第 14 章

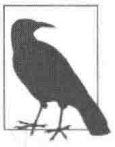
显示设备

14.0 引言

虽然树莓派可以选择显示器和 TV 作为显示设备，但是最好还是使用更加小巧和专用的显示设备。在本章中，我们将考察可供树莓派选用的各种显示设备。

其中，某些示例需要用到免焊面包板和 Female-to-Female（母头转母头）跳线（见 9.8 节）。

14.1 使用四位 LED 显示设备



请务必观看在 http://razzpisampler.oreilly.com 上与本节有关的视频。

面临问题

你想使用一个老式的七段发光二极管显示器来显示一个四位数字。

解决方案

你可以使用一个 I2C LED 模块，如图 14-1 中展示的那样，使用 Female-to-Female（母头转母头）跳线连接到树莓派上面。

为了进行本节中的试验，你需要：

- 4 个 Female-to-Female（母头转母头）跳线；
- Adafruit 背负 I2C 的 4×7 段 LED；

树莓派与该模块之间的连接如下所示。

- 该显示设备上面的 VCC（+）连接到树莓派 GPIO 接口上面的 5V 引脚。

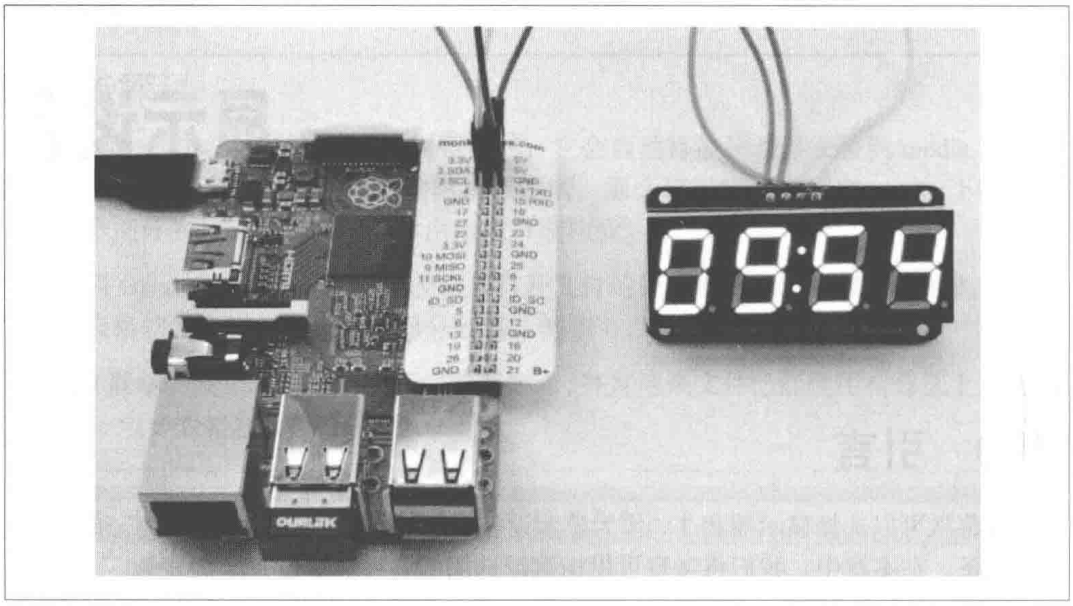


图 14-1　连接了七段 LED 显示设备的树莓派

- 该显示设备上面的 GND（-）连接到树莓派 GPIO 接口的 GND 引脚。
- 该显示设备上面的 SDA（D）连接到树莓派 GPIO 接口的 GPIO 2（SDA）上面。
- 该显示设备上面的 SCL（C）连接到树莓派 GPIO 接口的 GPIO 3（SCL）上面。

注意，Adafruit 还提供一个超大尺寸的 LED 显示屏。你可以使用上面的方式将其连接到树莓派上面，但是这个大尺寸的显示屏需要两个正极电源引脚：一个用于逻辑（V_IO），一个用于显示（5V）。为此，你只需另外使用一根 Female-to-Female（母头转母头）跳线将额外的引脚连接至 GPIO 接口的第二个 5V 引脚即可。这是因为由于该显示设备尺寸较大，所以需要更多的电流供发光二极管显示器使用。幸运的是树莓派可以提供足够的电流供它使用。

为了让本节中的示例正常运行，你需要首先按照 9.3 节中的介绍设置树莓派的 I2C。

Adafruit 为该显示设备提供了一个相应的 Python 库。但是，它不是作为一个严格意义上的库来安装的，所以要想使用该库的话，需要首先下载相应的文件夹结构。

```
$ git clone https://github.com/adafruit/Adafruit-Raspberry-Pi-Python-Code.git
```

通过下列命令切换到 Adafruit 代码所在目录。

```
$ cd Adafruit-Raspberry-Pi-Python-Code
$ cd Adafruit_LEDBackpack
```

在这个文件夹中，你可以找到一个显示时间的示例程序。为了运行这个程序，可以使

用下列代码。

```
$ sudo python ex_7segment_clock.py
```

进一步探讨

如果你利用 nano 打开这个示例文件 segment_clock.py 的话，将会看到如下所示的关键命令。

```
from Adafruit_7Segment import SevenSegment
```

上面命令的作用是将库代码导入到你的程序中，然后，你需要利用下面的一行代码来创建一个 SevenSegment 的实例。作为参数提供的地址实际上是 I2C 的地址（见 9.4 节）。

每个 I2C 从属设备都有一个地址数字。这个 LED 电路板的背面有 3 对焊盘，所以如果想改变地址的话，可以通过焊接方式将它们桥接起来。如果你想通过一个树莓派操作多个 I2C 设备的话，这是非常必要的。

```
segment = SevenSegment(address=0x70)
```

要想实际设置某个数字的内容，可以使用如下所示的命令。

```
segment.writeDigit(0, int(hour / 10))
```

第一个参数（0）是数字的位置，需要注意的是这些位置分别是 0、1、3 和 4。

位置 2 是预留给该显示设备中间的两个点的。

第二个参数是需要显示的数字本身。

参考资料

你可以从 http://bit.ly/HQBE6W 站点找到 Adafruit 库的详细介绍。

14.2 在 I2C LED 矩阵上面显示消息

面临问题

你希望控制彩色 LED 矩形显示设备上的像素。

解决方案

你可以使用一个 I2C LED 模块，如图 14-2 中展示的那样，使用 Female-to-Female（母头转母头）跳线连接到树莓派上面。

为了进行本节中的试验，你需要：

- 4 个 Female-to-Female（母头转母头）跳线；

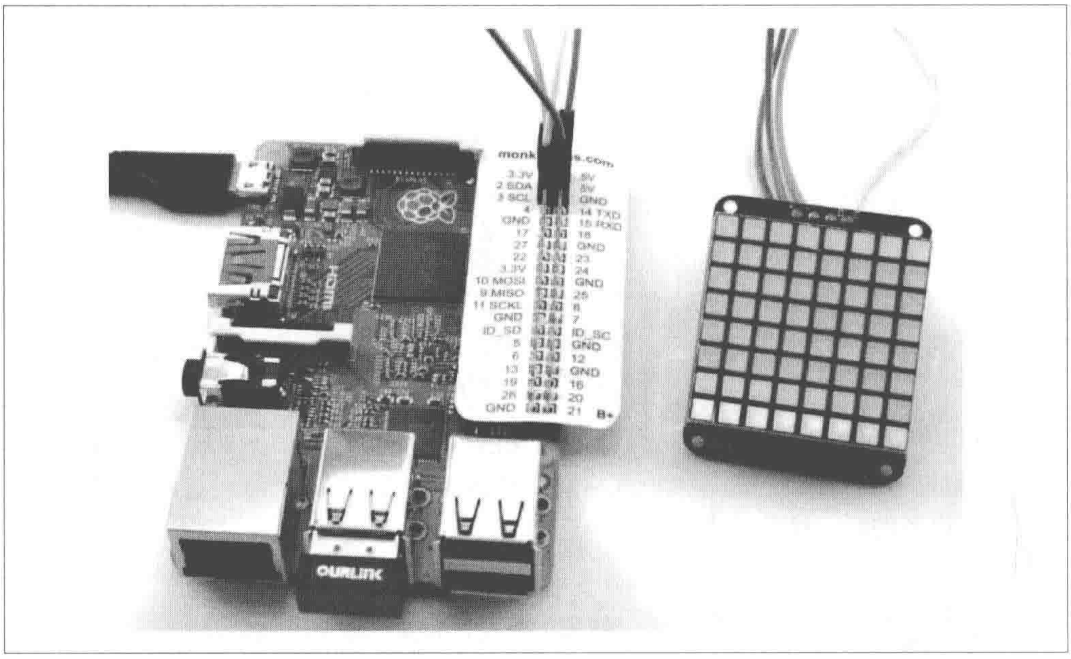


图 14-2　连接树莓派的 LED 矩阵显示设备

- Adafruit 背负 I2C 的双色 LED 方形像素矩阵显示设备。

树莓派与该模块之间的连接如下所示。

- 该显示设备上面的 VCC（+）连接到树莓派 GPIO 接口上面的 5V 引脚。
- 该显示设备上面的 GND（-）连接到树莓派 GPIO 接口的 GND 引脚。
- 该显示设备上面的 SDA（D）连接到树莓派 GPIO 接口的 GPIO 2（SDA）上面。
- 该显示设备上面的 SCL（C）连接到树莓派 GPIO 接口的 GPIO 3（SCL）上面。

为了让本节中的示例正常运行，你需要首先按照 9.3 节中的介绍设置树莓派的 I2C。

Adafruit 为该显示设备提供了一个相应的 Python 库。但是，它不是作为一个严格意义上的库来安装的，所以要想使用该库的话，需要首先下载相应的文件夹结构。

```
$ git clone https://github.com/adafruit/Adafruit-Raspberry-Pi-Python-Code.git
```

通过下列命令切换到 Adafruit 代码所在目录。

```
$ cd Adafruit-Raspberry-Pi-Python-Code
$ cd Adafruit_LEDBackpack
```

在这个文件夹下面，你可以找到一个使用滚动数字显示时间的测试程序。

要想运行该程序，可以使用如下所示的命令。

```
$ sudo python ex_8x8_color_pixels.py
```

进一步探讨

这个程序会按照顺序循环操作每个像素的所有颜色。下面给出的代码，已经删除了某些注释和不必要的导入代码。

```
import time
from Adafruit_8x8 import ColorEightByEight

grid = ColorEightByEight(address=0x70)

iter = 0

# Continually update the 8x8 display one pixel at a time
while(True):
  iter += 1

  for x in range(0, 8):
    for y in range(0, 8):
      grid.setPixel(x, y, iter % 4 )
      time.sleep(0.02)
```

在下面这行代码中，I2C 地址（见 9.4 节）是以参数的形式来提供的。

```
grid = ColorEightByEight(address=0x70)
```

每个 I2C 从属设备都有一个地址数字。这个 LED 电路板的背面有 3 对焊盘，所以如果想改变地址的话，可以通过焊接方式将它们桥接起来。如果你希望通过单个树莓派来操作多个具有相同基地址的 I2C 设备的话，这是非常有必要的。

每循环一次，变量 iter 的值就会加 1。命令 grid.setPixel 以 x 和 y 坐标作为其前两个参数。最后一个参数是给像素设置的颜色。该参数是一个介于 0 到 3 之间的数字（0 表示关闭，1 表示绿色，2 表示红色，而 3 则表示橘色）。

变量 iter 用于生成一个介于 0 到 3 之间的数字，这个数字是通过%运算符得到的，该运算符就是所谓的取模运算符，就本例来说就是除以 4 之后得到的余数。

参考资料

关于该产品的更多介绍，请参考 http://www.adafruit.com/products/902。

14.3 使用 Sense HAT LED 矩形显示器

面临问题

你想通过 Sense HAT 的显示器来展示消息和图像。

解决方案

首先，请按照 9.16 节介绍的方法来安装 Sense HAT 所需的软件，然后就可以使用相应

的库命令来显示文本了。

程序 sense_hat_clock.py 展示的是以滚动消息的形式重复显示日期与时间。正如本书中的所有示例程序一样，你可以从本书网站（http://www.raspberrypicookbook.com）的代码下载区下载它们。

```
from sense_hat import SenseHat
from datetime import datetime
import time

hat = SenseHat()
time_color = (0, 255, 0)
date_color = (255, 0, 0)

while True:
    now = datetime.now()
    date_message = '{:%d %B %Y}'.format(now)
    time_message = '{:%H:%M:%S}'.format(now)

    hat.show_message(date_message, text_colour=date_color)
    hat.show_message(time_message, text_colour=time_color)
```

进一步探讨

由于上面定义了两种颜色，所以消息的日期与时间部分将以不同的颜色来显示。然后，这些颜色将用作 show_message 的可选参数。

Show_message 的其他可选参数包括以下几个。

- scroll_speed 每个滚动步骤之间的时间延迟，并非滚动速度。所以这个值越大，滚动就会越慢。
- back_colour 设置背景色。需要注意的是这里的“back_colour”使用的是英式拼写，即带有字母“u”。该显示器的用途非常广泛，而非仅限于显示滚动的文本。就最简单的来说，你可以使用 set_pixel 设置特定的像素，利用 set_rotation 设置显示的方向，同时还可以通过 load_image 显示图像（虽然有点小）。下面的示例代码取自 sense_hat_taster.py，它很好地展示了这些函数的用法。就像本书中的所有示例程序一样，你也可以从本书网站（http://www.raspberrypicookbook.com）的代码下载区来下载该程序。

显示的图像的大小只能是 8×8 像素的，并且可以采用诸如.jpg 之类的常用图像格式，图像的位深度将会自动处理。

```
from sense_hat import SenseHat
import time

hat = SenseHat()

red = (255, 0, 0)
```

```
hat.load_image('small_image.png')
time.sleep(1)
hat.set_rotation(90)
time.sleep(1)
hat.set_rotation(180)
time.sleep(1)
hat.set_rotation(270)
time.sleep(1)

hat.clear()
hat.set_rotation(0)
for xy in range(0, 8):
    hat.set_pixel(xy, xy, red)
    hat.set_pixel(xy, 7-xy, red)
```

图 14-3 展示了正在显示一幅粗糙图像的 Sense HAT。

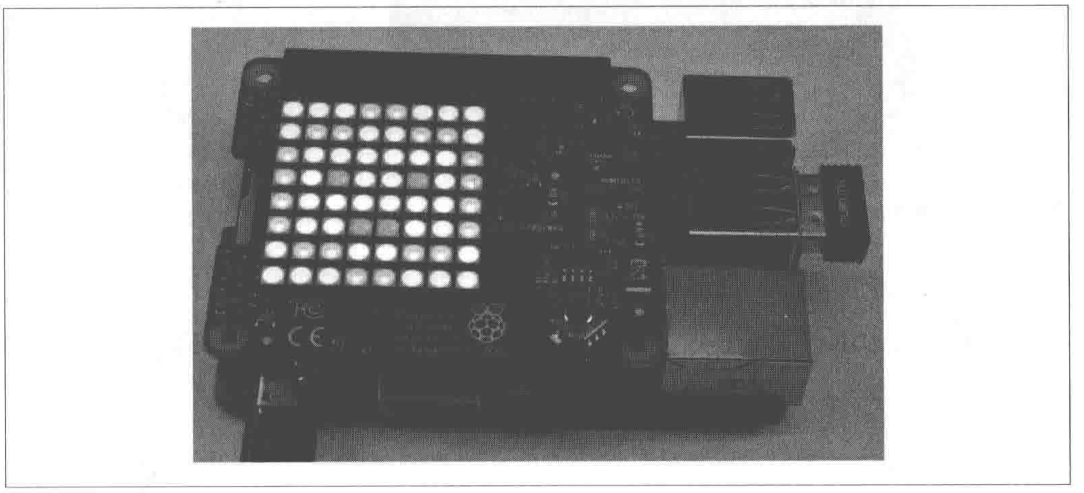


图 14-3　正在显示“图像”的 Sense HAT

参考资料

关于 Sense HAT 的完整文档，请参考 https://pythonhosted.org/sense-hat/api/。

关于格式化时间和日期的详细介绍，请参考 7.2 节。

其他用到 Sense HAT 的示例，请参考 9.16 节、13.10 节、13.13 节、13.14 节和 13.16 节。

14.4　在 Alphanumeric LCD HAT 上显示消息

面临问题

你想将几行文本优雅地显示到一个 LCD 显示设备上面。

解决方案

你可以像图 14-4 所示那样，在树莓派上面连接一个 Displayotron Pimoroni HAT。

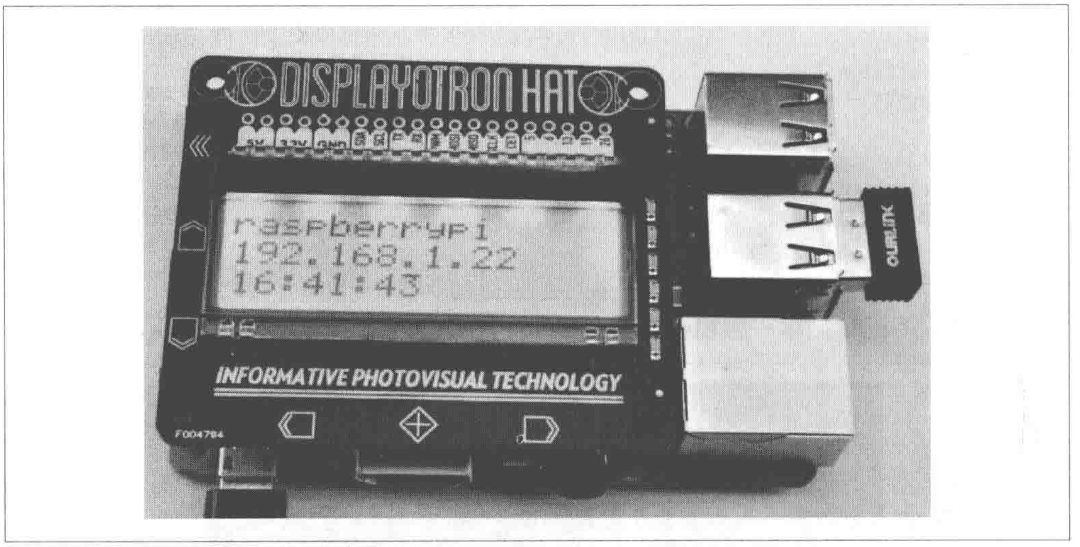


图 14-4　Displayotron LCD HAT

这个 HAT 要求同时启用 I2C 和 SPI，如果尚未启用的话，具体请参考 9.3 节和 9.5 节。

然后，从 GitHub 下载这个 HAT 对应的库代码，并通过下列代码进行安装。

```
$ git clone https://github.com/pimoroni/dot3k.git
$ cd dot3k/python/library
$ sudo python setup.py install
```

举例来说，下面的程序（displayotron_ip.py）可以查找树莓派的主机名和 IP 地址，并连同时间一起显示出来。如果一切正常的话，那么这个 LED 的背光就会变绿，但是如果网络连接有问题的话，背光就会变成红色。

就像本书中的所有示例程序一样，你也可以从本书网站（http://www.raspberrypiCookbook.com/）的代码下载区下载该程序。

```
import dothat.lcd as lcd
import dothat.backlight as backlight
import time
from datetime import datetime
import subprocess

while True:
    lcd.clear()
    backlight.rgb(0, 255, 0)
    try:
```

```
        hostname = subprocess.check_output(['hostname']).split()[0]
        ip = subprocess.check_output(['hostname', '-I']).split()[0]
        t = '{:%H:%M:%S}'.format(datetime.now())
        lcd.write(hostname)
        lcd.set_cursor_position(0, 1)
        lcd.write(ip)
        lcd.set_cursor_position(0, 2)
        lcd.write(t)
    except:
        backlight.rgb(255, 0, 0)
    time.sleep(1)
```

进一步探讨

这个测试程序将会导入所需的程序库，包括 subprocess 库（见 7.15 节），这些库被用来寻找树莓派的 IP 地址（见 2.2 节）及其主机名。

在这个库中，主要的方法包括如下几个。

- lcd.clear 清除显示的所有文本。
- lcd.set_cursor_position 设置新文本的写入位置，该方法以行和列作为参数指定具体位置。
- lcd.write 将作为参数提供给它的文本显示到当前光标位置。
- backlight.rgb 设置背光（0 到 255）的红、绿和蓝值。

参考资料

关于该 HAT 的更多信息，请参考 Pimoroni 相应的产品页面（https:// shop.pimoroni.com/products/display-o-tron-hat）。

如果你想直接给树莓派连接一个廉价的 LCD 模块，而非使用 HAT 的话，请参考 14.5 节。

14.5 在 Alphanumeric LCD 模块上显示消息

面临问题

你希望使用字母数字式的 LCD 显示设备来显示文字，而不是使用现成的显示器 HAT。

解决方案

你可以使用一个 HD44780-compatible LCD 模块，并将其连接到 GPIO 接口上面。

很明显，这比使用封装好的显示设备要复杂得多，但是，如果正在规划的项目非常复杂，并且使用一个元件作为其显示设备，同时还需要访问其他 GPIO 引脚的话，那么了

解直接使用 LCD 模块的方法将是非常有用的。

你可以找到许多廉价的 LCD 模块，例如图 14-5 中连接到树莓派上面的模块。这些模块都带有 16 个引脚，不过好在并非所有的引脚都需要连接到 GPIO 引脚上面。

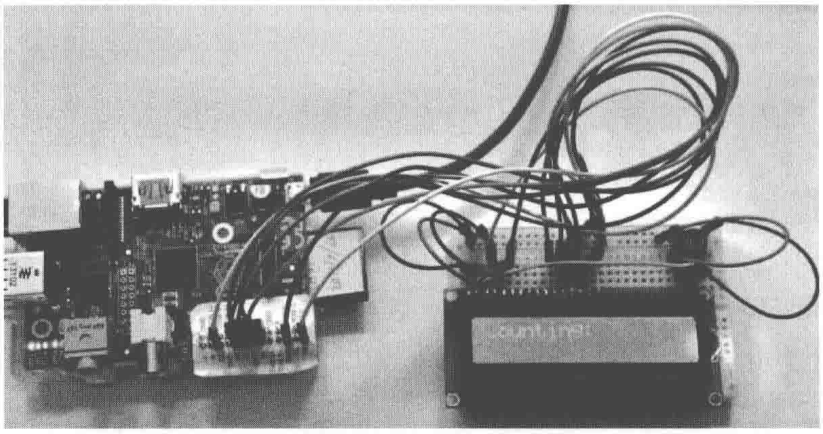


图 14-5　连接到树莓派上面的一个 16×2LCD 显示设备

这类模块的尺寸各异，并且每个尺寸都规定了可以显示的行数，以及每行可以显示的字符数。所以，举例来说，本节使用的模块的大小是 16×2，因为它可以显示两行文本，每行可以显示 16 个字符。

其他常见的尺寸有 8×1、16×1、16×2、20×2 和 20×4。

为了进行本节中的实验，你需要：

- 面包板和跳线；
- 16×2HD44780-compatible LCD 模块；
- 单行 16 接头引脚；
- 10kΩ 调谐电位器。

图 14-6 展示了连接该显示器的面包板布局。该 LCD 模块通常不提供插头引脚，所以你需要将这些引脚焊接到恰当的位置。

调谐电位器可以用来控制显示设备的对比度。需要注意的是当项目无法正常工作的时候，可以尝试将调谐电位器的旋钮在整个可移动范围内移动。这可能是由于对比度设置为关闭，所以导致显示的字符根本无法看到。

Adafruit 提供的树莓派示例代码可以从 GitHub (http://bit.ly/ 1gHetIt)上面下载，其中包括一个通过 HD44780 驱动程序来控制 LCD 显示设备的程序库。

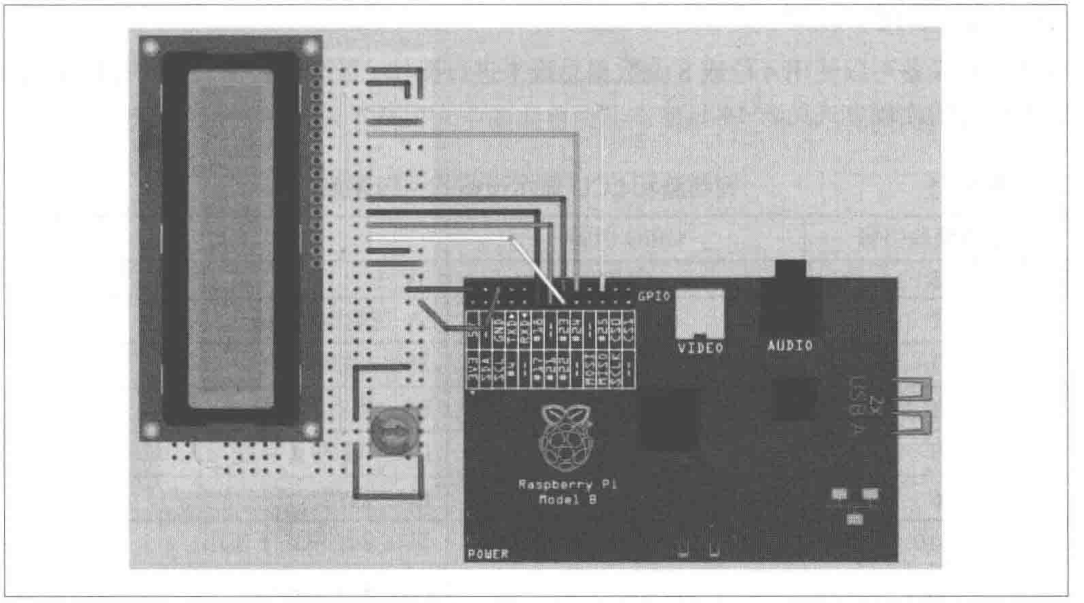


图 14-6　将 LCD 显示设备连接至树莓派

在安装该示例程序之前，需要按照 9.2 节中的介绍先把 RPi.GPIO 库安装好。

由于这个 Adafruit 库不是作为一个严格意义上的库来安装的，所以要想使用该库的话，需要首先下载相应的文件夹结构。

```
$ git clone https://github.com/adafruit/Adafruit-Raspberry-Pi-Python-Code.git
```

然后，将目录切换至存放 Adafruit 代码的目录下面，具体命令如下所示。

```
$ cd Adafruit-Raspberry-Pi-Python-Code
$ cd Adafruit_CharLCD
```

这里有一个测试程序，专门用来显示树莓派的时间和 IP 地址。

但是，如果你使用的是较新的第二版 B 型树莓派的话，那么你首先需要编辑文件 Adafruit_CharLCD.py。

```
$ nano Adafruit_CharLCD.py
```

找到下面这行。

```
def __init__ (self, pin_rs=25, pin_e=24, pins_db=[23, 17, 21, 22], GPIO = None):
```

将数字 21 替换为 27，使它看起来像下面这行一样，然后保存文件并退出。

```
def __init__ (self, pin_rs=25, pin_e=24, pins_db=[23, 17, 27, 22], GPIO = None):
```

现在，你就可以通过下面的命令来运行这个示例程序了。

```
$ sudo python Adafruit_CharLCD_IPclock_example.py
```

进一步探讨

这些显示设备可以使用 4 位或 8 位数据总线来进行操作，此外，还需要 3 个控制引脚。这些引脚的连接方式见表 14-1。

表 14-1　　树莓派和 LCD 显示设备的引脚连接

LCD 模块引脚	GPIO 引脚	备注
1	GND	0V
2	+5V	5V 逻辑电源电压
3	无连接	对比度控制电压
4	25	RS：寄存器选择
5	GND	RW：读/写（总是写入）
6	24	EN：启用
7-10	无连接	只在 8 位模式下使用
11	23	D4：4 号数据线
12	17	D5：5 号数据线
13	21	D6：6 号数据线
14	22	D7：7 号数据线
15	+5V	LED 背光
16	GND	LED 背光

Adafruit 库文件 Adafruit_CharLCD.py 的用途是设置数据引脚的值，并将其发送到显示模块。它提供了如下所示的函数，供你在编程的时候引用。

home()

移至左上角。

clear()

将显示的文本全部清除。

setCursor(column, row)

设置文本书写位置的光标位置。

cursor()

打开光标显示。

noCursor()

关闭光标显示（默认）。

message(text)

在当前光标位置写入文本。

下面这个迷你示例程序展示了利用这个库来显示一个简单消息是多么的容易。

```
from Adafruit_CharLCD import Adafruit_CharLCD
from time import sleep

lcd = Adafruit_CharLCD()
lcd.begin(16,2)

i = 0

while Tue:
    lcd.clear()
        lcd.message('Counting: ' + str(i))
        sleep(1)
        i = i + 1
```

参考资料

本节内容借鉴了 Adafruit 学习网站（http:// bit.ly/1kf4xWC）上面的一篇 Adafruit 教程。此外，Adafruit 还销售兼容本节内容的一款插件式电路板，该电路板上面附带了一个 LCD 显示设备。

为便利起见，可以使用 HAT 上面现成的字母数字式 LCD 显示设备，具体请参考 2.2 节。

14.6 使用 OLED 图形显示器

面临问题

你想给树莓派连接一个图形 OLED（有机 LED）显示器。

解决方案

你可以使用基于 SSD1306 驱动芯片的 OLED 显示器，这需要用到 I2C 接口（见图 14-7）。

为了进行本节中的试验，你需要：

- 4 个 Female-to-Female（母头转母头）跳线；
- I2C OLED 显示器，128 × 64 像素。

这些显示器中，有的只有 4 个引脚，其他的则有 8 个引脚。这些 8 个引脚的显示器（包括 Adafruit 的显示器）的引脚之所以多，是因为它们同时提供了 I2C 和 SPI 接口。而 4 引脚的显示器则只有 I2C 接口。

树莓派与该模块之间的连接如下所示。

- 该显示设备上面的 VCC 连接到树莓派 GPIO 接口上面的 5V 引脚。

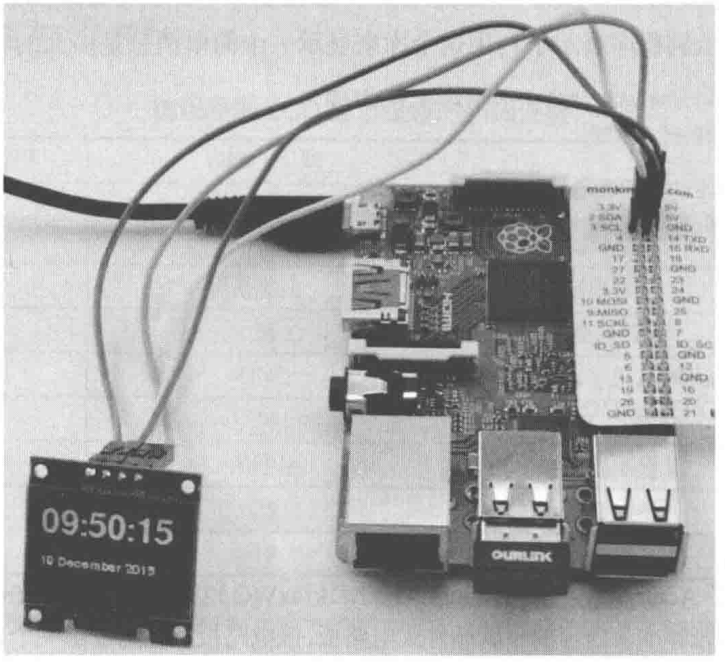


图 14-7　I2C OLED 显示器

- 该显示设备上面的 GND 连接到树莓派 GPIO 接口的 GND 引脚。
- 该显示设备上面的 SDA 连接到树莓派 GPIO 接口的 GPIO 2（SDA）上面。
- 该显示设备上面的 SCL 连接到树莓派 GPIO 接口的 GPIO 3（SCL）上面。

为了让本节中的示例正常运行，你需要首先按照 9.3 节中的介绍设置树莓派的 I2C。

Adafruit 为这些显示器提供了相应的库，具体安装方法如下所示。

```
$ git clone https://github.com/adafruit/Adafruit_Python_SSD1306.git
$ cd Adafruit_Python_SSD1306
$ sudo python setup.py install
```

这个库需要使用 Python 图像库（PIL），其安装命令如下所示。

```
$ sudo pip install pillow
```

示例程序 old_clock.py 可以在 OLED 显示器上面显示时间与时期。与本书中的其他示例程序一样，你也可以访问本书网站（http://www.raspberrypicookbook.com），并从代码下载区下载该程序。

```
from oled.device import ssd1306
from oled.render import canvas
```

```
from PIL import ImageFont
import time
from datetime import datetime

# Set up display
device = ssd1306(port=1, address=0x3C)
font_file = '/usr/share/fonts/truetype/freefont/FreeSansBold.ttf'
small_font = ImageFont.truetype('FreeSans.ttf', 12, filename=font_file)
large_font = ImageFont.truetype('FreeSans.ttf', 33, filename=font_file)

# Display a message on 3 lines, first line big font
def display_message(top_line, line_2):
    global device
    with canvas(device) as draw:
        draw.text((0, 0),  top_line, font=large_font, fill=255)
        draw.text((0, 50),  line_2, font=small_font, fill=255)

while True:
    now = datetime.now()
    date_message = '{:%d %B %Y}'.format(now)
    time_message = '{:%H:%M:%S}'.format(now)
    display_message(time_message, date_message)
    time.sleep(0.1)
```

每一个 I2C 从属设备都有一个地址，该地址可以通过下面代码进行设置。

```
device = ssd1306(port=1, address=0x3C)
```

对于许多廉价的 I2C 模块来说，它们的地址通常都是固定的 3C（十六进制数），但是有时也会有所不同，所以你应该仔细查看设备附带的文档，或者利用 I2C 工具（见 9.4 节）列出连接到总线上面的所有 I2C 设备，这样就能够从显示设备上面看到它们的地址了。

进一步探讨

小型的有机 LED（OLED）显示设备不但便宜、省电，并且虽然外形小巧，但是却具有非常高的分辨率。在许多消费产品中，它们正有取代 LCD 显示器的趋势。

参考资料

本节内容适用于 4 引脚的 I2C 接口。如果你想使用 SPI 接口的话，请参考 Adafruit 网站上的一篇相关教程，地址为 https://learn.adafruit.com/ ssd1306-oled-displays-with-raspberry-pi-and-beaglebone-black。

14.7 使用可寻址的 RGB LED 灯条

面临问题

你想给树莓派连接一个 RGB LED 灯条（Neopixels）。

解决方案

你可以在树莓派上面使用基于 WS2812 RGB LED 芯片的 LED 灯条。

这些 LED 灯条的用法其实非常简单，你只要直接将其连接到树莓派，并通过树莓派来给这些 LED 供电即可。当然，在正常情况下这种方法能够很好地工作，但是阅读本节的“进一步探讨”部分后你就会发现，这种方法有许多潜在的缺陷，不过别担心，我们同时提供了相应的解决方案，能够保证你无忧地使用 LED 灯条。

不要使用树莓派的 3.3V 引脚给 LED 供电

虽然利用 GPIO 接口的 3.3V 电源引脚为 LED 供电看上去非常有诱惑力，但是请不要这样做——它只能提供弱电流（见 9.2 节）。如果使用这个引脚供电的话，很容易损坏树莓派。

在图 14-8 中使用的 LED 灯条是从灯条盘卷中剪下来的。就本例来说，该灯条只有 10 个 LED。

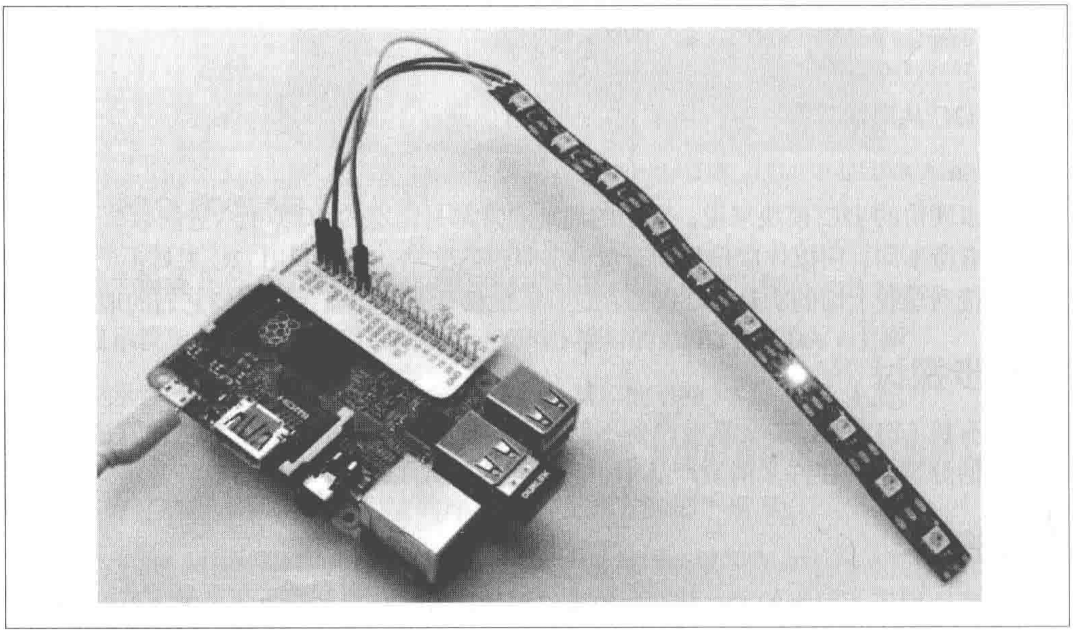


图 14-8　具有 10 个 LED 的灯条

因为每个 LED 都会消耗 60mA 的电流，所以，如果没有为 LED 灯条提供单独的电源的话，10 就是 LED 数目的一个比较合理的上限（请参考“进一步探讨”部分）。

为了将该灯条连接到树莓派上面，可以将带有塞孔的跳线的一端剪下来，并将导线端焊接到该 LED 灯条的 3 个连接上面：GND、DI（数据输入）和 5V 连接。然后，将这些跳线分别连接到 GPIO 接口的 GND、GPIO18 和 5V 引脚上面即可。

需要注意的是该 LED 灯条上面印有一行箭头（见图 14-9）。当你给这个 LED 灯条焊接引线的时候，请确保从左侧箭头的切头处开始。

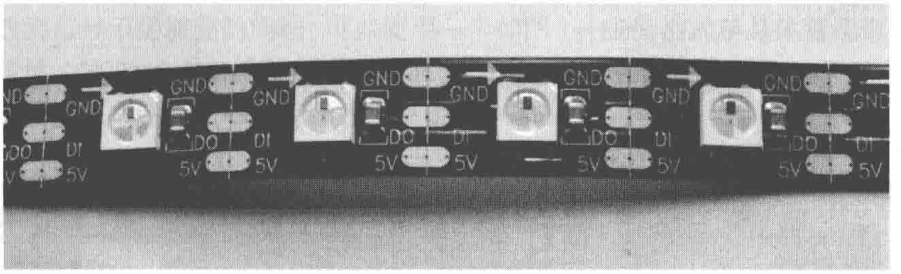


图 14-9　LED 灯条的特写

截止到写作本书时，寻找适用于所有树莓派型号的 WS2812 代码库还不是一件容易的事情。最初的代码库是由 Jeremy Garff 开发的（https://github.com/jgarf/rpi_ws281x），后来 Richard Hurst 提供了支持树莓派 2 的代码库（https://github.com/richardghirst/rpi_ws281x）。Adafrui 也参与了该库的开发。但是，Raspbian 切换到 Linux 的 4.1.6 内核后，有些事情就停了下来。截止写作本书时，Pimoroni Unicorn HAT（它使用了 Richard Hurst 的代码库的一个改进版）是最为可靠的一个版本。由于跟商业产品有关，所以每当 Raspbian 和树莓派电路板发生改进时，这个版本最有可能得到及时的维护。为了安装这个库，可以使用如下所示的代码。

```
$ git clone https://github.com/pimoroni/unicorn-hat.git
$ cd unicorn-hat/python/rpi-ws281x
$ make
$ sudo python setup.py install
```

下面的示例程序（led_strip.py）将会使 LED 沿着灯条的方向依次变红。与本书中的其他示例程序一样，你也可以访问本书网站（http://www.raspberrypicookbook.com），并从代码下载区下载该程序。

```
import time
from neopixel import *

# LED strip configuration:
LED_COUNT      = 10      # Number of LED pixels.
LED_PIN        = 18      # GPIO pin connected to the pixels (must support PWM!).
LED_FREQ_HZ    = 800000  # LED signal frequency in hertz (usually 800khz)
LED_DMA        = 5       # DMA channel to use for generating signal (try 5)
LED_BRIGHTNESS = 255     # Set to 0 for darkest and 255 for brightest
LED_INVERT     = False   # True to invert the signal

RED = Color(255, 0, 0)
NO_COLOR = Color(0, 0, 0)
```

```
strip = Adafruit_NeoPixel(LED_COUNT, LED_PIN, LED_FREQ_HZ, LED_DMA,
LED_INVERT, LED_BRIGHTNESS)
strip.begin()

def clear():
  for i in range(0, LED_COUNT):
    strip.setPixelColor(i, NO_COLOR)
  strip.show()

i = 0
while True:
  clear()
  strip.setPixelColor(i, RED)
  strip.show()
  time.sleep(1)
  i += 1
  if i >= LED_COUNT:
    i = 0
```

在 LED strip coniguration 部分中的各个参数，可以根据不同的 LED 配置进行相应的修改。如果你的灯条具有不同数量的 LED，那么可以修改 LED_COUNT。而其余的常数则无需改变。

灯条上面的每个 LED，都可以通过 setPixelColor 方法来为其设置单独的颜色。这个方法的第一个参数，是需要设置颜色的 LED 的下标位置（从 0 开始）。第二个参数是要设置的颜色。实际上，直到这个 show 方法被调用之后，LED 的颜色才会发生改变。

进一步探讨

灯条上面的每个 LED 的最大电流为 60mA 左右，但是，只有当所有 3 个颜色通道（红、绿和蓝）都达到最大亮度（255）时，才会消耗这么多的电流。因此，如果你计划使用大量 LED 的话，那么需要使用一个独立的 5V 电源，才能给灯条上面的 LED 提供足够的电流。图 14-10 展示了连接独立电源的方法。如果使用一个母头 DC 插孔到螺丝端子的适配器的话，可以使得外部电源到面包板的连接工作更加轻松。

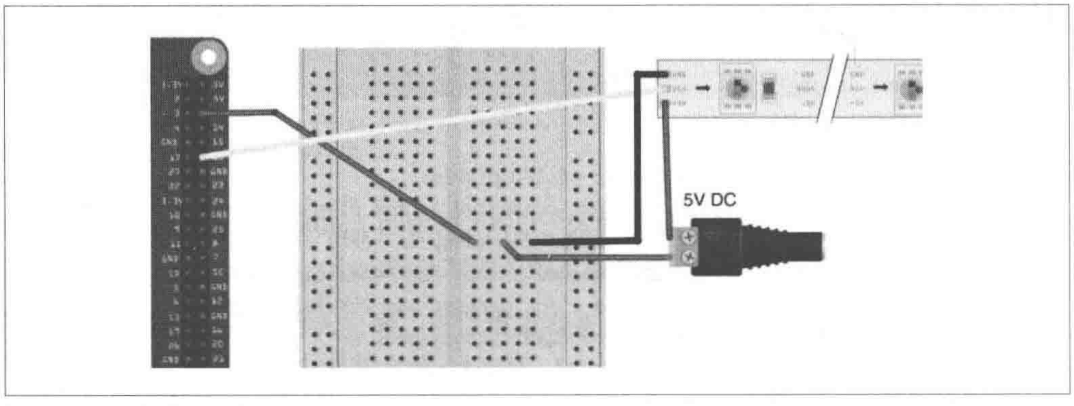


图 14-10　利用外部电源为 LED 灯条供电

根据 WS2812 的参数手册，用于 LED 的 5V 电源的输入信号的逻辑高电平至少要有 4V。因此，当使用树莓派的 3.3v GPIO 输出给它供电的时候，很难保证 LED 能够可靠工作。话虽如此，我却从未遇到过任何麻烦。

如果你遇到信号不足够强的问题，可以使用一个如图 14-11 所示的单晶体管逻辑电平转换器。这里，2N7000 是一个比较合适的晶体管（注意，在这种情况下 2N3904 无法正常工作，因为它不能迅速地响应反转脉冲）。

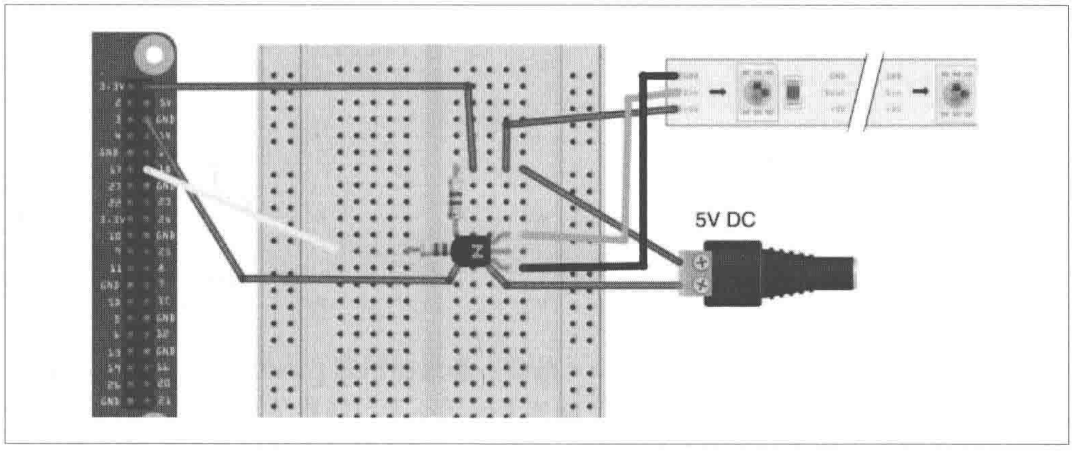


图 14-11　使用逻辑电平变换器

虽然这个晶体管能够将 3.3V 的信号转换为 5V，但是它有一个副作用，就是会反转信号，即如果 GPIO 引脚的输出是 HIGH 的话，进入 LED 灯条的信号将会变为 LOW，反之亦然。这就意味着你需要改变下面这行代码，来修改程序的逻辑。

```
LED_INVERT    = False
```

改为：

```
LED_INVERT    = True
```

参考资料

除了可以使用晶体管将信号电平从 3.3V 提高到 5V 之外，你还可以使用电平转换模块（见 9.13 节）。

Unicorn HAT（本节使用的就是它的程序库）具有 64 个 WS2812 LED，它们以 8×8 的网格形式排列。关于这个显示设备的更多介绍，请访问 https://shop.pimoroni.com/products/unicorn-hat。