raspberry Pi 上还可以运行其他很多种操作系统。如果要安装其他操作系统，只需下载相应的磁盘镜像并把它烧录到 SD 卡上。与下载相比，烧录 SD 卡的操作要更复杂一些。本章就会向你介绍如何在 OS X、Windows 或 Linux 下把磁盘镜像烧录到 SD 卡上。

在 OS X 中烧录 SD 卡

1. 打开终端工具(在 *Applications/Utilities* 中)，进入命令行环境。
2. 不要把 SD 卡插入读卡器，运行 `df -h`。`df` 程序会显示出你的各磁盘分区的剩余空间大小，并且会显示出当前已经挂载的磁盘分区。
3. 把 SD 卡插入读卡器，再次运行 `df -h`。
4. 通过与先前 `df` 命令的输出结果进行比较，判断哪一个分区是 SD 卡。例如，一张 SD 卡可能会挂载到系统的 *Volumes/Untitled* 目录中，而设备名称是 */dev/disk3s1*。根据你系统配置的不同，设备名称也可能会发生变化。设备名称是在设备被挂载时赋予的，所以如果你在 Finder 中挂载了其他设备或磁盘镜像，设备名称中的数字可能会变得更大。记录下你的 SD 卡对应的设备名称。
5. 你需要先把 SD 卡从系统中卸载后才能开始磁盘镜像文件的烧录。输入 `sudo diskutil umount /dev/disk3s1` (用上一步中记录下来的设备名称替换这条命令中的 */dev/disk3s1*)。注意，你必须使用命令行或 Disk Utility 来卸载。如果你在 Finder 中直接弹出设备，就得把 SD 卡重插一遍（而且你仍然需要用命令行



或 Disk Utility 来卸载它）。如果卸载过程失败了，请确认你关闭了所有可能正在访问 SD 卡的 Finder 窗口。

6. 下一步，你需要得知 SD 卡的原始设备名称。把刚才的设备名称中的 disk 改为 rdisk 并去掉 s1 后缀（这个后缀表示的是分区号）。例如，/dev/disk3s1 的原始设备名为 /dev/rdisk3。



获取正确的原始设备名极为重要！如果误把镜像文件烧录到你的硬盘上而不是 SD 卡上，你会丢失你硬盘上的所有数据。你可以再次执行 `df` 命令确认是否正确地找到了原始设备名。

7. 将下载的镜像文件解压缩，存放在你的主目录中，然后使用 UNIX 的 `dd` 命令把这个镜像文件原样写入 SD 卡。把下面的命令中的磁盘镜像文件的名称换成你所下载的文件名称，并把 /dev/rdisk3 改成你在第 6 步中获取的 SD 卡的原始设备名称。

第 3 章中对各种命令行工具有更多的介绍，不过在这里，你只需理解 `dd` 命令需要以 `root` 权限执行，它会把输入文件（由 `if` 参数指定）写入输出文件中（由 `of` 参数指定）。

```
sudo dd bs=1m if=~/2012-09-18-wheezy-raspbian.img of=/dev/rdisk3
```

8. 整个烧录过程需要几分钟才能完成。由于 `dd` 命令在运行过程中不会输出任何进度信息，所以你只能等待它执行完成。当 `dd` 命令运行结束后，它会显示出一些统计信息。这时，就可以把 SD 卡从系统中弹出，它已经可以放在 Pi 上使用了。

在 Windows 中烧录 SD 卡

1. 下载 Win32DiskImager (<https://launchpad.net/win32-image-writer>)



程序。

2. 把 SD 卡插入读卡器，观察资源管理器（Windows Explorer）弹出的提示，记录下对应的盘符。

3. 打开 Win32DiskImager 程序并打开 Raspbian 的磁盘镜像文件。

4. 选择 SD 卡对应的盘符，然后点击 Write 按钮。如果 Win32DiskImager 无法正常写入 SD 卡，可以先尝试一下在资源管理器（Windows Explorer）中把 SD 卡格式化一下。

5. 从读卡器中弹出 SD 卡并插入 Raspberry Pi，它应该已经可以正常使用了。

在 Linux 中烧录 SD 卡

在 Linux 下的操作方式与在 Mac 下非常相似：

1. 不要把 SD 卡插入读卡器，打开一个新的命令行终端，运行 `df -h` 观察系统中挂载了哪些磁盘分区。

2. 插入 SD 卡，再次运行 `df -h`。

3. 通过与先前 `df -h` 命令的输出结果进行比较，判断哪一个分区是 SD 卡上的分区。SD 卡对应的设备名称应该是 `/dev/sdd1` 这种形式的。根据你系统配置的不同，设备名称也可能会发生变化。记录下这个设备名称。

4. 你需要先把 SD 卡从系统中卸载后才能开始磁盘镜像文件的烧录。输入 `sudo umount /dev/sdd1`（用上一步中记录下来的设备名称替换这条命令中的 `/dev/sdd1`）来卸载它。如果无法正常卸载，请确认所有打开的终端中都没有把 SD 卡所挂载的目录作为当前工作目录。

5. 下一步，你需要得知 SD 卡的原始设备名称，把设备名称中的分区号去除后就得到了原始设备名称。例如，SD 卡分区的设备



名称是 `/dev/sdd1`，则原始设备名称为 `/dev/sdd`。



获取正确的原始设备名极为重要！如果误把镜像文件烧录到你的硬盘上而不是 SD 卡上，你会丢失你硬盘上的所有数据。你可以再次执行 `df` 命令确认是否正确地找到了原始设备名。

6. 将下载下来的镜像文件解压缩，存放在你的主目录中。然后使用 UNIX 的 `dd` 命令把这个镜像文件原样写入 SD 卡。把下面命令中的磁盘镜像文件的名称换成你所下载的文件名称，并把 `/dev/sdd` 改成你在第 5 步中获取到的 SD 卡的原始设备名称。

```
sudo dd bs=1M if=~/2012-09-18-wheezy-raspbian.img of=/dev/sdd
```

这条命令通过 `root` 权限来运行 `dd` 命令，把输入文件（由 `if` 参数指定）写入输出文件中（由 `of` 参数指定）。

7. 整个烧录过程需要几分钟才能完成。由于 `dd` 命令在运行过程中不会输出任何进度信息，所以你只能等待它执行完成。当 `dd` 命令运行结束后，它会显示出一些统计信息。这时，就可以把 SD 卡从系统中弹出。不过为了保险起见，可以运行一下 `sudo sync` 命令，确保所有的数据都被正确地从系统缓存中写回到 SD 卡上。

8. 从读卡器中弹出 SD 卡并插入 Raspberry Pi，它应该已经可以正常使用了。

BerryBoot

另外一种向 SD 卡上安装操作系统的方式是使用 BerryBoot 工具 (<http://www.berryterminal.com/doku.php/berryboot>)。

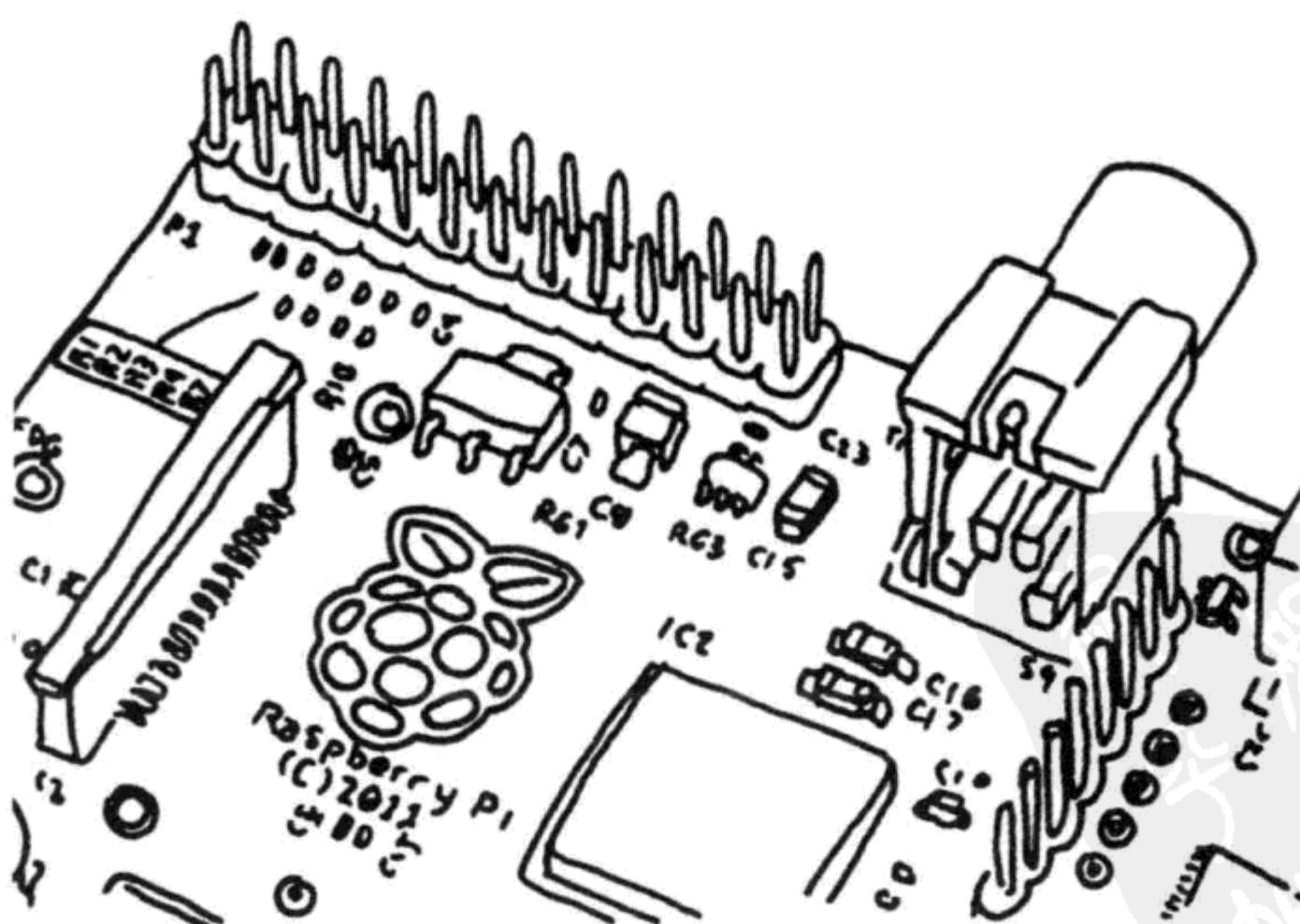


BerryBoot 是 BerryTerminal 精简客户端项目的一部分，它提供了在一张 SD 卡安装多个操作系统的功能。把 BerryBoot 的安装镜像写入 SD 卡，然后用这张 SD 卡启动 Raspberry Pi，这时系统会显示一个交互式的安装界面，让你从列表中选择要安装的系统。注意，BerryBoot 需要接入网络才能正常工作。

附录 B

星际入侵者游戏 完整版

Astral Trespassers Complete



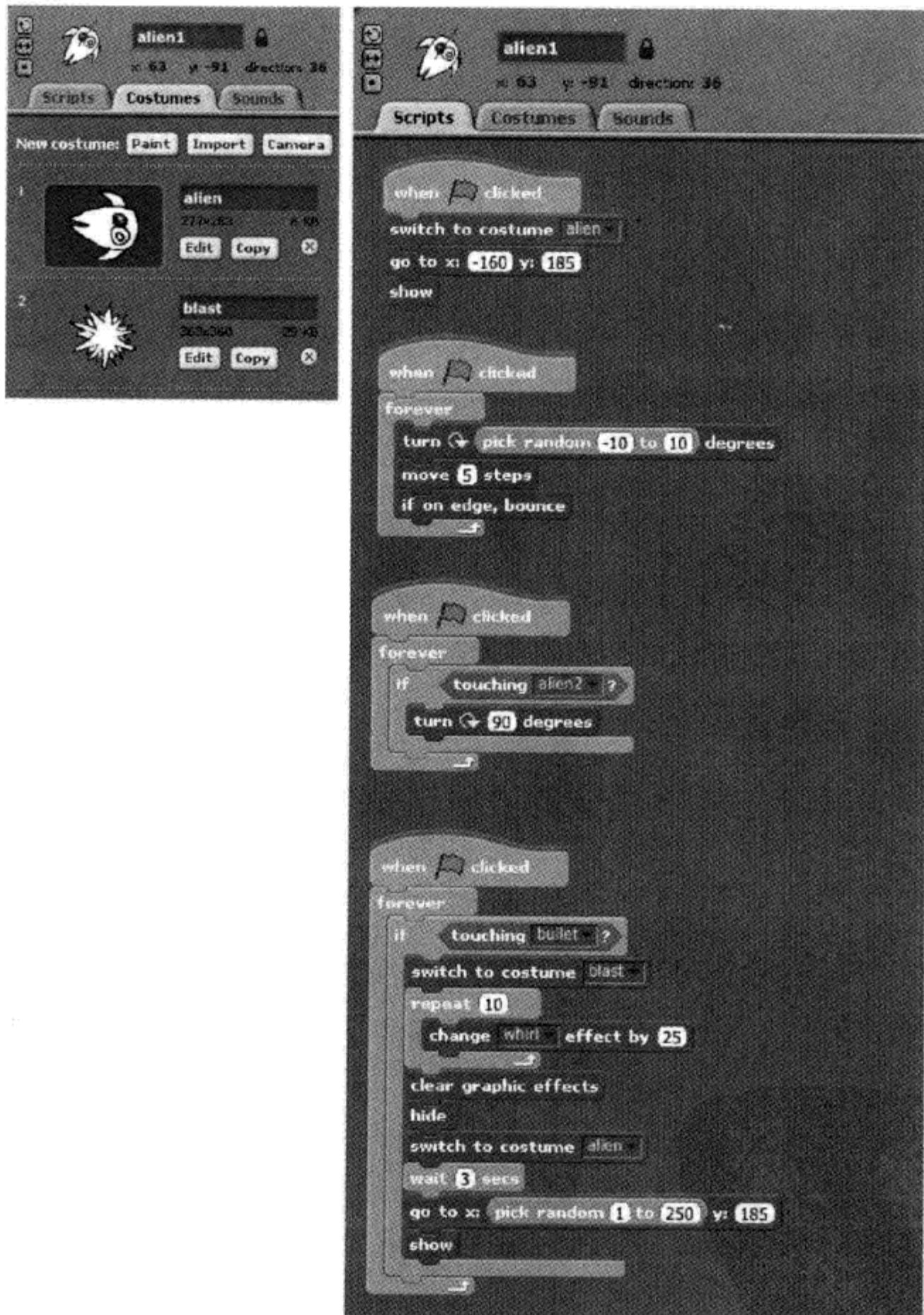
PDG



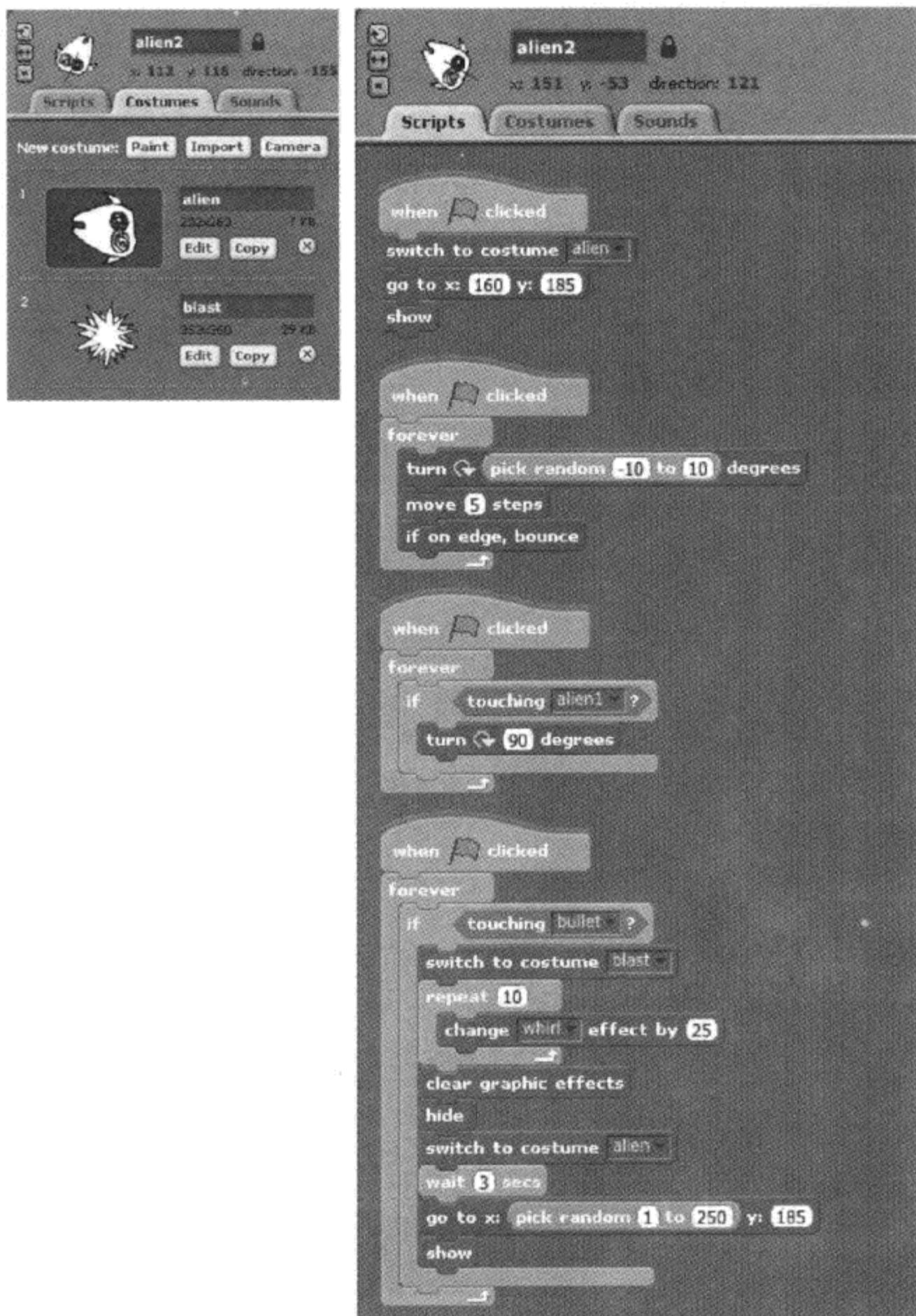
本附录中提供了第5章中的星际入侵者游戏中所有角色的所有相关脚本。第5章中其实已经提供了完整的程序，不过由于Scratch是一种图形化编程语言，所以把所有的脚本都整理在一起看起来可能会更清楚一些。每个角色的各个造型在这里也有完整的展示，如图B.1~图B.6所示。



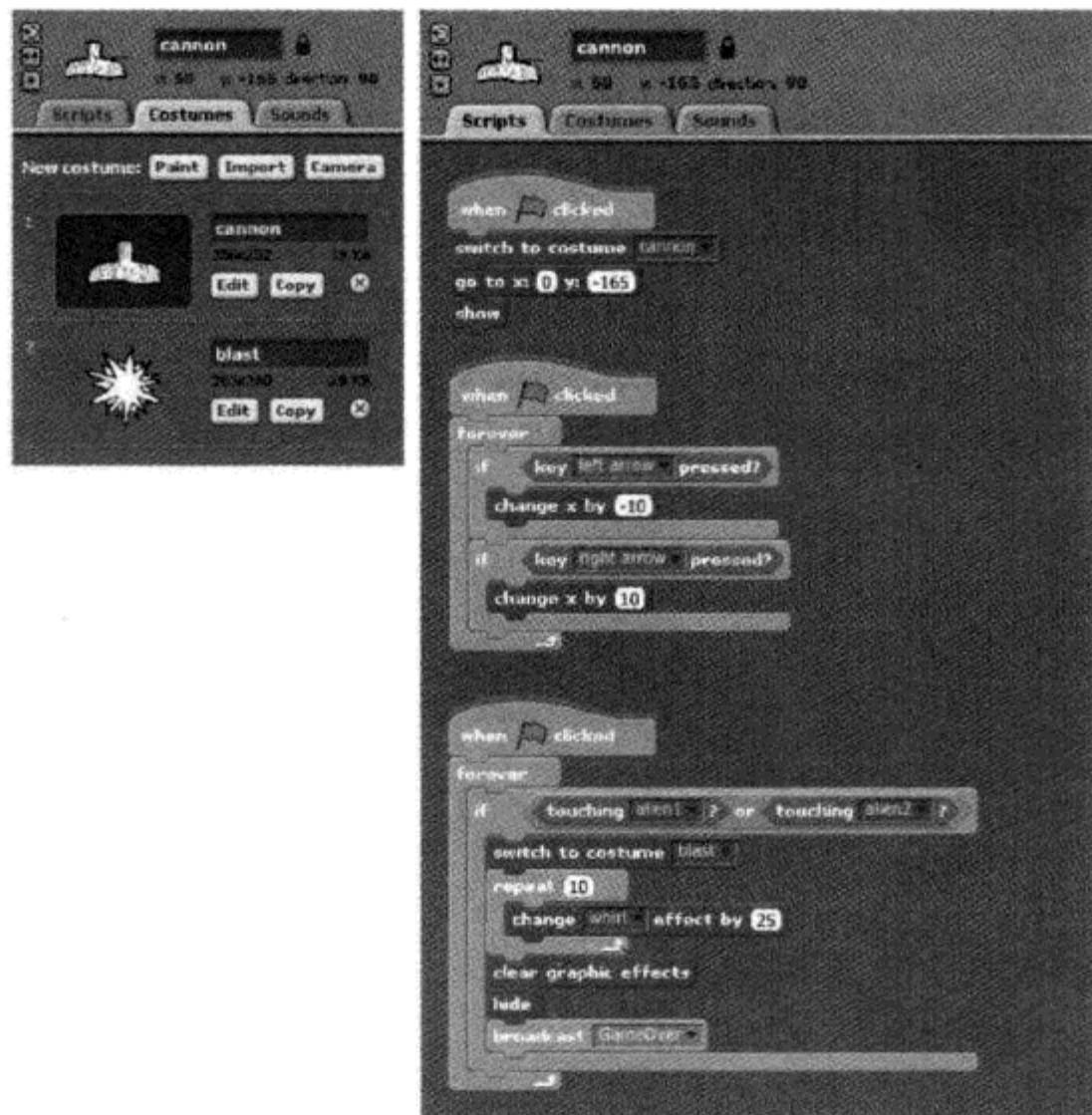
图B.1 角色列表与5个角色



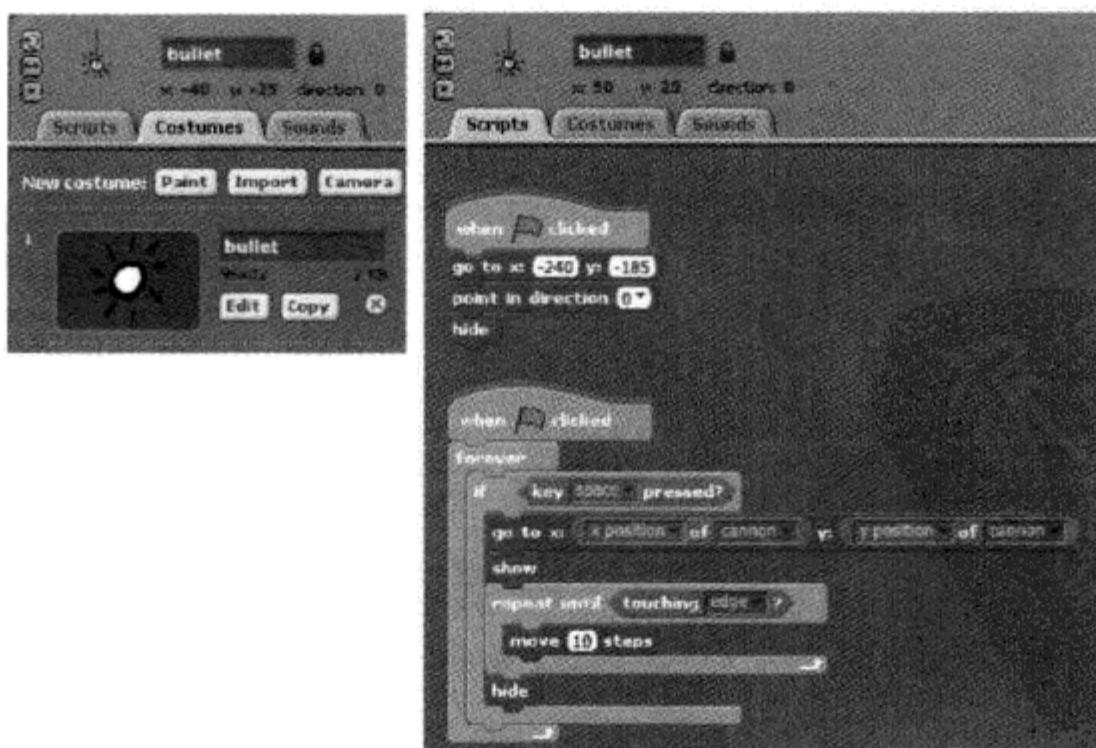
图B.2 第一艘外星飞船角色的造型（左）和完整的脚本（右）



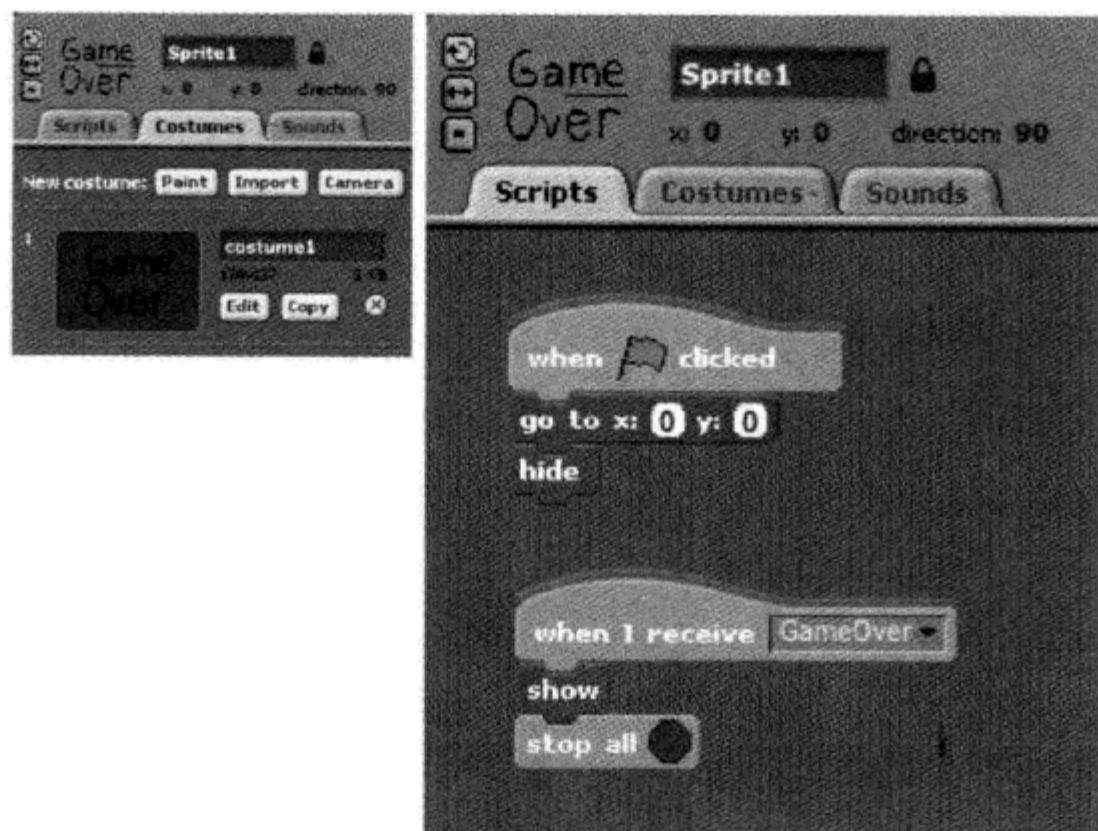
图B.3 第二艘外星飞船角色的造型和完整的脚本



图B.4 加农炮角色的造型和完整的脚本



图B.5 子弹角色的造型和完整的脚本

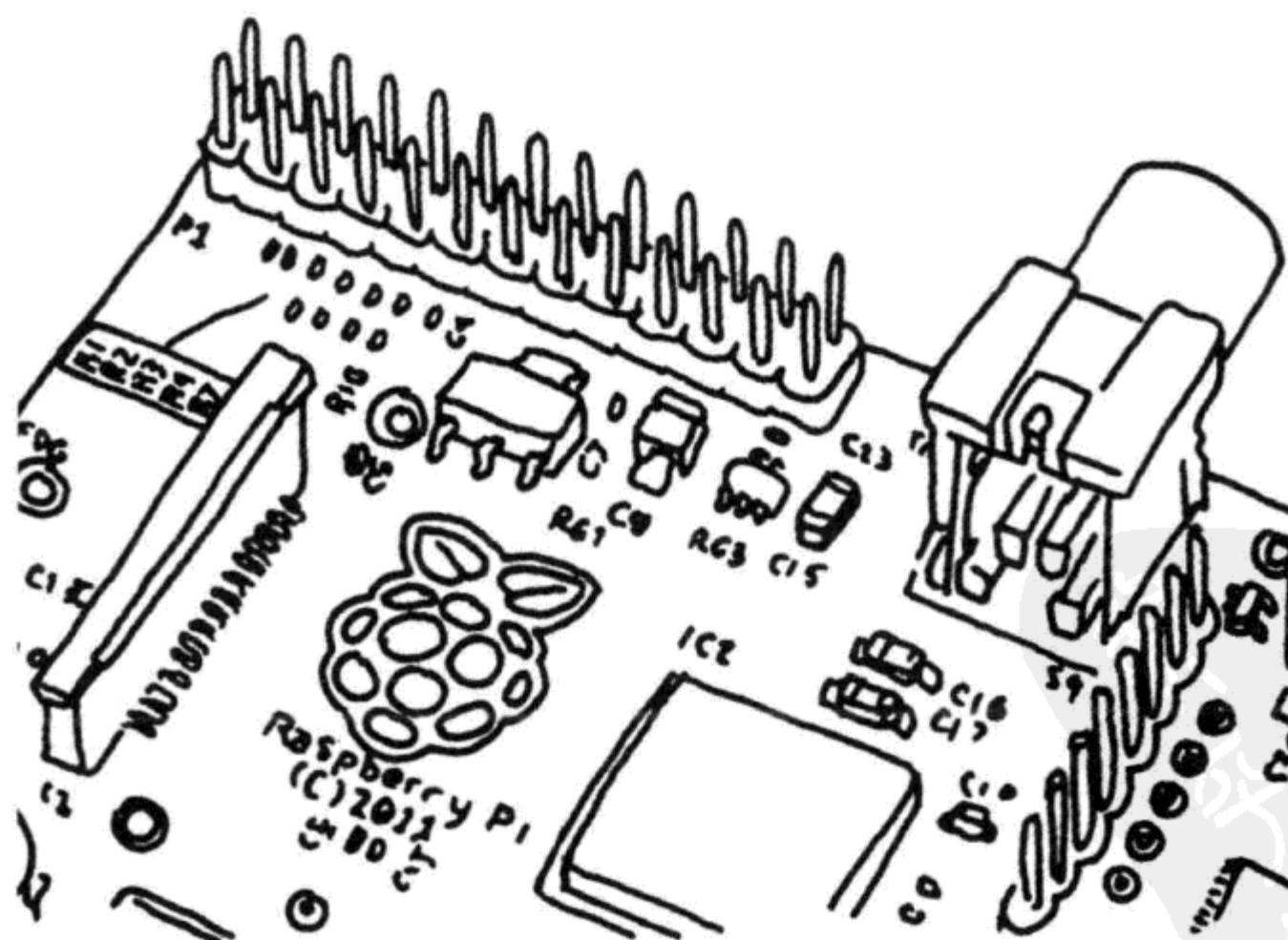


图B.6 游戏结束角色的造型和完整的脚本

附录 C

模拟信号输入

Analog Input



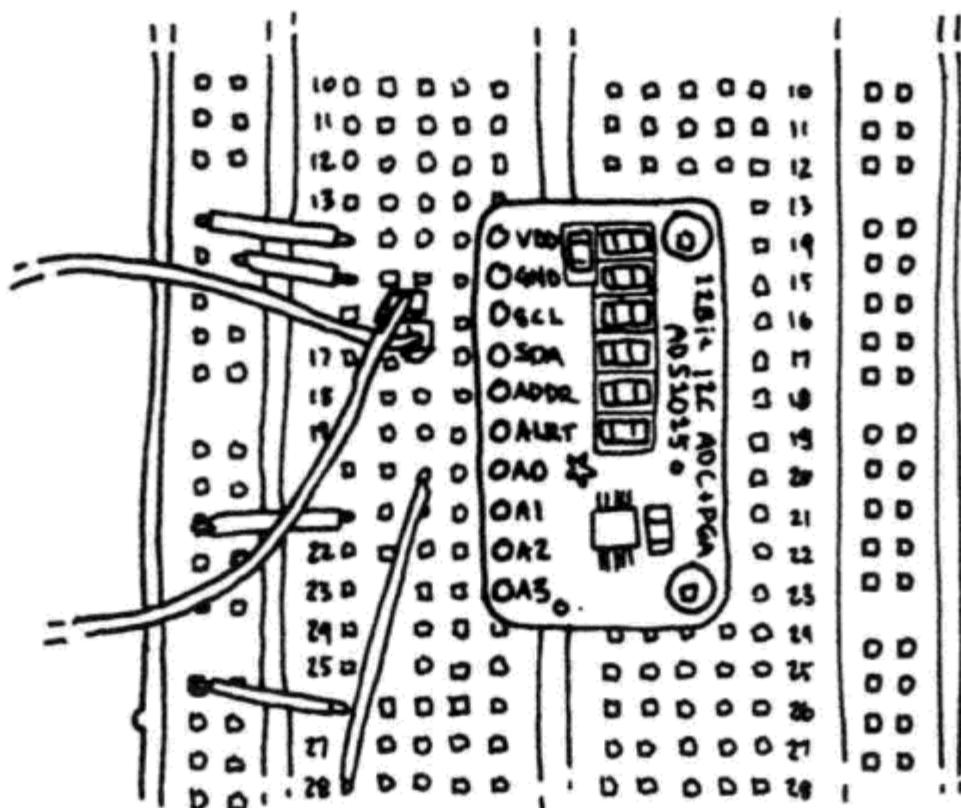


在本书中，你学到了如何用数字信号输入输出接口来连接按钮、开关、LED 和继电器。所有的这些器件都只有开或关两种状态，不会有中间状态。但是，现实中你可能需要去获取一些不是只有开、关两种状态的传感器信号，如温度、距离、亮度或旋钮的读数。

不幸的是，你不能直接用 Raspberry Pi 来读取这些传感器的值，因为 Raspberry Pi 的主板上只有数字信号的输入输出接口。但是，也不是完全没有办法解决这个问题，因为有很多的电路都可以帮助你实现用数字输入接口来读取模拟信号值的功能。

把模拟信号转换为数字信号

本附录将介绍一种通过使用 ADC（模数转换器，Analog to Digital Converter）把模拟信号转换为数字信号的方法。市面上 ADC 的种类很多，我们在这里选用 TI（德州仪器，Texas Instruments）的 ADS1015。ADS1015 的芯片封装方式不适合直接安装在面包板上，所以 Adafruit Industries 公司定制了一块扩展板 (<http://www.adafruit.com/products/1083>) 来解决这个问题，如图 C.1 所示。当你把插针都焊接到这个扩展板上以后，就可以把它直接插到面包板上使用。这块芯片使用 I²C 协议来传送它所获取到的模拟信号。我们并不需要完全理解这个协议就可以直接使用它，Adafruit 提供了一个很好的开源 Python 库，可以直接通过 I²C 从 ADS1015 及与之类似的 ADS1115 上读取数据。



图C.1 Adafruit的ADS1015模数转换器扩展板

用下面的方法连接 ADS1015 与你的 Raspberry Pi。

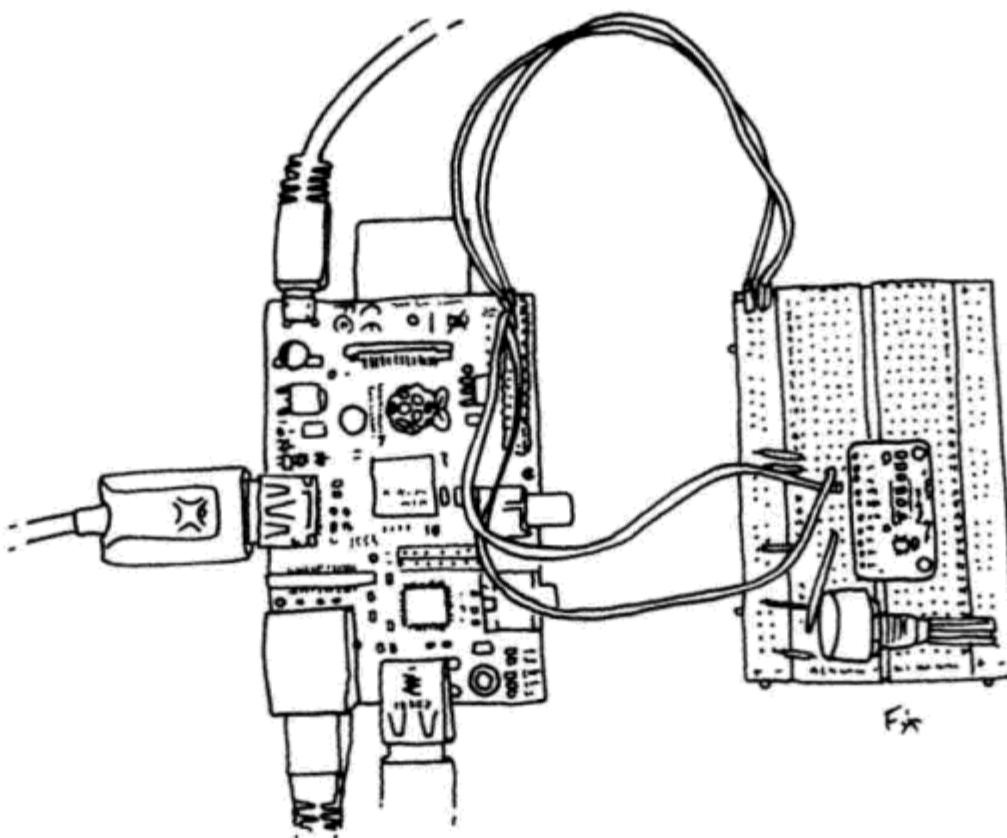
1. 把 Raspberry Pi 的 3.3V 电源输出与面包板的供电总线的正极相连。参考图 7.2 了解 Raspberry Pi 上的 GPIO 接口定义。
 2. 把 Raspberry Pi 的接地接口与面包板的供电总线的负极相连。
 3. 把 ADS1015 扩展板接入面包板，用连接线把 VDD 引脚与供电总线的正极相连，把 GND 引脚与供电总线的负极相连。
 4. 把 ADS1015 的 SCL 引脚与 Raspberry Pi 的 SCL 接口相连。Raspberry Pi 上 SCL 接口在 GPIO 接口中的一个接地接口的旁边。
 5. 把 ADS1015 的 SDA 引脚与 Raspberry Pi 的 SDA 接口相连。Raspberry Pi 上的 SDA 接口在 3.3V 电源接口与 SCL 接口之间。

下面，你就可以把以模拟信号为输出的传感器与 ADS1015 相连了。虽然有很多种传感器可以使用，但在我们的实验中，使用一个简单的 $2k\Omega$ 电位器，用它来作为 Raspberry Pi 上所连接的一个旋钮。所谓电位器，其实就是一个以旋钮或滑杆的形式出现的可变电阻。下面是把电位器与 ADS1015 相连的方法。



1. 把电位器插到面包板上。
2. 电位器有3个引脚, 把中间的引脚与ADS1015的A0引脚相连。
3. 把电位器另外两个引脚中的任意一个与面包板供电总线的正极相连。
4. 把电位器上剩下的引脚与面包板供电总线的负极相连。

电路连接完以后, 如图 C.2 所示。



图C.2 通过ADS1015把电位器与Raspberry Pi相连

在可以真正开始读取电位器信号前, 你还需要启用 I²C 并安装一些相关的库。

1. 在命令行上, 用文本编辑器以 root 权限打开 *raspi-blacklist.conf* 文件:

```
pi@raspberrypi ~ $ sudo nano /etc/modprobe.d/raspi-blacklist.conf
```

把 I²C 相关模块从黑名单中去除, 在 *blacklist i2c-bcm2708*



这一行前面加上一个#号，使这个文件如下所示：

```
# blacklist spi and i2c by default (many users don't need them)
blacklist spi-bcm2708
#blacklist i2c-bcm2708
```

2. 按 Control-X 键退出编辑器并按 Y 键保存文件。
3. 下一步，打开 */etc/modules* 文件：

```
pi@raspberrypi ~ $ sudo nano /etc/modules
```

4. 在文件末尾加上 *i2c-dev*，独占一行。文件如下所示：

```
# /etc/modules: kernel modules to load at boot time.
#
# This file contains the names of kernel modules that
should be loaded
# at boot time, one per line. Lines beginning with "#" are
ignored.
# Parameters can be specified after the module name.
snd-bcm2835
i2c-dev
```

5. 按 Control-X 键退出编辑器并按 Y 键保存文件。
6. 更新包列表：

```
pi@raspberrypi ~ $ sudo apt-get update
```

7. 安装 *i2c-tools* 工具与 *python-smbus* 包：

```
pi@raspberrypi ~ $ sudo apt-get install i2c-tools python-smbus
```

8. 重启 Raspberry Pi。
9. 在重启完 Raspberry Pi 后，用下面的方法来判断 Raspberry Pi



是否已经识别出 ADS1015。如果是第一版的 Raspberry Pi，用如下的命令：

```
pi@raspberrypi ~ $ sudo i2cdetect -y 0
```

如果是第二版的 Raspberry Pi，用如下的命令：

```
pi@raspberrypi ~ $ sudo i2cdetect -y 1
```

10. 如果 ADS1015 扩展板已经被正确地识别出来，你会看到如下的表格，里面显示了一些数字：

	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
00:																
10:	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
20:	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
30:	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
40:	--	--	--	--	--	--	--	--	48	--	--	--	--	--	--	--
50:	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
60:	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
70:	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--

11. 现在，我们可以确认电路已经正确连接，并被 Raspberry Pi 正确识别，下面就可以尝试读取电位器的输入值了。从 Adafruit 提供的代码库中下载供 Raspberry Pi 使用的 Python 库并存入你的主目录。在命令行上运行下面的命令，在同一行上运行整个命令，URL 中没有空格：

```
wget https://github.com/adafruit/Adafruit-Raspberry-Pi-Python-Code/archive/master.zip
```

12. 把它解压：

```
pi@raspberrypi ~ $ unzip master.zip
```



13. 切换到这个库的 *ADS1x15* 目录下：

```
$ cd Adafruit-Raspberry-Pi-Python-Code-master/Adafruit_ADS1x15
```

14. 运行示例程序：

```
$ sudo python ads1015_example.py
Channel 0 = 2.067 V
Channel 1 = 3.309 V
```

15. 把电位器朝某个方向旋转到底，再次运行这个程序。请注意，*Channel 0* 的值发生了变化：

```
$ sudo python ads1015_example.py
Channel 0 = 3.306 V
Channel 1 = 3.309
```

16. 把电位器朝反方向旋转到底，再次运行程序：

```
$ sudo python ads1015_example.py
Channel 0 = 0.000 V
Channel 1 = 3.309 V
```

正如你所看到的，旋转电位器的旋钮改变了向 *ADS1015* 的 0 号通道输入的电压值。下面的 *ads1015_example.py* 示例程序在从 ADC 上读取输入的电压值的基础上进行了一些计算。当然，如果你使用不同的传感器来输入数据，程序中要进行的计算也很可能不同。

在当前目录下创建一个新的文件，输入以下代码：

```
from Adafruit_ADS1x15 import ADS1x15❶
from time import sleep
```



```
adc = ADS1x15()①

while True:
    result = adc.readADCSingleEnded(0)②
    print result
    sleep(.5)
```

① 导入 Adafruit 的 ADS1x15 库。

② 创建一个 ADS1x15 的对象，名为 `adc`。

③ 从 ADS1015 的 A0 通道上读取值并存入 `result` 变量。

当你以 `root` 权限运行这段代码时，程序会以每秒两次的频率显示出从 ADS1015 上读取到的输入值。旋转电位器的旋钮会使这个值变大或变小。

只要你完成了最初的设计，ADS1x15 库就会自动完成读取模拟信号所需的复杂工作，在你的项目中使用输出模拟信号的传感器就会变得很容易。例如，你想要编写一个类似于 Pong^① 的游戏，你可以读取两个电位器的状态并用 Pygame 在屏幕上绘制相应的图形。有关使用 Pygame 的方法，可以参考第 4 章。

Adafruit 的教学用 Linux 发行版

Adafruit 工业公司从 Raspbian/Wheezy Linux 上创建了一个分支，向这个分支中加入了一些驱动程序和软件，使得完成电子项目更为方便。这就是 Adafruit Raspberry Pi 教学版 Linux (<http://learn.adafruit.com/adafruit-raspberry-pi-educational-linux-distro>)，它的代码名称是 *Occidentalis*，来源于黑树莓的物种名称。

① Pong 是 Atari 公司于 1972 年推出的一款街机游戏，模拟了乒乓球。这款游戏通常被认为是历史上第一款街机游戏。——译者注



Occidentalis 是开源软件开发模式中的一个示例：Linux 生态系统一直采用多分支多发行版的模式，每一个分支可以为某一种特殊的应用场景进行相关的优化。这些分支版本可以用来验证所进行的优化的有效性，以便以后可以把相关的改动合并入其他分支。

目前 *Occidentalis* 中的改进和增加的模块主要是为了做到系统无须安装额外软件就可以直接支持一些常见的传感器以及 Adafruit 的一些产品。同时，它也提供了一些底层的改进，使得用 PWM (Pulse Width Modulation，脉冲宽度调制) 变得更为方便，并且还可以实现通过 Raspberry Pi 直接控制伺服电机。

你可以参考项目概述页面 (<http://learn.adafruit.com/adafruit-raspberry-pi-educational-linux-distro>) 了解 *Occidentalis* 发行版的完整特性，还可以学习 Adafruit 上的各种教程 (<http://learn.adafruit.com/category/raspberry-pi>) 来了解如何使用 *Occidentalis*。