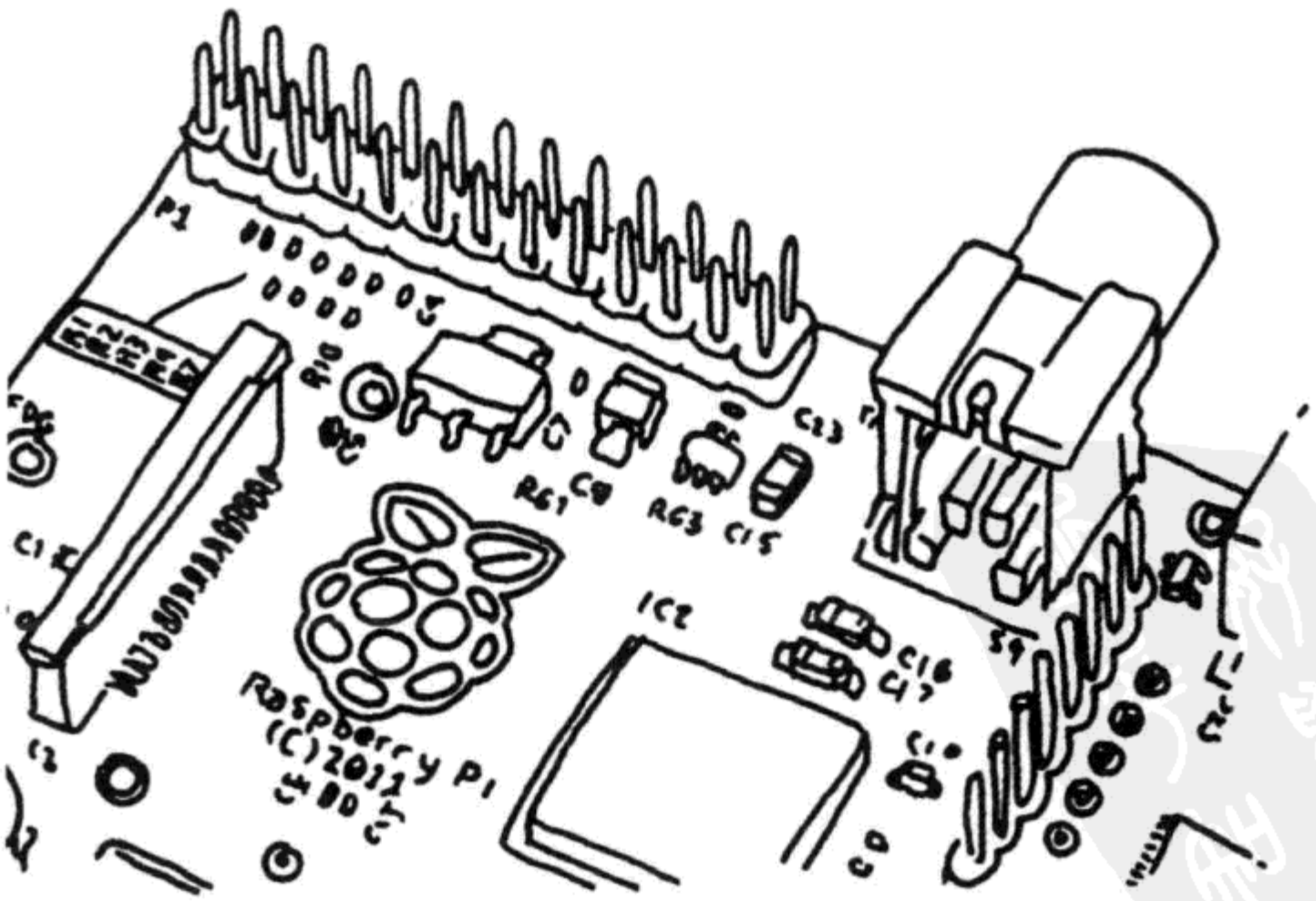
附录 A

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。

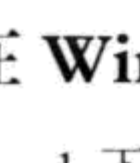
3. 把 SD 卡插入读卡器，再次运行 `df -h`。

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*/Volumns/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

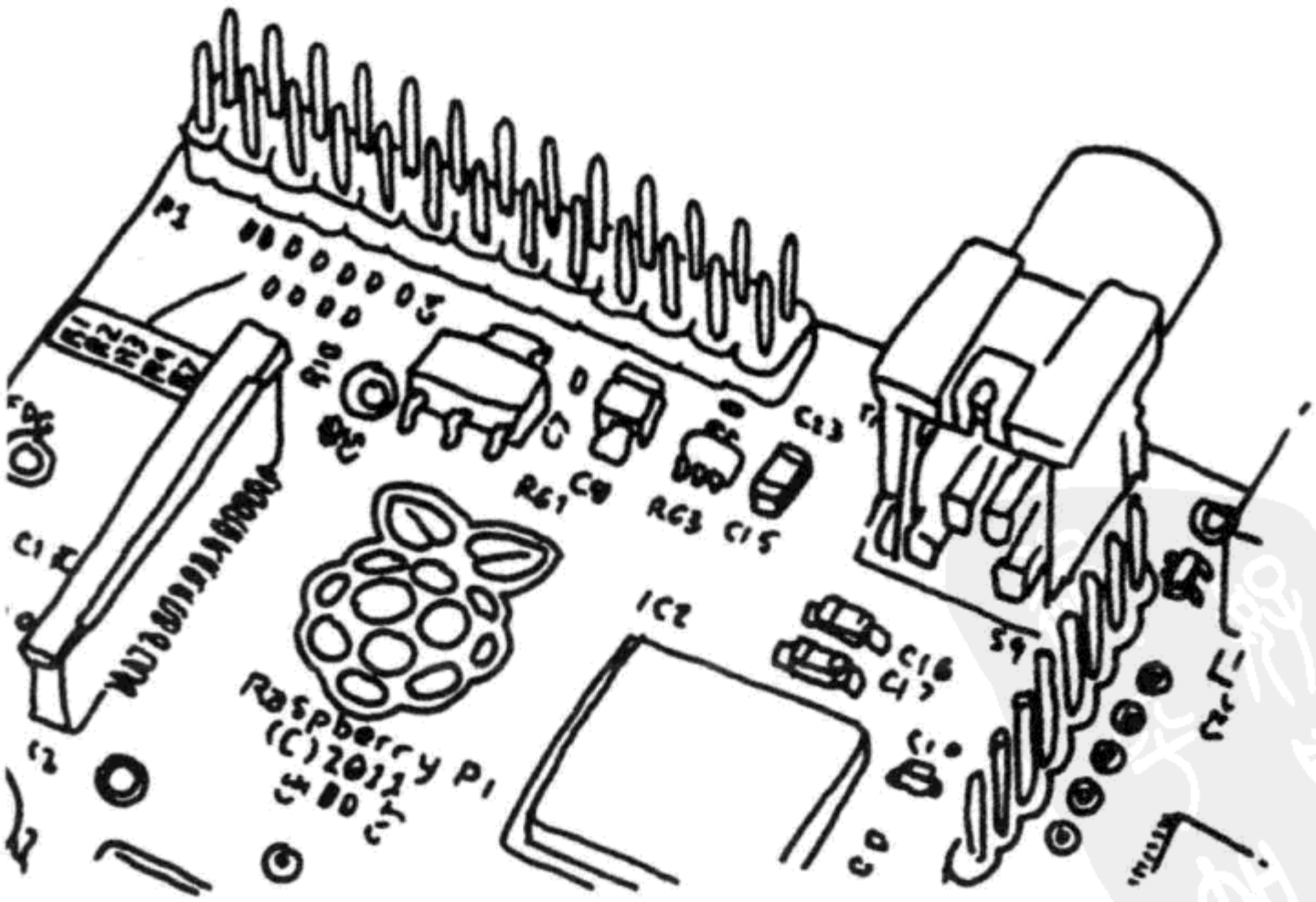
另外一种向 SD 卡上安装操作系统的方式是使用 BerryBoot 工　具（*http://www.berryterminal.com/doku.php/berryboot*）。

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能力。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

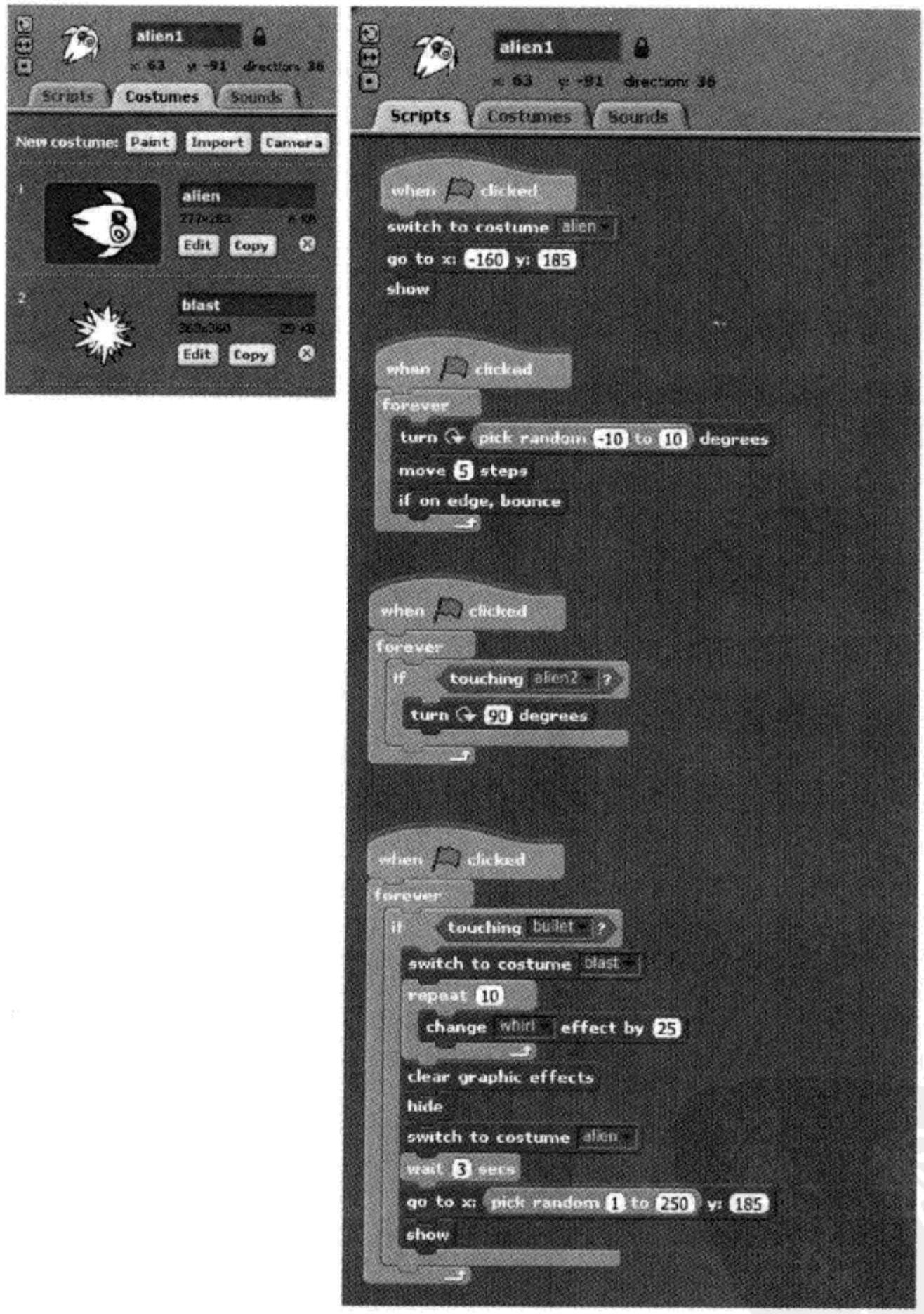
星际入侵者游戏完整版

Astral Trespassers Complete

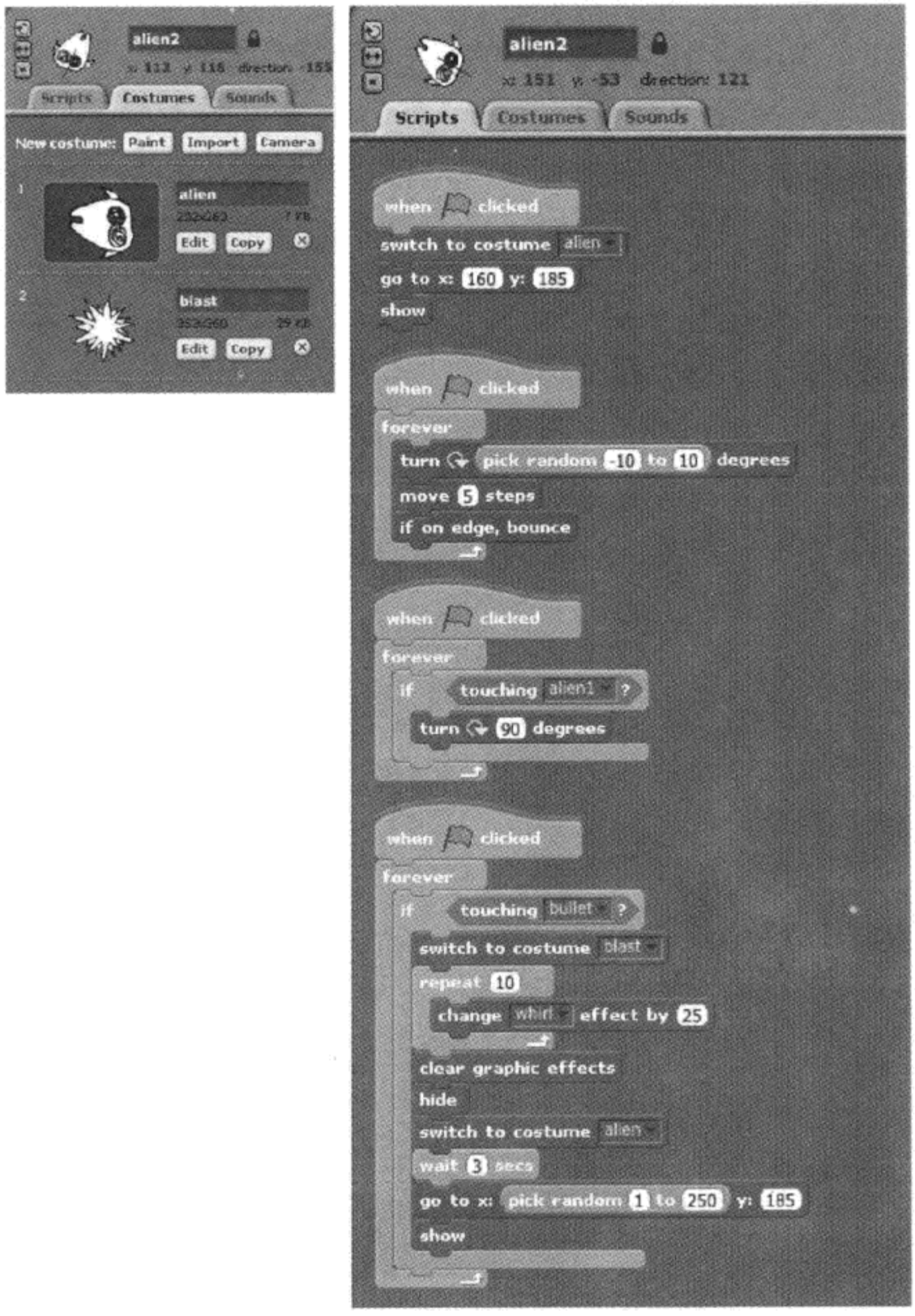


本附录中提供了第 5 章中的星际入侵者游戏中所有角色的所有相关脚本。第 5 章中其实已经提供了完整的程序，不过由于 Scratch 是一种图形化编程语言，所以把所有的脚本都整理在一起看起来可能会更清楚一些。每个角色的各个造型在这里也有完整的展示，如图 B.1 ~ 图 B.6 所示。

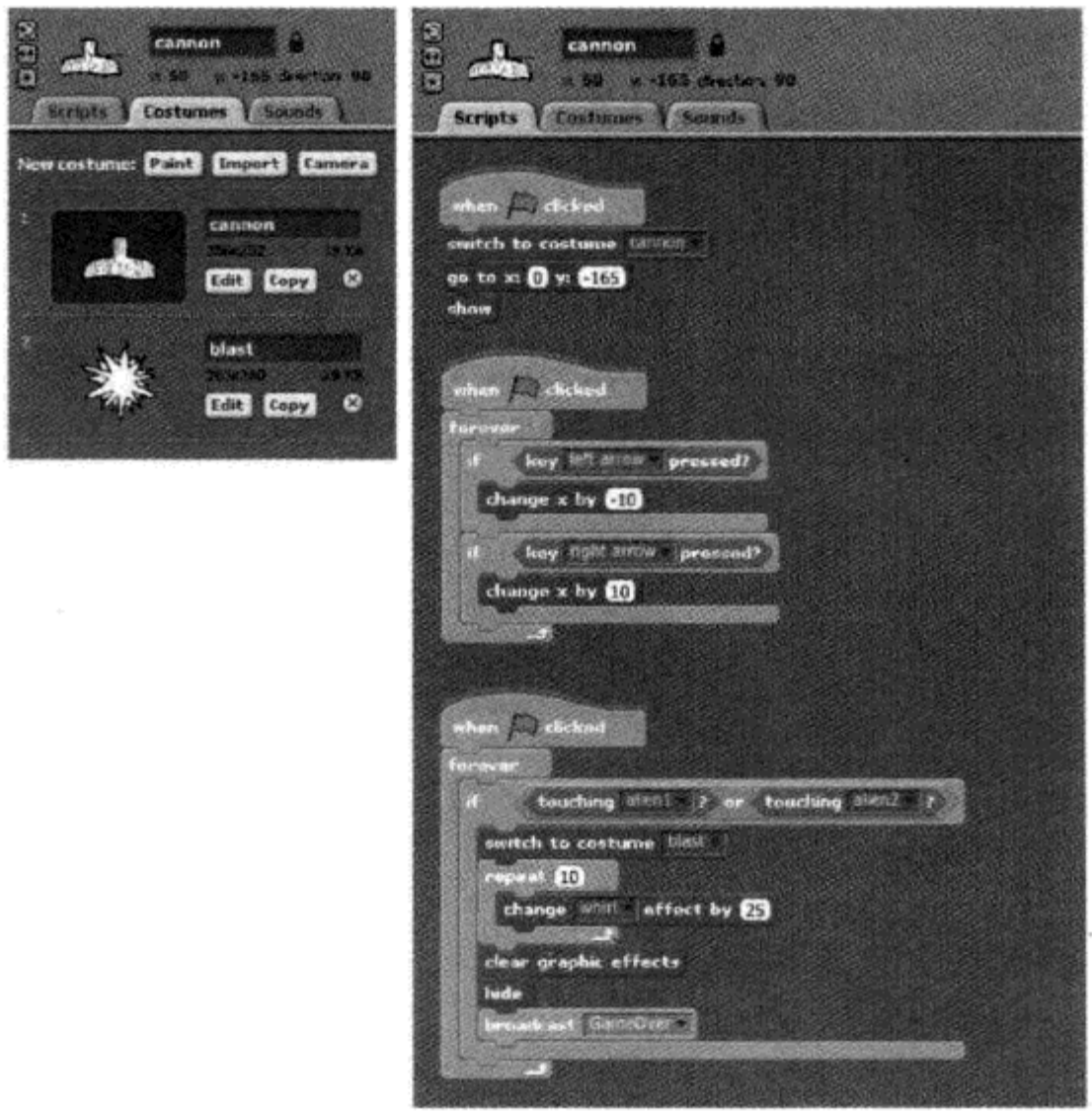
图B.1　角色列表与5个角色



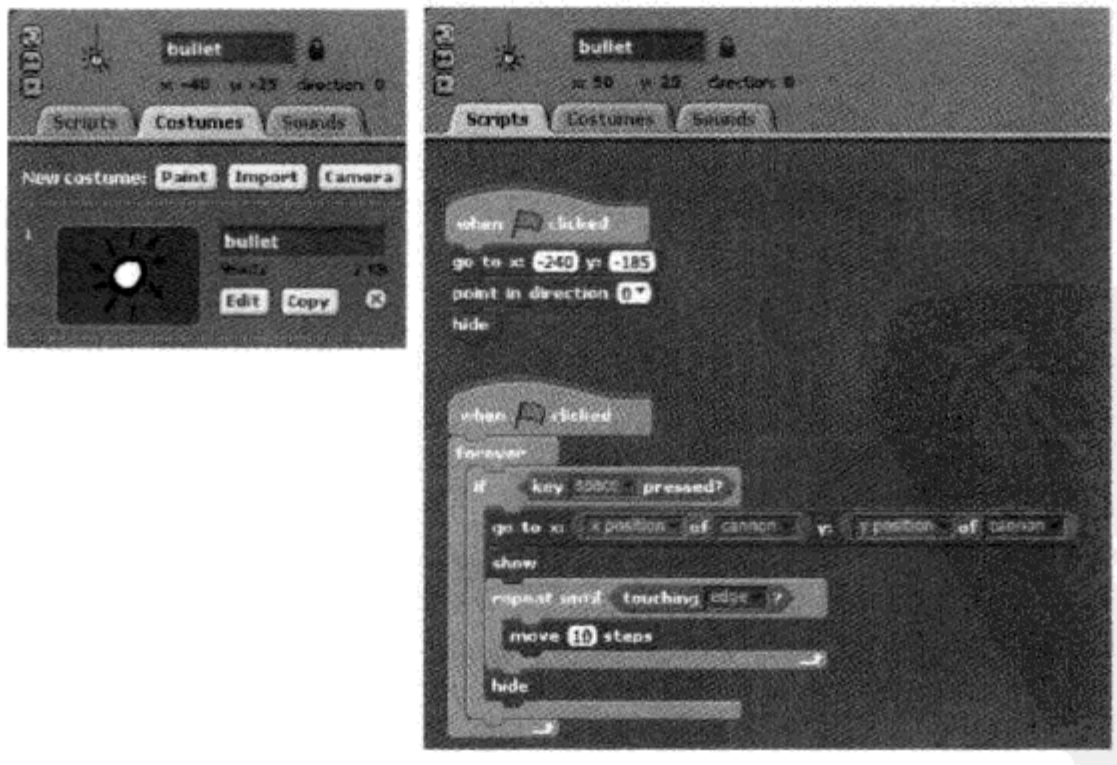
图B.2　第一艘外星飞船角色的造型（左）和完整的脚本（右）



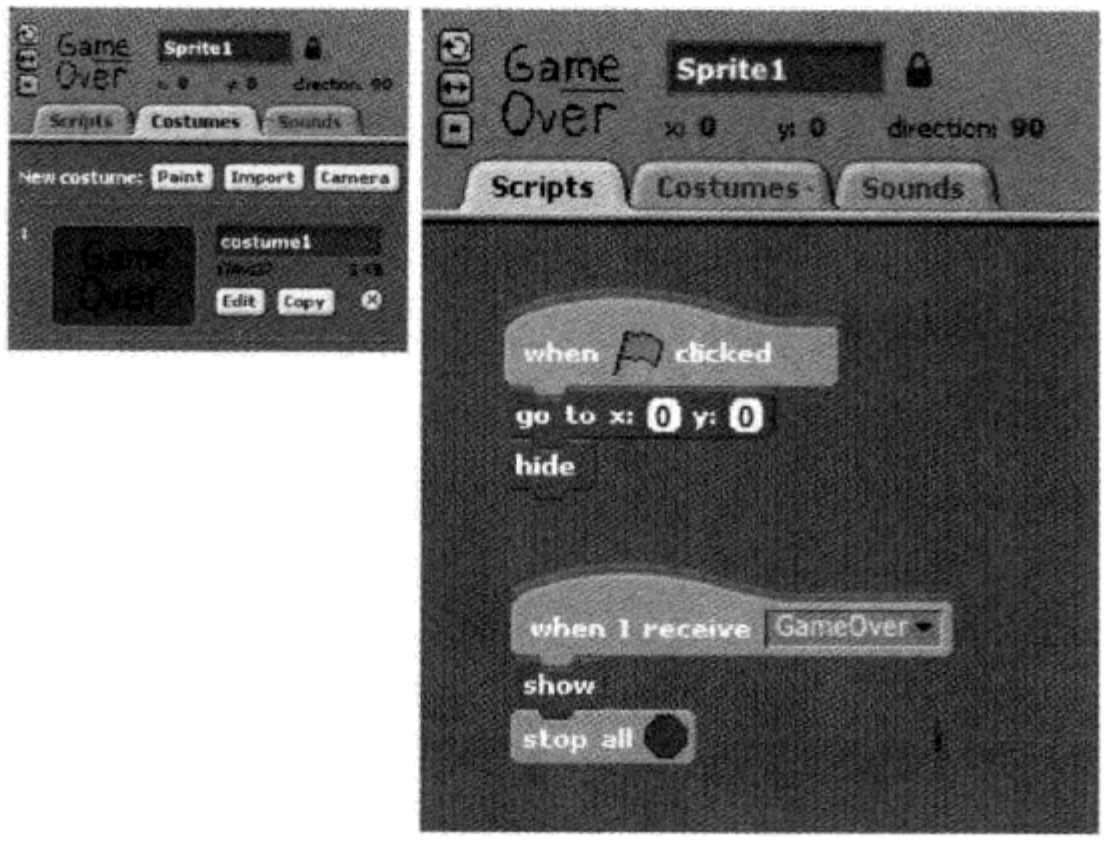
图B.3　第二艘外星飞船角色的造型和完整的脚本



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

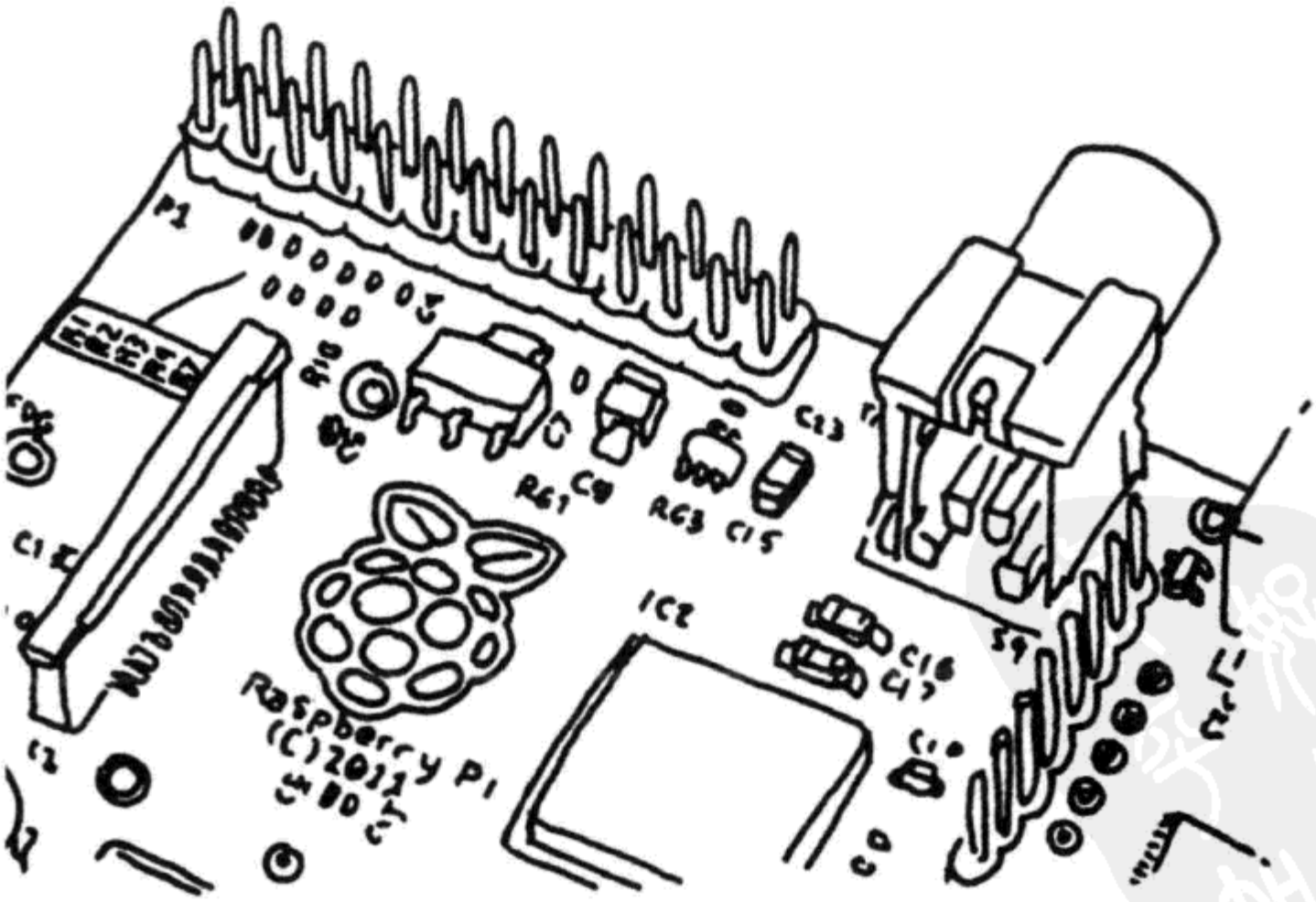


图B.6　游戏结束角色的造型和完整的脚本

附录 C

模拟信号输入

Analog Input

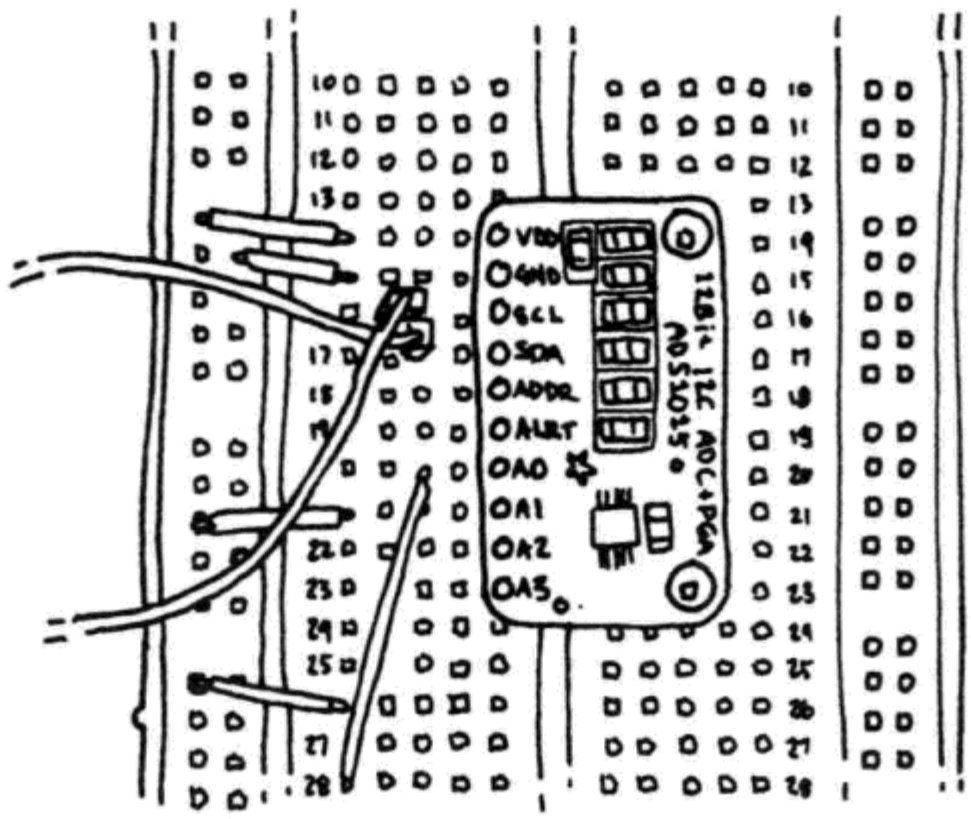


在本书中，你学到了如何用数字信号输入输出接口来连接按钮、开关、LED 和继电器。所有的这些器件都只有开或关两种状态，不会有中间状态。但是，现实中你可能需要去获取一些不是只有开、关两种状态的传感器信号，如温度、距离、亮度或旋钮的读数。

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 I^2C 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 I^2C 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 2kΩ 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。

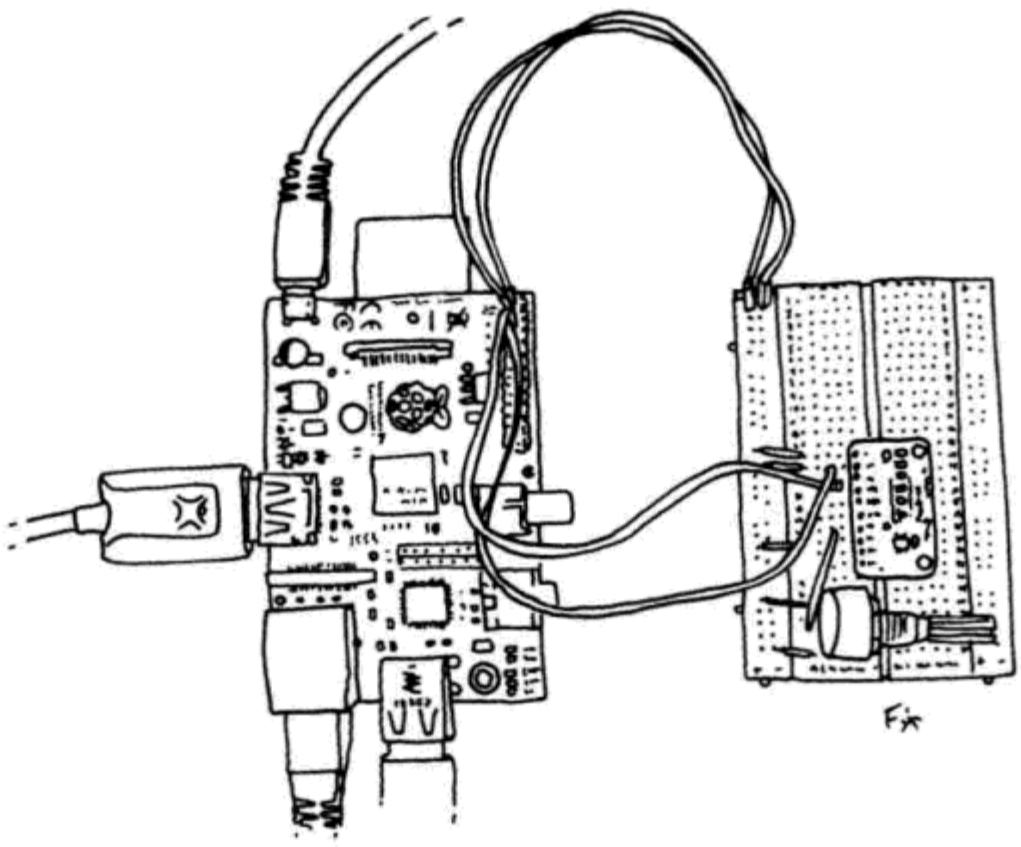
1. 把电位器插到面包板上。

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 I^2C 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 I^2C 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()❷

while True:
    result = adc.readADCSingleEnded(0)❸
    print result
    sleep(.5)
```

❶ 导入 Adafruit 的 ADS1x15 库。

❷ 创建一个 ADS1x15 的对象，名为 adc。

❸ 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置，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*。