ГЛАВА 14

Дисплеи

Введение

Несмотря на возможность вывода информации на телевизор или компьютерный монитор, Raspberry Pi поддерживает еще один тип устройств вывода несравнимо более компактного размера: дисплеи. В этой главе рассмотрены самые разные устройства данного типа и показано, как ими управлять из Raspberry Pi.

В некоторых примерах вам придется использовать макетную плату беспаечного монтажа и проволочные перемычки с гнездово-штыревыми разъемами на концах.

14.1. Четырехразрядный светодиодный индикатор

Задача

Вывести четырехзначное число на винтажном четырехразрядном семисегментном индикаторе, подключенном к Raspberry Pi.

Решение

Используйте светодиодный модуль, показанный на рис. 14.1, передача данных с которого в Raspberry Pi осуществляется через последовательный интерфейс I2C. Для подключения этого модуля расширения вам понадобятся проволочные перемычки со штыревыми разъемами на концах.

Для выполнения текущего рецепта вам понадобятся следующие электронные компоненты:

- четыре проволочные перемычки с только гнездовыми разъемами на концах (см. приложение А);
- четырехразрядный семисегментный светодиодный индикатор с интерфейсом I2C, производимый компанией Adafruit (см. приложение А).

Соедините следующие выводы перемычками, чтобы физически подключить устройства одно к другому:

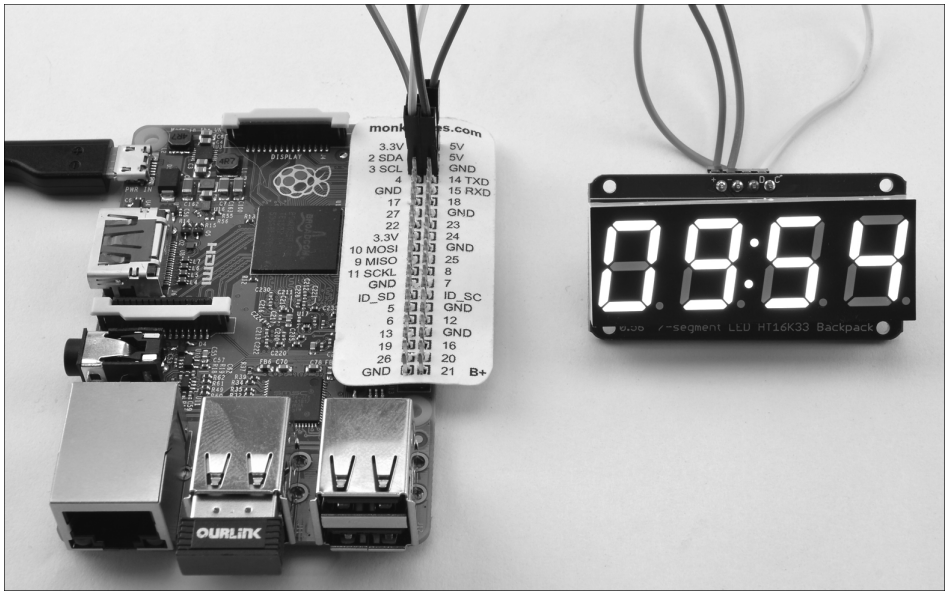

Рис. 14.1. Семисегментный светодиодный индикатор, подключенный к Raspberry Pi

- вывод VCC(+) индикатора соедините с выводом 5V интерфейса GPIO платы Raspberry Pi;
- вывод GND(–) индикатора соедините с выводом GND интерфейса GPIO платы Raspberry Pi;
- вывод SDA(D) индикатора соедините с выводом GPIO 2 (SDA) платы Raspberry Pi;
- вывод SCL(C) индикатора соедините с выводом GPIO 3 (SCL) платы Raspberry Pi.

Просматривая каталог компании Adafruit, легко обнаружить индикатор намного большего размера. Крупный индикатор потребляет намного большую мощность, чем маленький. Поэтому он, сохраняя общую схему подключения к Raspberry Pi, оснащается двумя выводами подключения питания: для микросхемной логики (V_IO) и собственно дисплея (5V). Подключение второй линии питания выполняется с помощью еще одной перемычки, соединяющей второй вывод 5V порта GPIO с выводом V_IO (или 5V) дисплея. К счастью, выходной мощности на выходе интерфейса GPIO достаточно для "запитывания" даже крупного светодиодного индикатора.

Для реализации текущего рецепта требуется включить в Raspberry Pi поддержку последовательного интерфейса I2C. О том, как это сделать, рассказывалось в рецепте 9.3.

Компания Adafruit снабжает свои индикаторы специальной программной библиотекой, обеспечивающей доступ ко всем функциональным средствам индикатора. Она устанавливается не так, как большинство общедоступных библиотек Python. Отличие состоит в необходимости перенастройки ее файловой структуры.

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

Измените название конечного каталога, содержащего код библиотеки.

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

В этом каталоге находится тестовая программа, выводящая на экран индикатора системное время. Для ее запуска выполните следующую команду:

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

Обсуждение

Открыв файл тестовой программы `ex_7segment_clock.py` в редакторе `nano`, вы увидите, что основной ее код подключается следующей строкой:

```
from Adafruit_7Segment import SevenSegment
```

Именно этот код отвечает за импорт библиотеки в конечную программу. Для управления индикатором в самой программе создается экземпляр класса `Seven Segment`. В качестве аргумента ему передается адрес I2C устройства, подключенного к Raspberry Pi (см. рецепт 9.4).

Каждому подчиненному устройству I2C назначается свой уникальный адрес. На задней тыльной стороне платы светодиодного индикатора находятся три пары контактных площадок. В случае необходимости на них можно нанести перемычки из припоя, изменив таким образом адресацию I2C-устройства. К счастью, такую операцию потребуется выполнить только при подключении к Raspberry Pi сразу двух одинаковых светодиодных индикаторов.

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

Чтобы передать на определенный разряд индикатора информацию, используйте следующий код:

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

Первый аргумент (с нашем случае — `0`) представляет порядковый номер или положение разряда. Обратите внимание на то, что разряды в индикаторе нумеруются следующим образом: 0, 1, 3, 4. Позиция с номером 2 зарезервирована для необязательного символа двоеточия, выводимого посредине индикатора.

Цифра, отображаемая в указанной первым аргументом позиции, задается вторым аргументом функции.

Дополнительные сведения

Детально библиотека управления индикаторами Adafruit описана на сайте производителя.

14.2. Матричный светодиодный дисплей I2C

Задача

Научиться управлять пикселями многоцветного матричного дисплея.

Решение

Давайте рассмотрим светодиодный модуль, содержащий матрицу многоцветных элементов, который подключается к Raspberry Pi с помощью интерфейса I2C. Для его подключения (рис. 14.2) потребуются проволочные перемычки с только гнездовыми разъемами на концах.

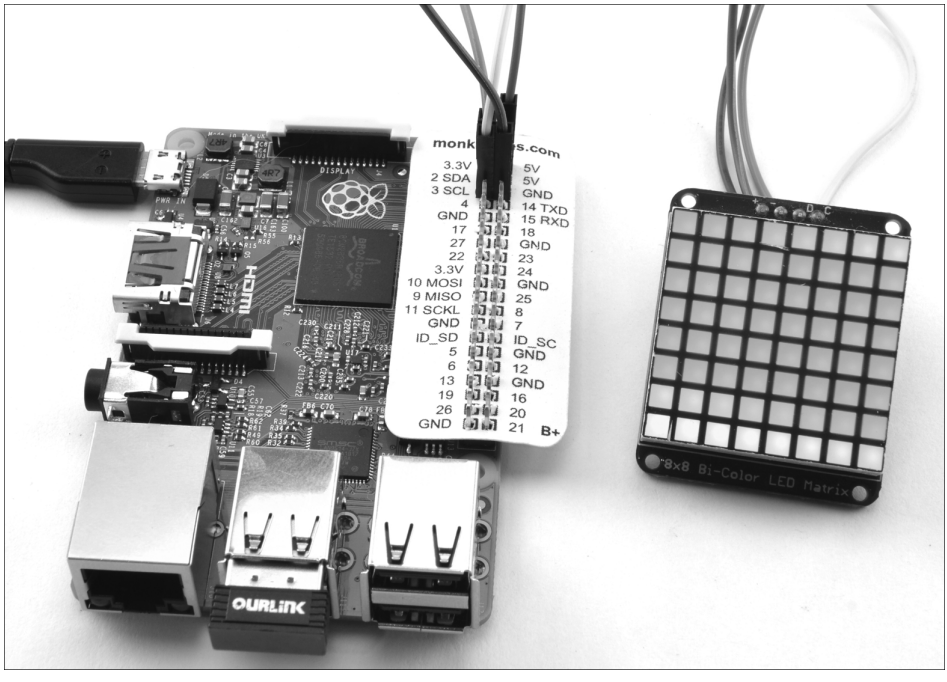

Рис. 14.2. Матричный светодиодный дисплей, подключенный к Raspberry Pi

Для выполнения текущего задания необходимо следующее оборудование:

- четыре проволочные перемычки с гнездовыми разъемами на концах (см. приложение А);
- двухцветный матричный светодиодный дисплей производства компании Adafruit с квадратными пикселями, оснащенный интерфейсом I2C (см. приложение А).

Соедините следующие выводы перемычками, чтобы физически подключить устройства одно к другому:

- вывод VCC(+) дисплея соедините с выводом 5V интерфейса GPIO платы Raspberry Pi;
- вывод GND(–) дисплея соедините с выводом GND интерфейса GPIO платы Raspberry Pi;
- вывод SDA(D) дисплея соедините с выводом GPIO 2 (SDA) платы Raspberry Pi;
- вывод SCL(C) дисплея соедините с выводом GPIO 3 (SCL) платы Raspberry Pi.

Для реализации рецепта требуется включить в Raspberry Pi поддержку последовательного интерфейса I2C. О том, как это сделать, рассказывалось в рецепте 9.3.

Компания Adafruit снабжает свои дисплеи специальной программной библиотекой, обеспечивающей доступ ко всем их функциональным средствам. Она устанавливается не так, как большинство общедоступных библиотек Python. Отличие состоит в необходимости перенастройки ее файловой структуры.

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

Измените название конечного каталога, содержащего код библиотеки.

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

В этом каталоге находится тестовая программа, выводящая на экран дисплея системное время, представленное бегущей строкой. Для ее запуска выполните следующую программу:

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

Обсуждение

На каждом этапе программа перебирает цвета для каждого пикселя. В приведенном ниже коде показаны только строки, определяющие поведение программы, — из него исключены команды импорта всех ненужных для данного примера библиотек.

```
import time
from Adafruit_8x8 import ColorEightByEight

grid = ColorEightByEight(address=0x70)

iter = 0

# Последовательное обновление всех пикселей дисплея 8×8
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)
```

В следующей строке функции в качестве аргумента передается адрес I2C дисплея (см. рецепт 9.4):

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

Каждому подчиненному устройству I2C назначается свой уникальный адрес. На тыльной стороне платы светодиодного дисплея находятся три пары контактных площадок. В случае необходимости на них можно нанести перемычки из припоя, изменив таким образом адресацию I2C-устройства. К счастью, такую операцию потребуется выполнить только при подключении к Raspberry Pi сразу двух одинаковых светодиодных дисплеев.

На каждой итерации цикла значение переменной `iter` увеличивается на единицу, принимая участие в формировании цветового оттенка для всех последовательно перебираемых пикселей матрицы. Функции `grid.setPixel()` в качестве первых двух аргументов передаются координаты пикселя, обрабатываемого в текущей итерации. Последний аргумент определяет цвет выбранного пикселя — он принимает значение в диапазоне от 0 до 3 (`0` — выключен, `1` — зеленый, `2` — красный, `3` — оранжевый).

Как уже указывалось, переменная `iter` задействуется в формуле расчета цвета пикселя, в которой символ `%` представляет операцию вычисления остатка от модуля числа (в данном случае остатка от деления значения переменной `iter` на число `4`).

Дополнительные сведения

Детально это устройство вывода описано на сайте производителя.

14.3. Матричный светодиодный дисплей Sense HAT

Задача

Вывести сообщения и графические изображения на дисплей модуля расширения Sense HAT.

Решение

Выполните инструкции рецепта 9.16 для установки программного обеспечения, необходимого для управления модулем Sense HAT.

Программа `sense_hat_clock.py` позволяет увидеть дисплей модуля в действии — на него выводится бегущая строка, в которой отображаются системные дата и время. Как и любые другие примеры программ, она доступна для загрузки с сайта книги (см. раздел Code).

```
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)
```

Обсуждение

Программа работает так, что дата и время в бегущей строке отображаются разными цветами. Цвет текста указывается в качестве необязательного аргумента функции `show_message()`.

У этой функции есть другие необязательные параметры.

- `scroll_speed`. Указывает задержку перед следующим шагом прокрутки бегущего сообщения. Таким образом, чем больше значение, тем медленнее прокрутка.
- `back_colour`. Определяет фоновый цвет бегущей строки.

Дисплей модуля Sense HAT позволяет не только выводить простой текст. В частности, в программе можно отдельно управлять каждым пикселем дисплея (функция `set_pixel()`), поворачивать изображение на дисплее (функция `set_rotation()`), а также выводить на него готовый графический файл (функция `load_image()`). Для иллюстрации описанных возможностей автором создана программа `sense_hat_taster.py`, приведенная ниже и доступная для загрузки на сайте книги (см. раздел Code).

На дисплей модуля выводятся только изображения, имеющие размер 8×8 пикселей. Тем не менее в программу можно загружать файлы, сохраненные в популярных форматах JPG и PNG и обладающие другой битовой глубиной — они приводятся к необходимому формату автоматически.

```
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)
```

Примитивное изображение, выведенное на дисплей модуля расширения Sense HAT, показано на рис. 14.3.

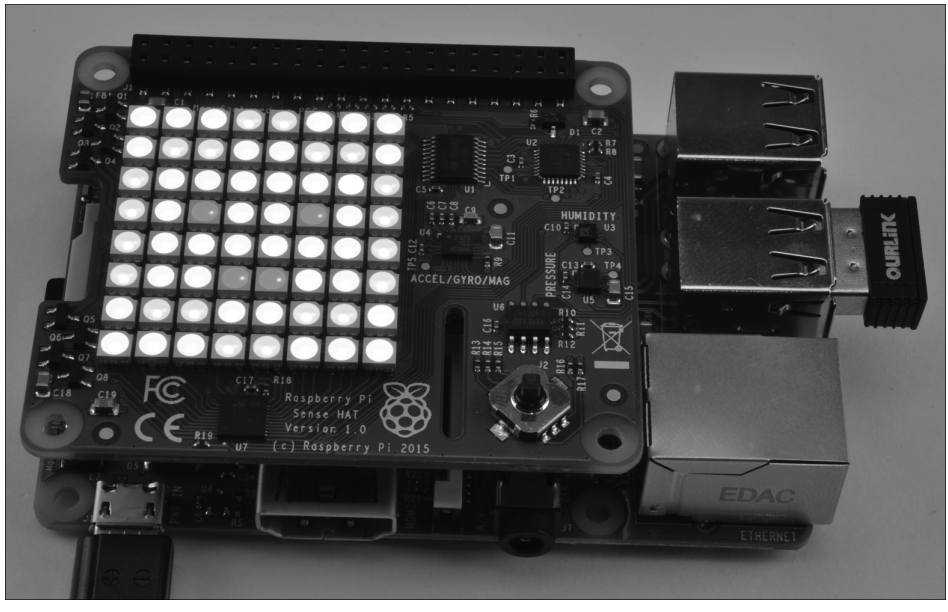

Рис. 14.3. Вывод "изображения" на дисплей модуля Sense HAT

Дополнительные сведения

Полная документация к модулю Sense HAT приведена на следующем сайте:

`https://pythonhosted.org/sense-hat/api/`

О форматировании даты и времени в Python рассказывалось в рецепте 7.2.

Остальные возможности модуля Sense HAT были рассмотрены в рецептах 9.16, 13.10, 13.13, 13.14 и 13.16.

14.4. Алфавитно-цифровой ЖК-дисплей модуля расширения

Задача

Вывести на жидкокристаллический дисплей текст, введенный в несколько строк.

Решение

Используйте устройство Displayotron LCD HAT компании Pimoroni, монтируемое поверх платы Raspberry Pi, как показано на рис. 14.4.

Рис. 14.4. Модуль Displayotron LCD HAT

Для обеспечения нормальной работы модуля в настройках Raspberry Pi нужно включить поддержку сразу двух последовательных интерфейсов: I2C (см. рецепт 9.3) и SPI (см. рецепт 9.5).

Загрузите код, отвечающий за управление оборудованием модуля, из GitHub и установите его в системе, выполнив следующие команды.

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

В качестве примера рассмотрим приведенную ниже программу (`displayotron_ip.py`), которая находит имя хоста и IP-адрес платы Raspberry Pi и выводит их, наряду с текущим временем, на дисплее модуля. При успешном выполнении всех

поставленных задач на дисплее сохраняется зеленая подсветка, в противном случае он подсвечивается отталкивающим красным оттенком.

Как и любые другие примеры, программа доступна для загрузки на сайте книги (см. раздел Code).

```
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)
```

Обсуждение

В начале тестовой программы импортируются библиотеки, обеспечивающие выполнение всех основных действий, включая такие задачи, как поиск IP-адреса и имени хоста Raspberry Pi.

В программе используются следующие методы, предоставленные библиотеками.

- `lcd.clear()`. Очищает дисплей от текста.
- `lcd.set_cursor_position()`. Указывает положение выводимого на дисплей текста, представляя его номером строки и столбца (два разных аргумента).
- `lcd.write()`. Определяет текст, передаваемый как аргумент и выводимый в месте текущего расположения курсора.
- `backlight.rgb()`. Устанавливает красную, зеленую и синюю компоненты подсветки дисплея.

Дополнительные сведения

Детально о модуле Displayotron LCD HAT можно узнать на сайте его производителя — компании Pimoroni.

Детально о подключении простого ЖК-дисплея непосредственно к Raspberry Pi рассказывается в следующем рецепте.

14.5. Отдельный алфавитно-цифровой ЖК-дисплей

Задача

Вывести текст на ЖК-дисплей, представленный отдельным устройством, а не включенный в состав модуля расширения.

Решение

Подключите к выводам GPIO платы Raspberry Pi жидкокристаллический модуль HD44780.

Использовать его намного сложнее, чем дисплей, включенный в один из многофункциональных модулей расширения. Но если вы хотите научиться эффективно использовать выводы Raspberry Pi в сложных проектах, то нужно уметь подключать ЖК-дисплей к интерфейсу GPIO напрямую.

Вы легко найдете в продаже самые разные ЖК-дисплеи, включая изображенный на рис. 14.5. Показанная на рисунке модель имеет 16 выводов, но далеко не все они подключаются к колодке GPIO платы Raspberry Pi.

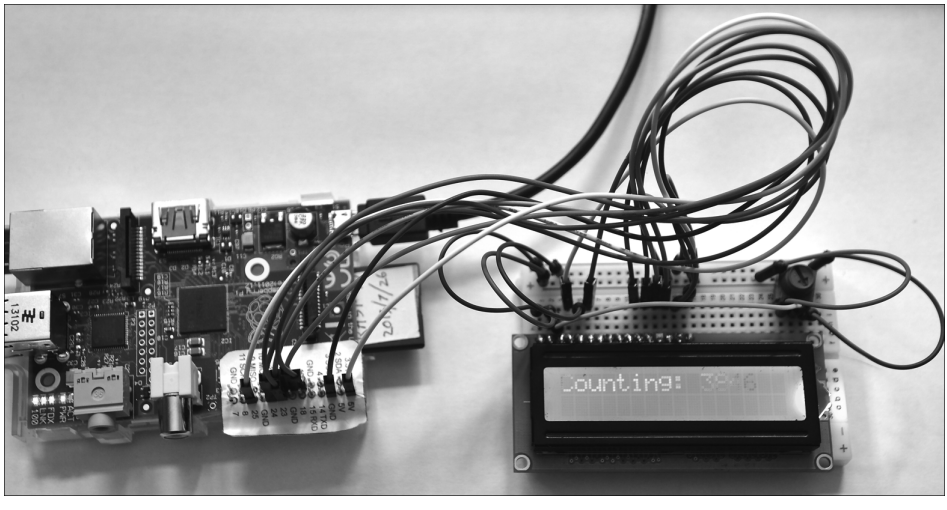

Рис. 14.5. Жидкокристаллический дисплей (16×2), подключенный к Raspberry Pi

Устройства этой категории бывают разных размеров, а одна из основных их характеристик указывает количество символов и текстовых строк, одновременно отображаемых на экране. Например, описанный в текущем рецепте дисплей имеет размер 16×2, что указывает на вывод информации на нем в две строки, каждая из которых может содержать максимум 16 символов. Существуют ЖК-дисплеи с такими размерами экранов: 8×1, 16×1, 16×2, 20×2 и 20×4.

Для выполнения текущего проекта понадобятся следующие компоненты:

- плата беспаечного монтажа и проволочные перемычки (см. приложение А);
- ЖК-дисплей, основанный на драйвере HD44780, с размером экрана 16×2 (см. приложение А);
- штыревая контактная колодка на 16 выводов;
- подстроечный резистор с максимальным сопротивлением 10 кОм (см. приложение А).

Схема подключения дисплея к Raspberry Pi изображена на рис. 14.6. Обычно модули дисплеев оснащаются не штыревыми выводами, а одними только контактными площадками. Вам придется приобрести такую штыревую колодку отдельно и припаять ее к контактной площадке платы дисплея вручную.

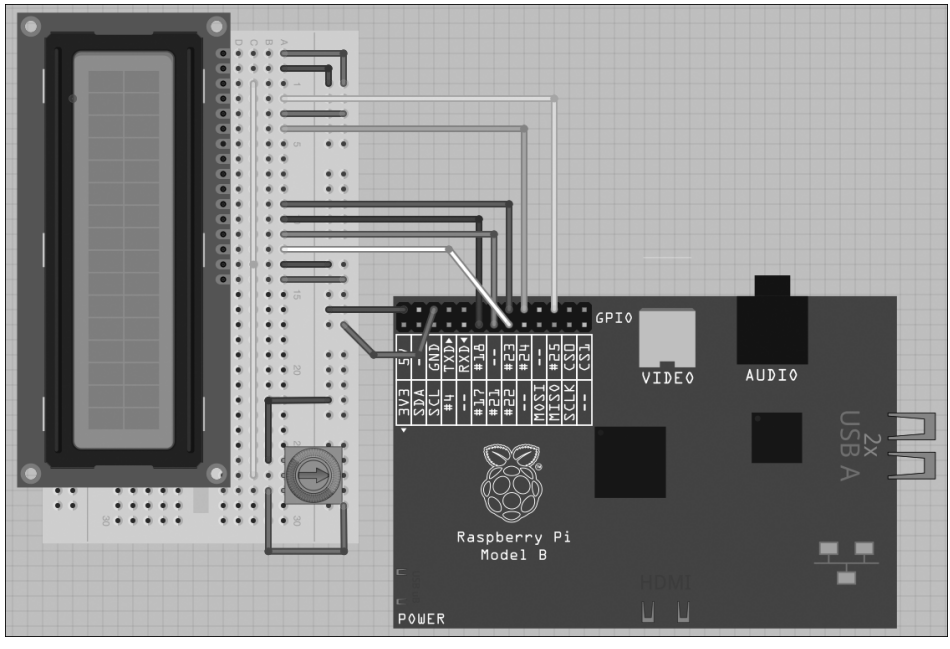

Рис. 14.6. Схема подключения простого дисплея к Raspberry Pi

Подстроечный резистор включен в схему, чтобы предоставить возможность регулировать контраст дисплея. Часто кажется, что на экран дисплея ничего не

выводится, но если покрутить регулятор подстроечного резистора во всем диапазоне, то текст проявится максимально четко. Конечно, можно обойтись и без такого резистора, но наперед никогда не угадаешь.

На GitHub вы найдете библиотеку, обеспечивающую управление ЖК-дисплеями на основе драйвера HD44780, а также несколько тестовых программ вывода на них текста. Перед ее установкой выполните рецепт 9.2 и импортируйте в систему библиотеку `RPi.GPIO`.

Компания Adafruit снабжает свои дисплеи специальным программным модулем, обеспечивающим доступ ко всем его функциональным средствам. Он устанавливается не так, как большинство общедоступных библиотек Python. Отличие состоит в необходимости перенастройки его файловой структуры.

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

Измените название конечного каталога, содержащего код библиотеки.

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

В этом каталоге находится тестовая программа, выводящая на экран дисплея системное время и IP-адрес Raspberry Pi. В случае использования Raspberry Pi модели B второй ревизии перед запуском приведенной ниже программы нужно отредактировать файл `Adafruit_CharLCD.py`. Выполните такую команду:

```
$ nano Adafruit_CharLCD.py
```

Найдите в файле следующую строку:

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

Замените число 21 числом 27 и сохраните файл. Указанная строка должна принять показанный ниже вид.

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

Запустите тестовую программу, выполнив такую команду:

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

Обсуждение

В простых ЖК-дисплеях шина передачи данных либо четырех-, либо восьмибитовая, что требует использования трех управляющих выводов. Назначение всех выводов ЖК-дисплея описано в табл. 14.1.

Таблица 14.1. Выводы, принимающие участие в подключении ЖК-дисплея к Raspberry Pi

Вывод дисплея	Вывод GPIO	Назначение
1	GND	0 В (земля)
2	+5V	Питание микросхем логики
3	—	Управление контрастом
4	25	RS: выбор регистра
5	GND	RW: чтение/запись (всегда запись)
6	24	EN: Вкл.
7–10	—	Используется только в восьмибитовом режиме
11	23	D4: линия передачи данных 4
12	17	D5: линия передачи данных 5
13	21	D6: линия передачи данных 6
14	22	D7: линия передачи данных 7
15	+5V	Питание лампы подсветки
16	GND	Земля лампы подсветки

За установку правильных сигналов на выводах дисплея и передачу в него данных отвечает библиотека `Adafruit_CharLCD.py`, предоставляемая в открытый доступ компанией Adafruit. Кроме всего прочего, она снабжает вас следующими функциями управления модулем дисплея, которые можно применять в пользовательских проектах.

- `home()`. Перемещение курсора влево.
- `clear()`. Очистка дисплея от текста.
- `setCursor(`*`столбец`*`, `*`строка`*`)`. Перемещение курсора в позицию ввода текста.
- `cursor()`. Режим отображения курсора.
- `noCursor()`. Режим отключения курсора (по умолчанию).
- `message(text)`. Ввод текста в текущем положении курсора.

В приведенном ниже простом примере показано, как, используя средства библиотеки `Adafruit_CharLCD.py`, вывести на дисплей простое сообщение.

```
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
```

Дополнительные сведения

У Adafruit имеется плата расширения, на которой смонтирован описанный в текущем рецепте дисплей. Все приведенные выше выкладки справедливы и для него.

14.6. Графический OLED-дисплей

Задача

Подключить графический OLED-дисплей (дисплей на органических светодиодах) к Raspberry Pi.

Решение

Подключите OLED-дисплей (рис. 14.7), управляемый драйвером SSD1306, к Raspberry Pi через I2C-соединение.

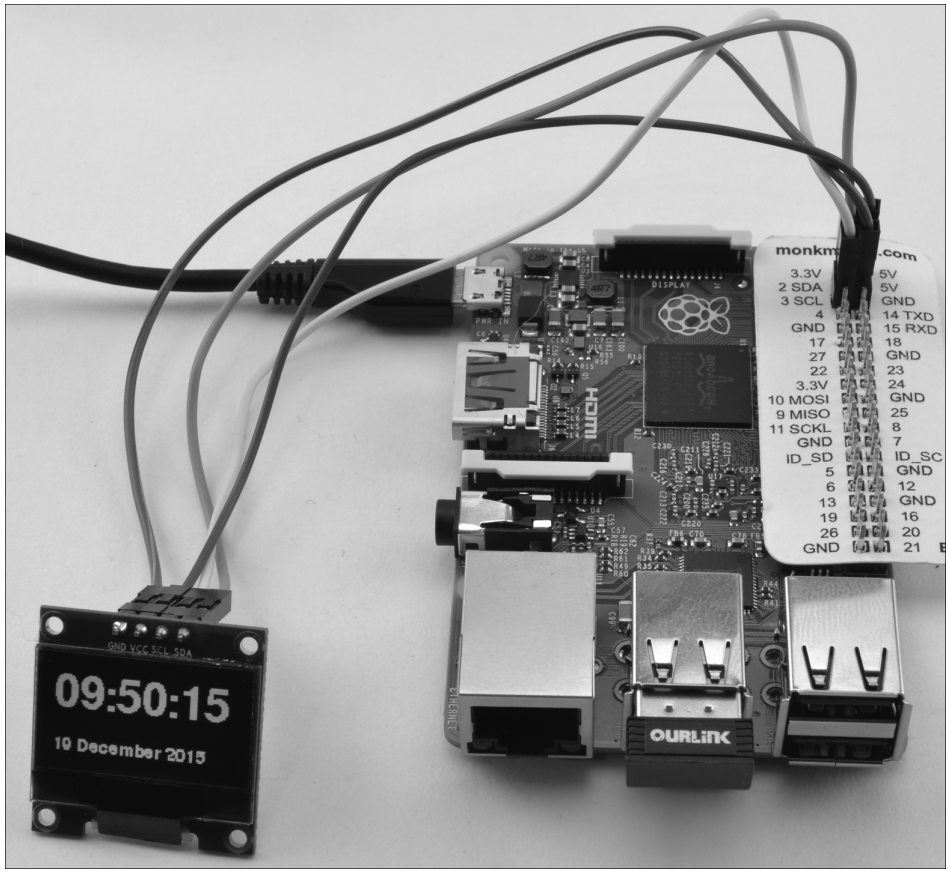

Рис. 14.7. OLED-дисплей, данные на который передаются через интерфейс I2C

Для выполнения текущего проекта понадобятся следующие компоненты:

- проволочные перемычки только со штыревыми разъемами на концах (см. приложение А);
- ЖК-дисплей, снабженный интерфейсом I2C, с размером экрана 128×64 (см. приложение А).

Некоторые из таких дисплеев оснащены всего четырьмя выводами, остальные — целыми восемью. Восьмивыводные модели поддерживают работу не только с интерфейсом I2C, но и с последовательным портом SPI. К сожалению, обладателям четырехвыводных дисплеев приходится довольствоваться передачей данных только через соединение I2C. К счастью, интерфейс I2C лучше подходит для управления дисплеем в нашем проекте, поскольку питание он получает от источника с напряжением 5 В, а на его выводы SDA и SCL (интерфейса I2C) подается сигнал с высоким уровнем 3,3 В, что полностью согласуется с возможностями Raspberry Pi. В то же время использование интерфейса SPI сопряжено с необходимостью дополнительного приведения сигналов к единому уровню.

Соедините следующие выводы перемычками, чтобы физически подключить устройства одно к другому:

- вывод VCC дисплея соедините с выводом 5V интерфейса GPIO платы Raspberry Pi;
- вывод GND дисплея соедините с выводом GND интерфейса GPIO платы Raspberry Pi;
- вывод SDA дисплея соедините с выводом GPIO 2 (SDA) платы Raspberry Pi;
- вывод SCL дисплея соедините с выводом GPIO 3 (SCL) платы Raspberry Pi.

Для реализации рецепта требуется включить в Raspberry Pi поддержку последовательного интерфейса I2C. О том, как это сделать, рассказывалось в рецепте 9.3.

Компания Adafruit снабжает свои дисплеи специальной программной библиотекой, обеспечивающей доступ ко всем его функциональным средствам. Установите ее, выполнив следующие три команды.

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

В этой библиотеке используются программные инструменты другой библиотеки — `PIL` (Python Image Library). Установите последнюю так, как показано ниже.

```
$ sudo pip install pillow
```

Приведенная ниже программа, сохраненная в файле `old_clock.py`, выводит на OLED-дисплей текущие дату и время. Как и любые другие примеры программ, она доступна для загрузки на сайте книги (см. раздел Code).

```
from oled.device import ssd1306
from oled.render import canvas
from PIL import ImageFont
import time
from datetime import datetime

# Настройка дисплея
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)

# Вывод сообщения в трех строках, первая строка — крупным шрифтом
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)
```

Каждое ведомое устройство в интерфейсе I2C имеет свой уникальный адрес, который для дисплея устанавливается следующим образом:

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

Будьте внимательны! Несмотря на то что у большинства простых дисплеев этот адрес определяется шестнадцатеричным значением `3C`, конкретно в вашей модели он может отличаться, поэтому обязательно сверьтесь с документацией к устройству или используйте специальные инструменты диагностики (см. рецепт 9.4). С помощью последних вы обнаружите все подключенные через интерфейс I2C устройства и узнаете назначенные им адреса.

Обсуждение

Компактные OLED-дисплеи подкупают невысокой ценой, низким энергопотреблением и высоким разрешением. Они успешно вытесняют обычные ЖК-дисплеи в устройствах любительской электроники.

Дополнительные сведения

В рецепте приведены инструкции для 4-выводного OLED-дисплея. Подключение 8-выводного устройства к Raspberry Pi через последовательное соединение SPI описано в руководстве Adafruit, приведенном по такому адресу:

`https://learn.adafruit.com/ssd1306-oled-displays-with-raspberry-pi-and-beaglebone-black`

14.7. Цветная светодиодная лента

Задача

Подключить ленту RGB-светодиодов (Neopixels) к Raspberry Pi.

Решение

Используйте цветную светодиодную ленту, управляемую контроллером WS2812.

Самый простой вариант заключается в подключении светодиодной ленты непосредственно к выводам GPIO (рис. 14.8), с которых светодиоды и получают питание. Но у этого подхода есть свои недостатки, о которых нужно знать и которые нужно уметь обходить, чтобы обеспечить полную работоспособность светодиодного устройства (детальнее об этом рассказано в разделе “Обсуждение”).

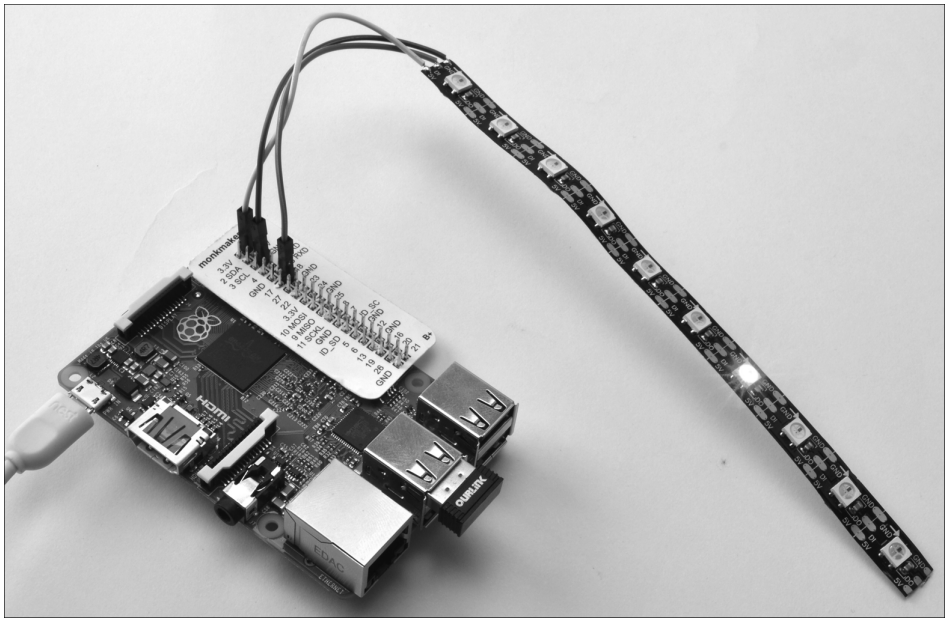

Рис. 14.8. Подключение ленты из 10 светодиодов

Не подавайте питание на светодиоды с вывода 3.3V

Идея подачи питания на светодиоды с вывода 3.3V порта GPIO кажется очень заманчивой, но старайтесь всячески избегать такого способа подключения ленты. Если через такой вывод потечет слишком большой ток (см. рецепт 9.2), то плата Raspberry Pi гарантировано выйдет из строя.

На рис. 14.8 показано, как к Raspberry Pi можно подключить фрагмент ленты, отрезанный от большой бобины. Поскольку каждый светодиод потребляет максимальный ток 60 мА, простые вычисления показывают, что от одного вывода GPIO можно запитать не более 10 светодиодов. В остальных случаях для электропитания ленты нужен отдельный источник (см. раздел "Обсуждение").

Чтобы подключить отрезок светодиодной ленты к Raspberry Pi, припаяйте к ее дорожкам GND, DI и 5V три проволочные перемычки, заканчивающиеся на противоположном конце штыревыми разъемами. Накиньте штыревые разъемы перемычек на соответствующие выводы GND, GPIO 18 и 5V интерфейса GPIO.

Обратите внимание на стрелку, нанесенную на поверхность ленты (рис. 14.9). Следите за тем, чтобы перемычки паялись к срезу, по направлению от которого указывает стрелка.

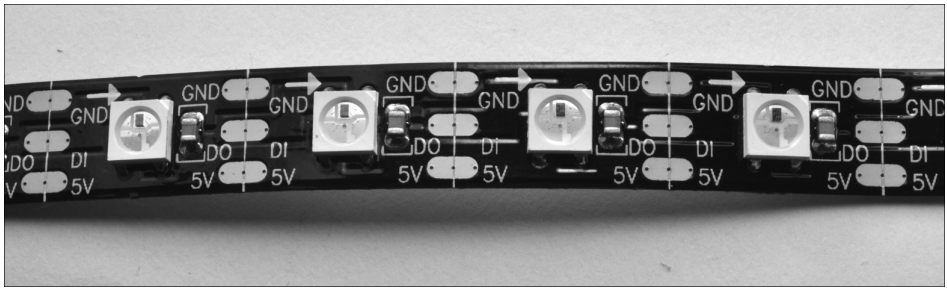

Рис. 14.9. Светодиодная лента крупным планом

Поиск библиотеки, обеспечивающей управление светодиодной лентой, основанной на драйвере WS2812, оказался не таким уж и простым делом. Исходно она разрабатывалась Джереми Гарффом, а ее адаптированная для Raspberry Pi 2 версия создавалась Ричардом Херстом. К развитию библиотеки также имели отношение специалисты компании Adafruit. Но переход операционной системы Raspbian на ядро Linux Kernel 4.1.6 свел на нет все их труды. На момент написания книги лучше всего с управлением светодиодными лентами справлялся модуль расширения Unicorn HAT компании Pimoroni (программное управление им основано на библиотеке, разработанной Ричардом Херстом). На данный момент именно эта библиотека считается наиболее надежной для использования в платах Raspberry Pi. Чтобы установить ее в Raspbian, выполните следующие команды.

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

Приведенная ниже программа, сохраненная в файле `led_strip.py`, последовательно зажигает все имеющиеся на ленте красные светодиоды. Как и любые другие примеры программы, она доступна для загрузки на сайте книги (см. раздел Code).

```
import time
from neopixel import *

# Конфигурация светодиодов на ленте:
LED_COUNT = 10         # Общее количество
LED_PIN = 18           # GPIO-вывод подключения
                       # (должен поддерживать ШИМ!)
LED_FREQ_HZ = 800000   # Частота управляющего сигнала в герцах
                       # (обычно 800 кГц)
LED_DMA = 5            # Канал DMA для сигнала (обычно 5)
LED_BRIGHTNESS = 255   # 0 — не светится, 255 — максимальная яркость
LED_INVERT = False     # Для инвертирования сигнала — True

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
```

Параметры раздела `Конфигурация светодиодов на ленте`: подлежат перенастройке согласно типу светодиодной ленты, имеющейся в наличии. В частности, если на отрезке ленты расположено другое количество светодиодов (отличное от 10), то измените константу `LED_COUNT`. Остальные константы можно не изменять.

Цвет каждого светодиода устанавливается отдельно с помощью метода `set PixelColor()`. Его первый аргумент указывает положение светодиода на ленте (счет начинается с нуля), а вот цвет свечения определяется вторым аргументом. За обновление рабочих режимов светодиодов отвечает метод `show()`.

Обсуждение

Как вы знаете, через светодиод может протекать максимальный ток, равный 60 мА. Он достигает такого уровня только в случае свечения всеми тремя каналами (красный, зеленый, синий) с максимальной яркостью (255). Если к Raspberry Pi планируется подключить ленту с большим количеством светодиодов, то ее нужно снабдить отдельным источником питания с выходным напряжением 5 В. Схема устройства, в котором реализован данный метод подключения светодиодной ленты, показана на рис. 14.10. Вам понадобится специальный переходник с винтовыми зажимами для разъема стандартного источника питания (см. приложение А). Без него подключить блок питания к плате беспаечного монтажа весьма проблематично.

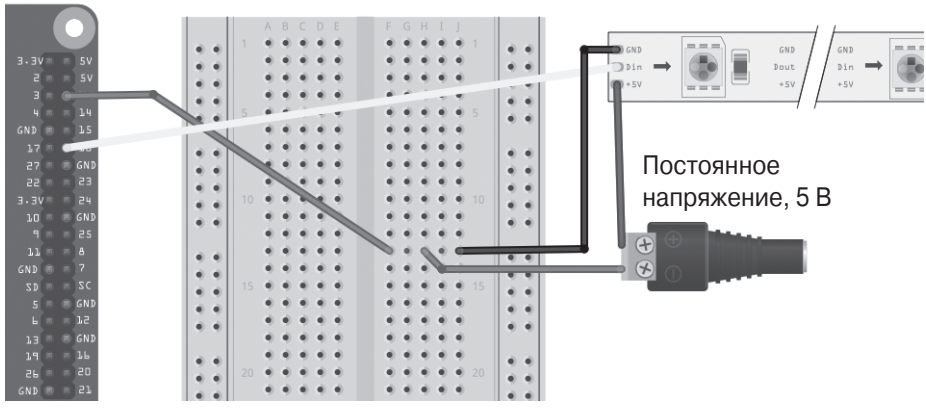

Рис. 14.10. Подача питания на светодиодную ленту с внешнего источника

В технических характеристиках драйвера WS2812 ленты указано, что высокий уровень управляющего сигнала должен составлять 4 В при подаче на светодиоды питания от источника с выходным напряжением 5 В. Нет никакой гарантии, что подача на ленту сигнала с выводов GPIO, имеющих логический уровень всего 3,3 В, позволит надежно управлять ею. Тем не менее в большинстве случаев особых проблем с управлением предложенной схемой не наблюдается.

Как бы там ни было, испытывая затруднения с управляющим сигналом, постройте простую схему преобразователя уровней, показанную на рис. 14.11. В ней использован транзистор 2N7000 (транзистор 2N3904 использовать нельзя из-за низкого быстродействия — он попросту не успевает инвертировать импульсы).

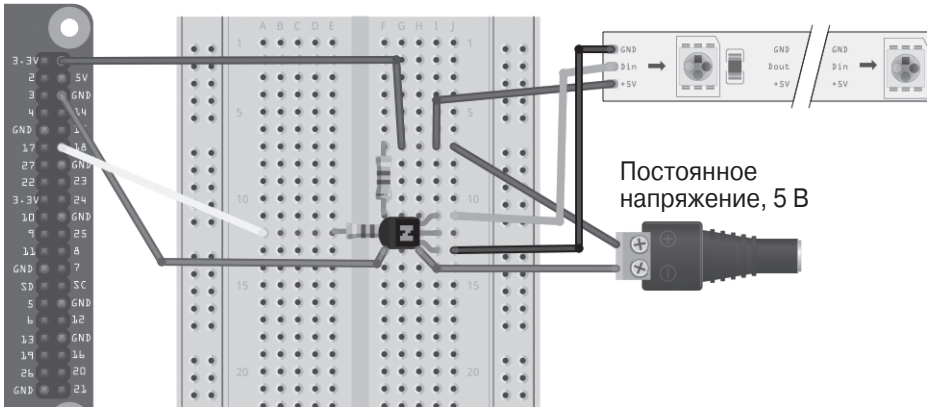

Рис. 14.11. Преобразователь уровней

Данный преобразователь не только изменяет уровень сигнала с 3,3 до 5 В, но и инвертирует его. Таким образом, при подаче на вывод GPIO сигнала высокого уровня на светодиодную ленту приходит сигнал низкого уровня и наоборот. Исходя из этого, в программе нужно изменить сигнальную логику. Замените строку

```
LED_INVERT = False
```

строкой

```
LED_INVERT = True
```

Дополнительные сведения

Повысить уровень сигнала с 3,3 до 5 В можно с помощью не только транзистора, но и специального модуля расширения (см. рецепт 9.13).