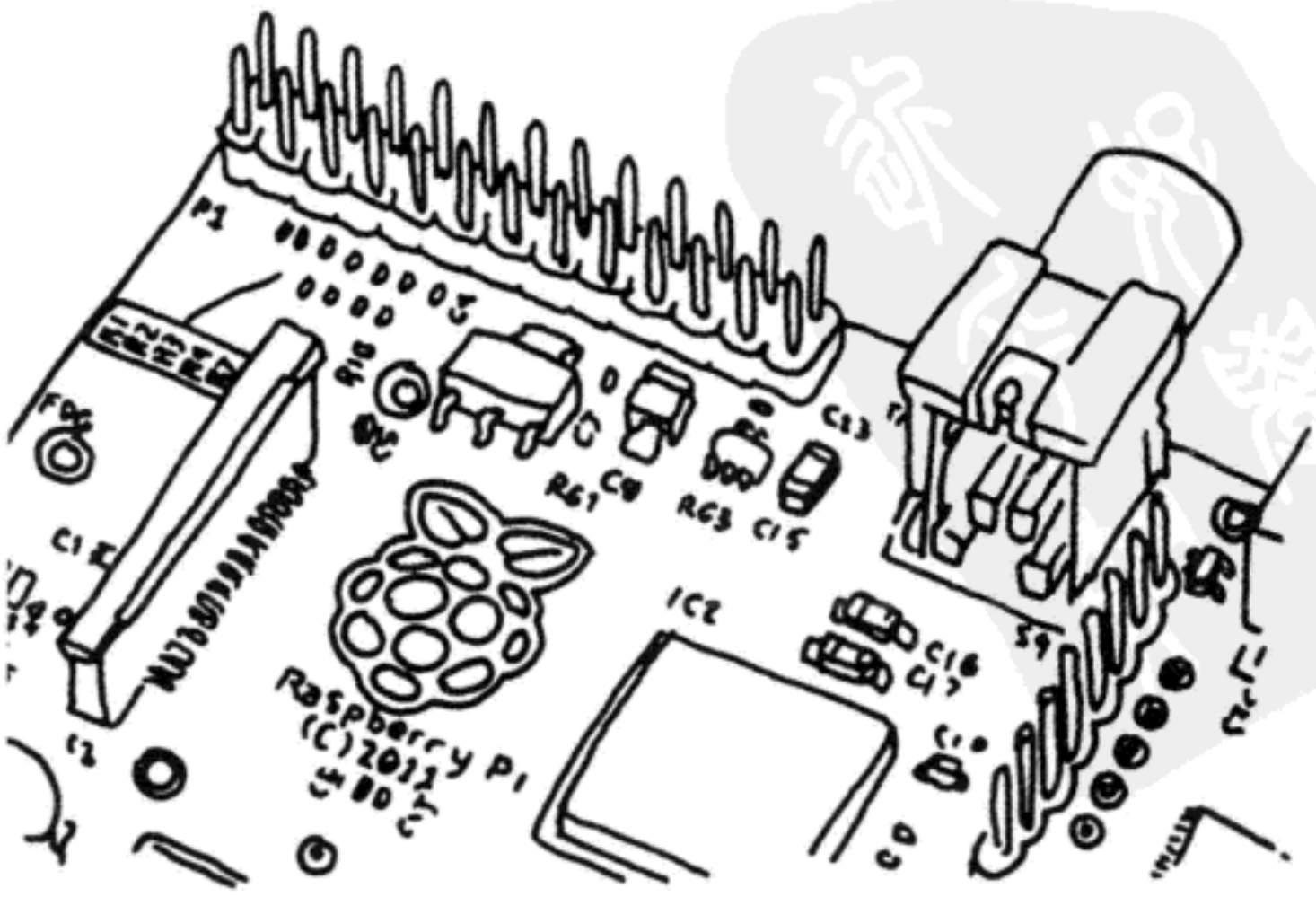
第6章

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）。

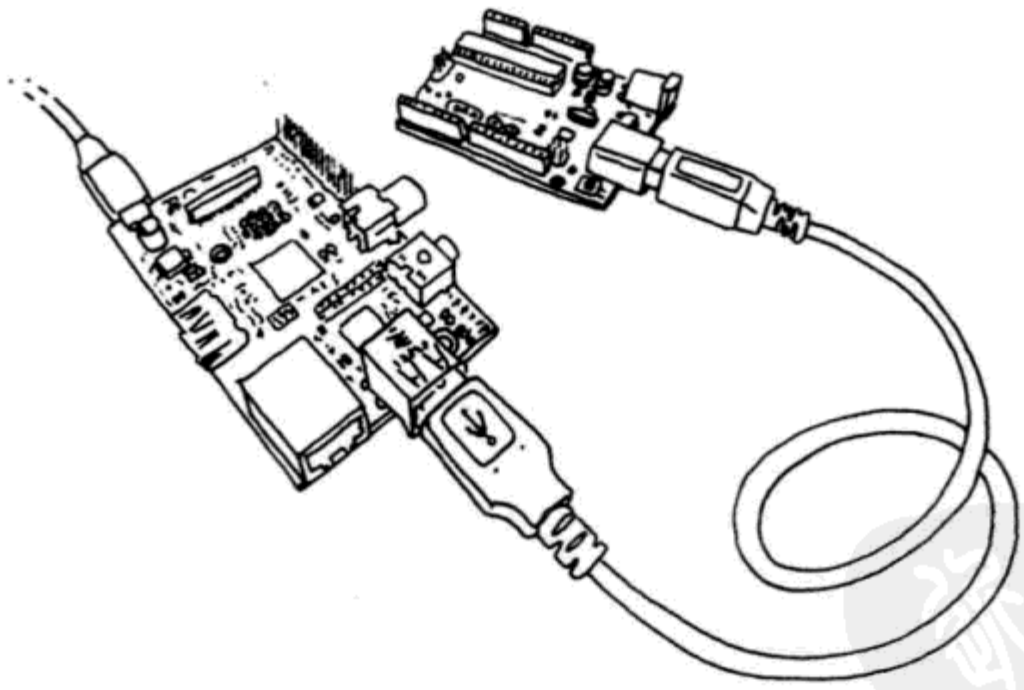


图6.1　Arduino与Raspberry Pi是“史上最好的朋友”

- 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 update❶
sudo apt-get install arduino❷
```

❶ 把软件仓库更新到最新版本。

❷ 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/ttyAMA0*（这实际上是板载的 USB Hub），当我插入 Adruino 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-incosolata
```

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/ttyACM0"
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。

❹ 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 Adruino 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 Ardunio 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”这样的入门知识后，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*）。